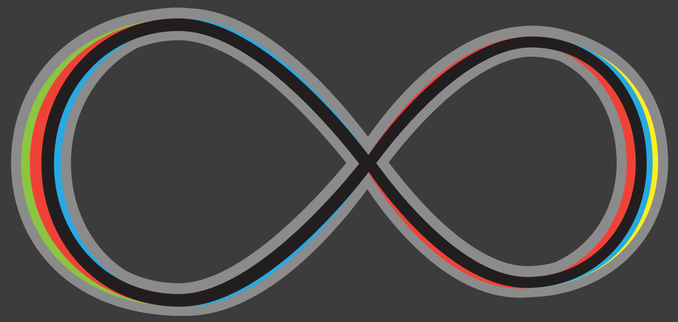


DFINITY

The Decentralized Cloud

A Brief Tour of DFINITY

Doc vers. 3rd January 2017

DFINITY

Mission

Extend Ethereum Ecosystem With Cloud Network

ethereum

DFINITY

Extreme availability

Casper

Scaling : Speed : Public-Private IOP

crypto:3

Ungoverned

The Code is Law

Governed

Blockchain Nervous System

A Boon To Ethereum

~~Virtual~~ Machine Battle

PowerPC

Old co Apple

The Macintosh

8086

Microsoft, Dell, IBM, HP... Apple

Sparc

Sun Microsystems

SPARCstation 5

Compatibility More Popular!

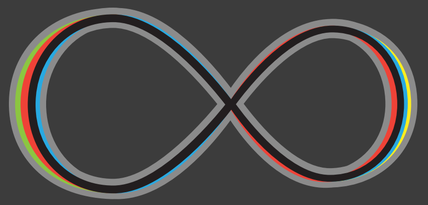
Virtual Machine Battle

Not compatible

EVM

The
Code is Law

The
Decentralized Cloud



Not compatible

Compatibility More Popular!

DFINITY Foundation

- Enable decentralized cloud computing
- Promote compatibility for Dapp developers
- Advance the Ethereum ecosystem

- **Fund and advance core Ethereum technologies**
  e.g. Ethereum Virtual Machine, Solidity, state channels, etc
- **Fund and advance novel DFINITY technologies**
  e.g. crypto:3 protocols, Blockchain Nervous System, etc

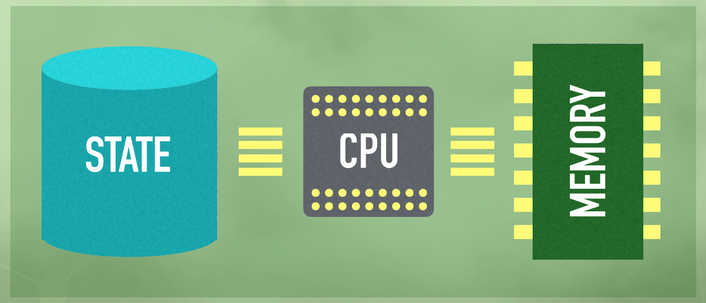
DFINITY

Core Vision

The Decentralized Cloud

A Revolutionary Limitless Virtual Computer

created by a decentralized network protocol rather than servers



Hmm... doesn't that means software systems running on the virtual machine might be protocols too?

DFINITY

The Decentralized Cloud Can Crush IT Costs

COSTS

IT system built on traditional cloud e.g. AWS

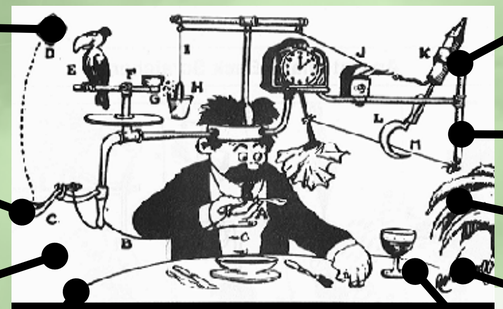
COSTS

IT system built on decentralized cloud

That's incredible. The decentralized cloud involves much more computation but it can save the world billions of dollars

DFINITY

IT Systems Should Not Be Rube Goldberg Machines



Using AWS/Amazon

lication Services
Step Functions
zon API Gateway
zon Elastic Transcoder
zon AppStream

Management Tools
Amazon CloudWatch
Amazon EC2 Systems Manager
AWS CloudFormation
AWS CloudTrail
AWS Config
AWS OpsWorks
AWS Service Catalog
Trusted Advisor
AWS Personal Health Dashboard
AWS Command Line Interface

Messaging
Amazon SQS
Amazon Pinpoint
Amazon SES
Amazon SNS

Storage
Amazon Simple Storage Service (S3)
Amazon Elastic Block Storage (EBS)
Amazon Elastic File System (EFS)
Amazon Glacier
AWS Storage Gateway
AWS Snowball
AWS Snowball Edge
AWS Snowmobile

Compute
Amazon EC2
Amazon EC2 Container Registry
Amazon EC2 Container Service
Amazon Lightsail
Amazon VPC
AWS Batch
AWS Elastic Beanstalk
AWS Lambda
Auto Scaling
Elastic Load Balancing

Database
Amazon Aurora
Amazon RDS
Amazon DynamoDB
Amazon ElastiCache
Amazon Redshift
AWS Database Migration Service

tworking & Content Delivery
azon VPC
Amazon CloudFront
Amazon Route 53
AWS Direct Connect
Elastic Load Balancing

Security & Identity, Compliance
AWS Identity and Access Management (IAM)
Amazon Inspector
AWS Certificate Manager
AWS CloudHSM
AWS Directory Service
AWS Key Management Service
AWS Organizations
AWS Shield

Developer Tools
AWS CodeCommit
AWS CodeBuild
AWS CodeDeploy
AWS CodePipeline
AWS X-Ray
AWS Command Line

DFINITY

Code Business Logic Not Systems

MetaCoin.sol

```
pragma solidity ^0.4.4;

import "ConvertLib.sol";

// This is just a simple example
// It is not standards compatible
// coin/token contracts. If you w
// token, see: https://github.com/

contract MetaCoin {
    mapping (address => uint) bal

    event Transfer(address _from,

    function MetaCoin() {
        balances[tx.origin] = 100
    }

    function sendCoin(address rec
        if (balances[msg.sender]
        balances[msg.sender] -= a
        balances[receiver] += amou
        Transfer(msg.sender, rece
        return true;
    }

    function getBalanceInEth(addr
        return ConvertLib.convert(
    }

    function getBalance(address a
        return balances[addr];
    }
}
```

app.js

```
var accounts;
var account;

function setStatus(message) {
  var status = document.getElementById("status");
  status.innerHTML = message;
};

function refreshBalance() {
  var meta = MetaCoin.deployed();

  meta.getBalance.call(account, {from: account}).then(function(value) {
    var balance_element = document.getElementById("balance");
    balance_element.innerHTML = value.valueOf();
  }).catch(function(e) {
    console.log(e);
    setStatus("Error getting balance; see log.");
  });
};

function sendCoin() {
  var meta = MetaCoin.deployed();

  var amount = parseInt(document.getElementById("amount").value);
  var receiver = document.getElementById("receiver").value;

  setStatus("Initiating transaction... (please wait)");

  meta.sendCoin(receiver, amount, {from: account}).then(function() {
    setStatus("Transaction complete!");
    refreshBalance();
  }).catch(function(e) {
    console.log(e);
    setStatus("Error sending coin; see log.");
  });
};

window.onload = function() {
  web3.eth.getAccounts(function(err, accs) {
```

Abstraction Wins

In the GUI code directly against persistent smart contract objects as though they were libraries. No server. No database. No scaling. Just simple UX and business logic...

Decentralized Cloud Properties

Autonomous | systems w/o dependencies

Simple | code processing not components

Verifiable | know the code you interact with

Shareable | share code and governance

Unstoppable | no servers to fail

Tamperproof | no servers to meddle with

Cyberspace | no servers so no geography

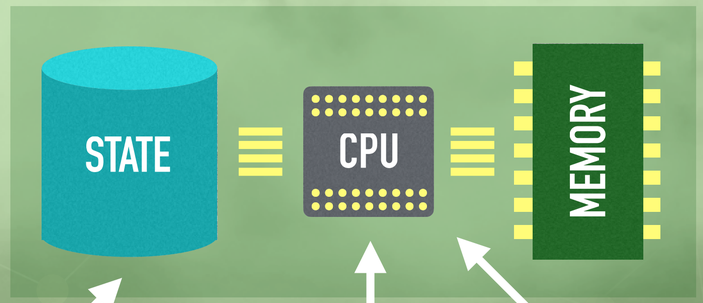
Interoperability | no server boundaries

DFINITY

Decentralized Core Public Services

“Generic” Business Systems And Their Data Embedded

as open protocols within fabric of Internet



Twitter

ebay

Uber

DFINITY

Decentralized Core Public Services

Sharing

Micro-blogging

“Email”

Search

Storage

DFINITY was originally founded to provide the massive scalability needed

DFINITY

Root Level Business Reengineering

Combine DFINITY with Interoperable Private Cloud Computing Networks

urities & Trading | Supply chain & Trade finance | Intellectual property management | Logistics & Asset Tracking | Charity | Educat

Legal | Real Estate & Property | Internet of Things | Financial markets and exchanges | Electronic medical records | Digital right

Smart property | Workforce | Collaborative economy | Voting & Governance Systems | Insurance | Global identity | Provena

ss-border finance | Customer Acquisition & Loyalty | Escrow & Custody services | Identity management | Auditing & Compliance

DFINITY

Decentralization of Core Functions

string

PHI Decentralized Commercial Banking

Similarly to commercial banks, PHI originates new loans by creating new stable cryptocurrency that piggybacks the local currency. Loans granted using a “random validator chain” technique. Can originate loans more judiciously at a fraction of the cost...

DFINITY

Scale-out

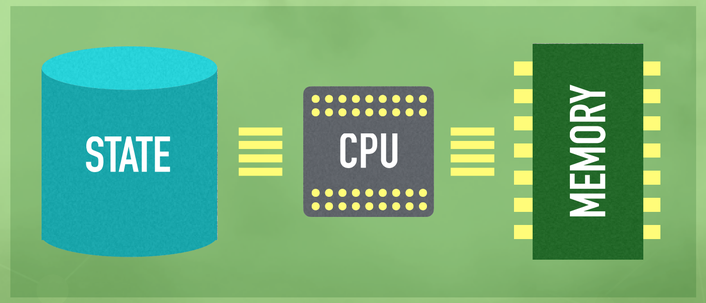
By applying crypto:3 techniques

crypto:3 Scale-out

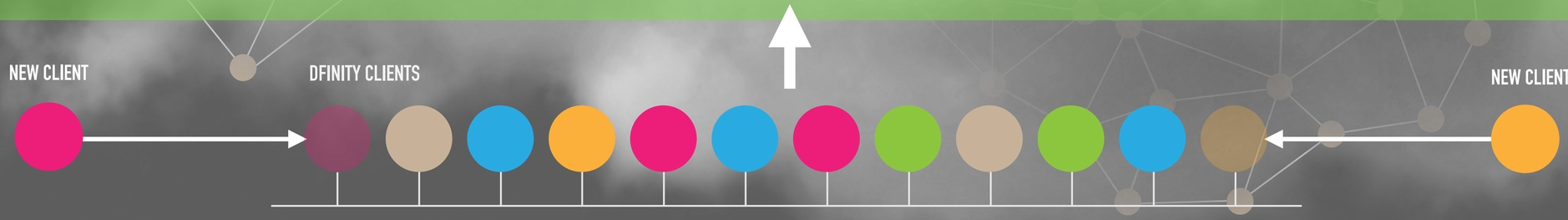
A Network of DFINITY Mining Clients Run crypto:3 Protocols

virtual computer formed from protocol interactions

STORAGE CAPACITY: 1.9 EXABYTES



PROCESSING CAPACITY: 583473 GAS



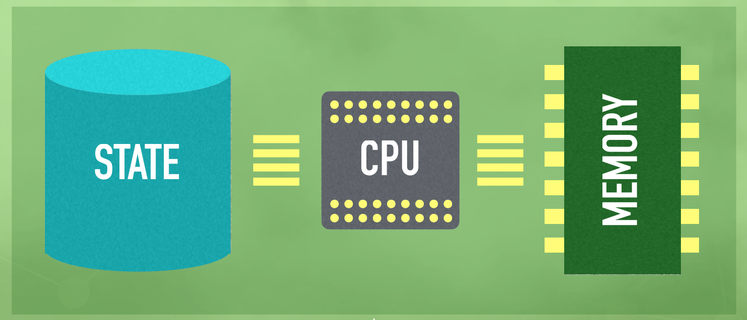
DFINITY

crypto:3 Scale-out

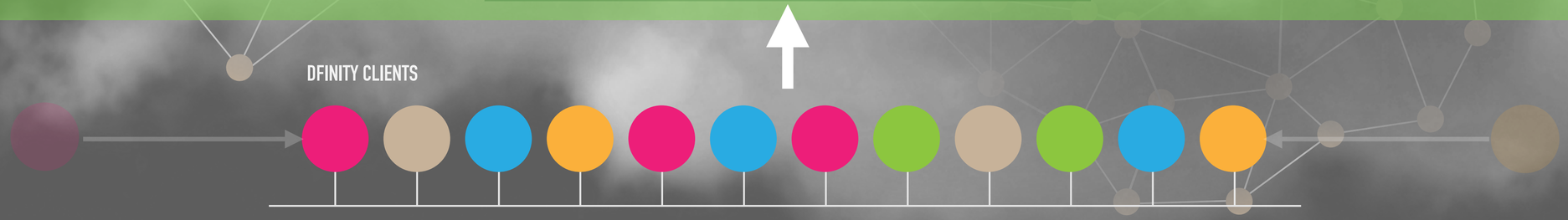
As New Mining Clients Join The Network...

computational and storage capacity grows indefinitely

STORAGE CAPACITY: 2.1 EXABYTES



PROCESSING CAPACITY: 731011 GAS



DFINITY

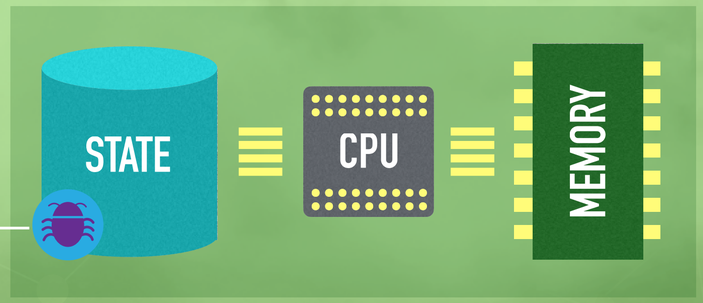
Blockchain Nervous System

Decentralized governance

Blockchain Nervous System AI

“Code is Law” Now Contingent Upon a Decentralized Intelligence

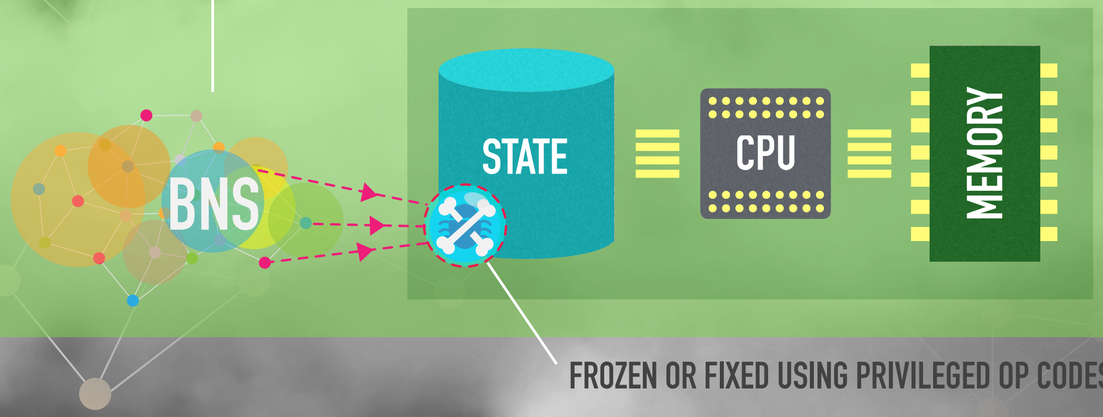
this governance system uses privileged machine instructions...



DFINITY

Blockchain Nervous System AI

FIX DEADLOCKS | MITIGATE HACKS | RETURN FUNDS | PROTOCOL UPGRADES | FREEZE VICE/VIOLENCE | CONFIG



DFINITY

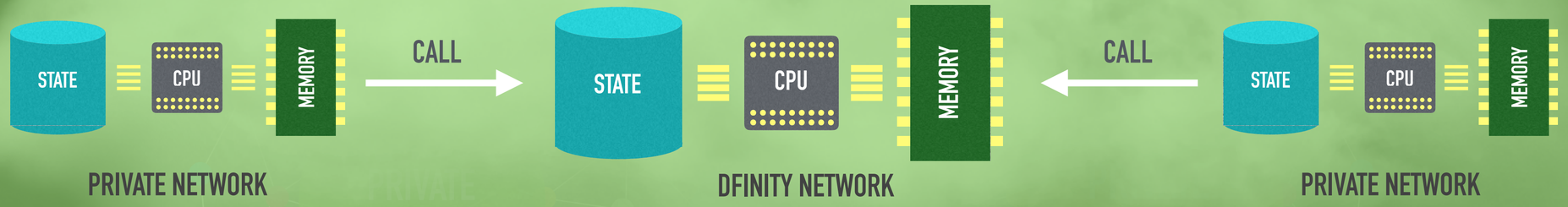
Public-Private Cloud System Interoperability

Made possible by crypto:3 network technology

Public-Private Interoperability

Private Network Software System Calls Into Public Software System

ZINC RELEASE - one top-level private method can be twinned with a public method



DFINITY

Public-Private Interoperability

Corporates Build Quickly Using Open Business Systems Like Software Libraries

e.g. incorporate stable coin, identity, arbitration, interact through an exchange...

DFINITY

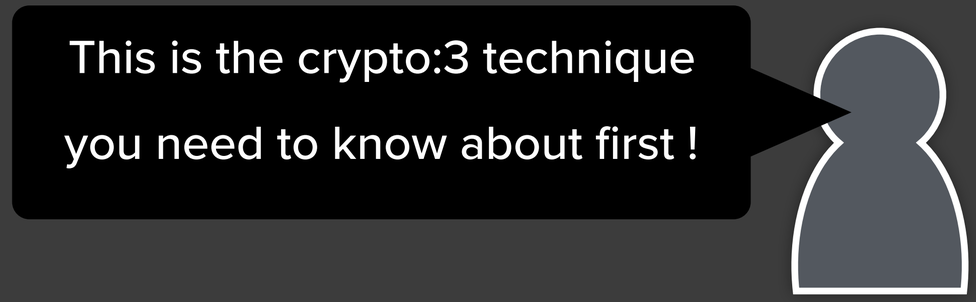
Review Technicalities...

1. Threshold Relay
2. PSP blockchain (Threshold Relay)
3. Applications of Randomness
4. Blockchain Nervous System
5. Public-Private Interoperability

1

Threshold Relay

Unmanipulable, unpredictable randomness on demand in decentralized networks



Protocol Design Challenge

How to organize 1M+ mining clients to produce 1 massive virtual computer (DFINITY cloud)?

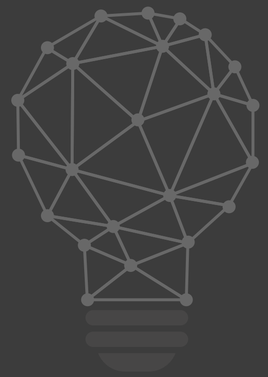
Composed 1M+ servers

Composed 1M+ servers

Start with randomness...

OBSERVATION

Randomness is the fundamental engine used to drive stateful decentralized networks

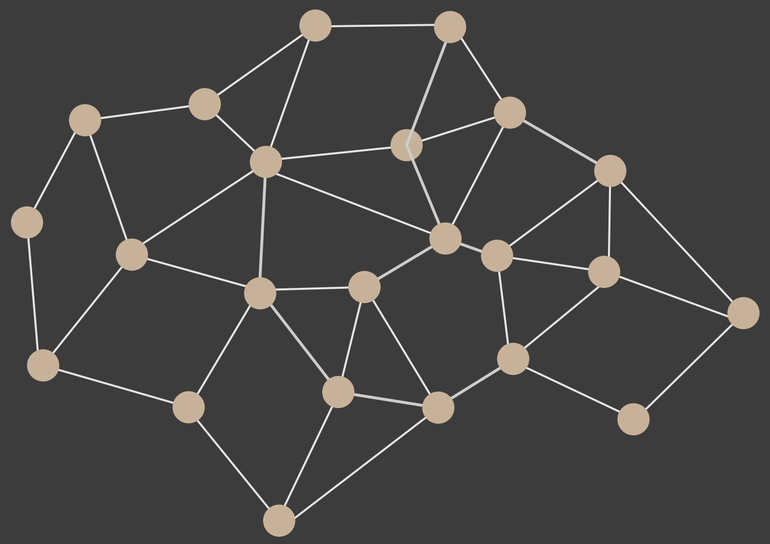


FOR EXAMPLE...

Proof of Work

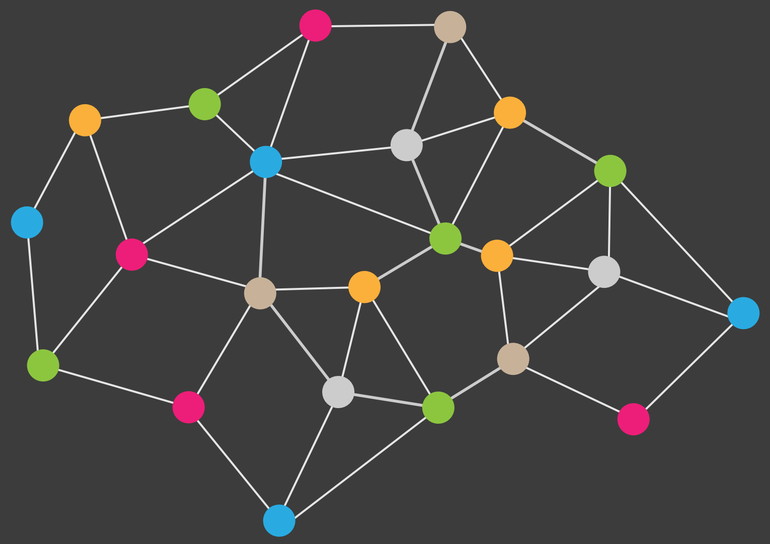
- **Miners race to solve a current puzzle**
  - Solutions found randomly in Poisson distribution
- **Winner appends block to blockchain**
  - In effect was elected a temporary “leader”
- **Next leader is unknown**
  - Cannot DOS, manipulate etc.
- **Honest majority**
  - Chain functions correctly if majority of leaders are honest (~ since selfish mining...)
  - Adversary cannot control chain

A network of processes...



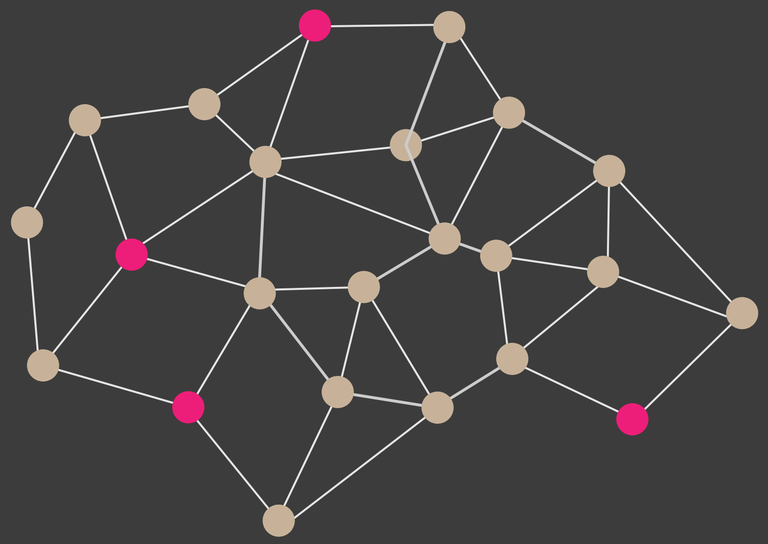
- **Mining “client software” is a process**
  - Fundamental unit of computational resource
- **Connected in P2P broadcast net**
  - E.g. gossip based. Can use Kademlia structure
- **Each process has “mining identity”**
  - Public key with meta data attached
- **IDs mediate participation**
  - Private network: trusted dealer defines list
  - Public network: CC security deposit, USCIDs
- **Massive network size**

is organized into random groups...



- **Random members**
  - Each process is a member of multiple groups
  - Groups intersect, have e.g. 400 members
- **Groups setup threshold crypto**
  - Run VSS secret sharing protocol
  - 51% threshold e.g. 201+/400 needed create signature
- **BLS signature scheme**
  - **Math magic...** If 51% of group members broadcast “signature shares” on a message, these can be combined to create the group’s threshold signature. This will always be the same irrespective of which 51% subset signs (the system is “unique and deterministic”)

current group signs... signature of previous group



- **Signature *is* random number**
  - Otherwise it would be predictable/insecure
- **Number *se/ects* next group**
  - g = G[r % |G|]
- **Next group use prev no. as message**
  - Thus sequence is entirely deterministic
- **Verifiable Random Function**
  - Numbers verifiable using group public key
  - New values produced on threshold agreement
  - Unmanipulable, unpredictable...

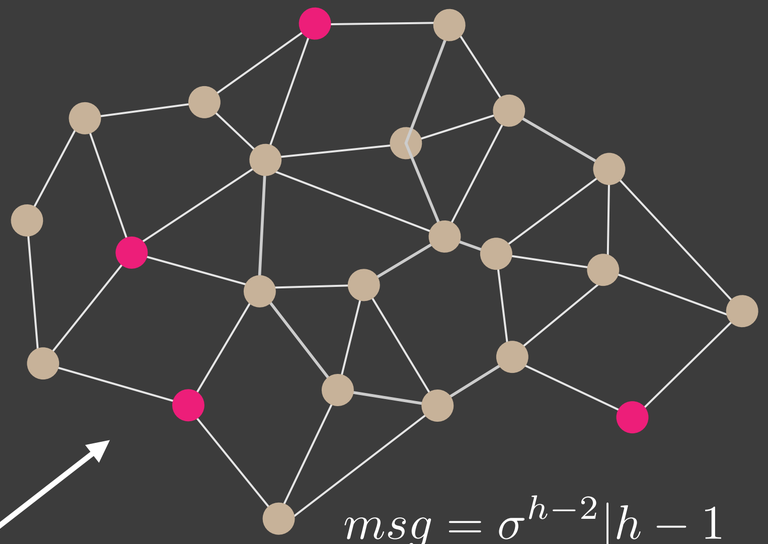
LEGEND BY EXAMPLE

σ^h Threshold signature at height h (h-1 signature used as message)

σ_p^h Signature “share” on h-1 threshold signature by process p

G^h Threshold group that will sign at height h

$|\mathcal{G}|$ The total number of threshold groups in network



$msg = \sigma^{h-2}|h-1$

ECTS NEXT GROUP

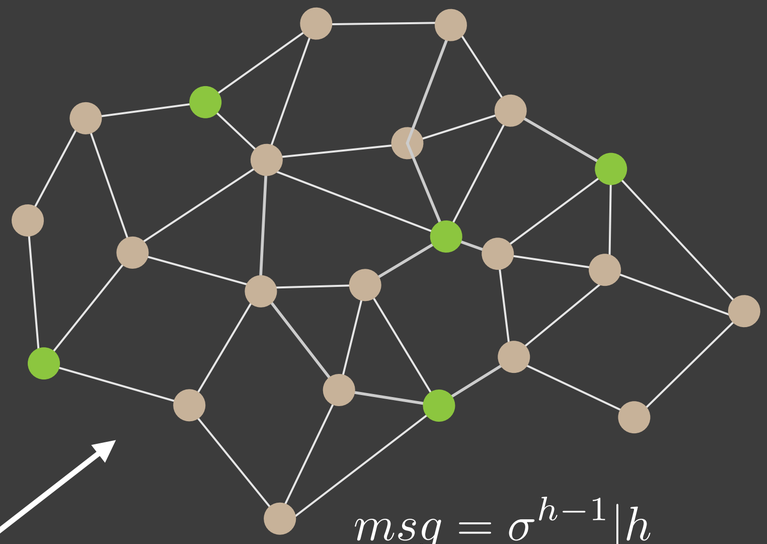
$\sigma^{h-2} \bmod |\mathcal{G}|]$

NEW RANDOM NO. (BLS THRESHOLD SIGNATURE)

$\sigma^{h-1} = bls\_ts(\{\sigma_p^{h-1}, p \in G^{h-1}\})$

SELECTS NEXT GROUP

$G^h = \mathcal{G}[\sigma^{h-1} \bmod |\mathcal{G}|]$



$msg = \sigma^{h-1}|h$

NEW RANDOM NO. (BLS THRESHOLD SIGNATURE)

$\sigma^h = bls\_ts(\{\sigma_p^h, p \in G^h\})$

AD INFINITUM

BLOCK HEIGHT

$h-1$ h

Overhead Example

MESSAGE FORMAT

Process ID	20 bytes
Signature share	32 bytes
Signature on comms	32 bytes
Total	84 bytes

When a group must sign the previous signature, each member process creates a signature share using it as the message. This must be broadcast together with some other information

GROUP SIZE

Group size	400
Threshold	201

COMMUNICATION OVERHEAD

Maximum	*only* 34 KB

If all group members are active, a total of 34 KB messages will be created each round. In practice, broadcast halts as soon as the 32 byte group signature is broadcast (requires 17 KB of messages)

Resilience Example

NETWORK MAKEUP

Processes	10,000
Faulty	3,000
(Correct)	7,000
Group Size	400
Threshold	201

Note: our example assumes almost one third mining processes in the network are faulty. In practice this would be an extreme situation where professional mining is involved.

Probability that 200 or more processes in randomly selected group are faulty, preventing production of signature:

1e-17

Calculate odds using any hypergeometric probability calculator

http://www.geneprof.org/GeneProf/tools/hypergeometric.jsp

Nb. groups expire to address “adaptive” adversaries

2

Threshold Relay Blockchain

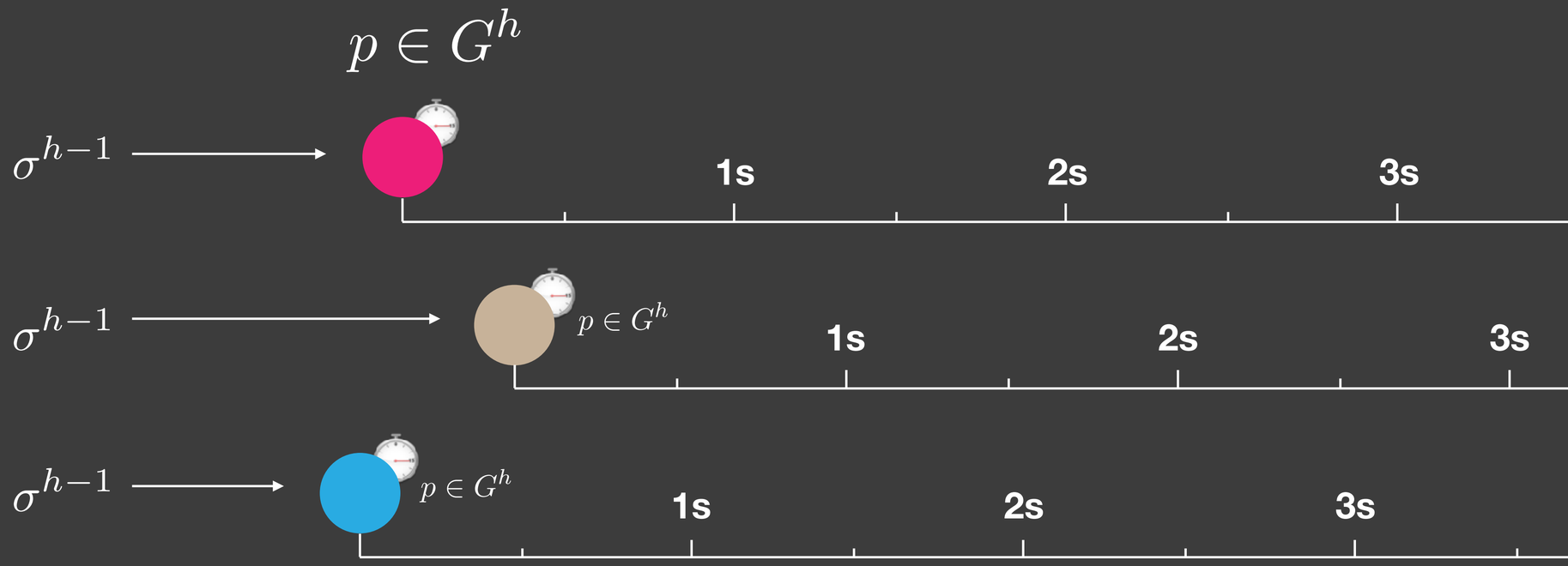
Probabilistic Slot Protocol (PSP)

Upon Selection Group Members Start Their Timers

and wait for the current Block Time to expire...

Waiting For The "Block Time"

When a new group is selected by the previous group's threshold signature (the "verifiable random number") the member's start their stopwatches and wait for the "Block Time" to expire. It's fine that the stopwatch of each member is slightly out of sync.



Processes Assigned To Priority Forger List

Randomness at *h-1* orders processes into new list at *h*

SLOT	Publish	Points	*h*
0	5+ secs	1 pt	P_{2313}
1	6+ secs	$^1/_2$ pt	P_{3493}
2	7+ secs	$^1/_4$ pt	P_{939}
3	8+ secs	$^1/_8$ pt	P_{93843}

d = 6s t = +1s q = 0.5

Blocks published by process “2313” recognized after 5 seconds.

Blocks published by process “3493” recognized after 6 seconds.

The nominal “Block Time” here is 5 seconds.

Staggering the times at which published forger blocks are recognized is only a network performance optimization and the protocol does not depend upon it for correctness of safety.

Random but shared ordering of all processes in network…

Processes Assigned To Priority Forger List

A new list is created at each block height

SLOT	Publish	Points	h	h+1	h+2	h+3	h+4	h+5
0	5+ secs	1 pt	P_{2313}	P_{4792}	P_{1101}	P_{3883}	P_{877}	P_{7615}
1	6+ secs	$^1/_2$ pt	P_{3493}	P_{103}	P_{2993}	P_{4652}	P_{2840}	P_{5620}
2	7+ secs	$^1/_4$ pt	P_{939}	P_{9291}	P_{9742}	P_{8746}	P_{7221}	P_{66}
3	8+ secs	$^1/_8$ pt	P_{93843}	P_{382}	P_{4207}	P_{7723}	P_{9837}	P_{2811}

d = 6s t = +1s q = 0.5

Chain Choice Rules

Correct processes try to build on the highest scoring chain

SLOT	Publish	Points	h	h+1	h+2	h+3	h+4	h+5
0	5+ secs	1 pt	P_{2313}	P_{4792}	P_{1101}	P_{3883}	P_{877}	P_{7615}
1	6+ secs	1/2 pt	P_{3493}	P_{103}	P_{2993}	P_{4652}	P_{2840}	P_{5620}
2	7+ secs	1/4 pt	P_{939}	P_{9291}	P_{9742}	P_{8746}	P_{7221}	P_{66}
3	8+ secs	1/8 pt	P_{93843}	P_{382}	P_{4207}	P_{7723}	P_{9837}	P_{2811}

→ **BEST PARENT**
+ 4 points

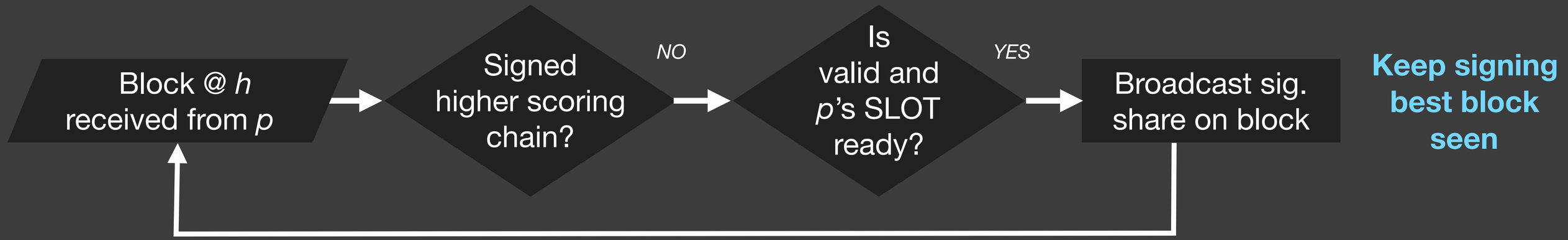
+ 2 3/4 points

d = 6s t = +1s q = 0.5

Threshold Block Notarization

Group members sign until ≥1 block gets threshold signature

The signing behavior of a group member at block height *h*



Threshold Block Notarization

Of course, generally only 1 block gets signed....

Before the Block Time expires, received blocks placed in priority queue	Group members only sign blocks if not seen higher priority block...	When the chain score drops, the protocol lengthens the Block Time
Group members place received blocks into a priority queue while waiting for the Block Time to expire. If a block from the process in SLOT 0 is received, it will be placed at the head of the queue.	After expiry of the Block Time, a member will usually have a block from SLOT 0 waiting in the priority queue. This is signed first, so no others will be signed*. Generally, we expect groups to sign only 1 block.	Imagine if SLOT 0 was produced in Europe and SLOT 1 in America. Both might get signed if Block Time too low. If score best chain drops, the protocol lengthens the Block Time to accommodate network speed.

* Notwithstanding equivocation. Always sign equal highest priority prevent deadlock

Notarization Drives Rapid Convergence (Consistency)

Only blocks with group signatures are valid...

SLOT	Publish	Points	h	h+1	h+2	h+3	h+4	h+5
0	5+ secs	1 pt	P_{2313} σ	P_{4792} σ	P_{1101} σ	FAULTY P_{3883}	P_{877} σ	P_{7615} σ
1	6+ secs	$^1/_2$ pt	P_{3493}	P_{103} σ	P_{2993}	P_{4652} σ	P_{2840}	P_{5620}
2	7+ secs	$^1/_4$ pt	P_{939}	P_{9291}	P_{9742}	P_{8746}	P_{7221}	P_{66}
3	8+ secs	$^1/_8$ pt	P_{93843}	P_{382}	P_{4207}	P_{7723}	P_{9837}	P_{2811}

d = 6s t = +1s q = 0.5

Timestamping Benefits

Threshold power!

SUPER FAST - during normal operation expect overwhelming probability of transaction irreversibility (“finality”) in:

7.5s

Threshold groups notarizing blocks resolves several security challenges...

- Nothing At Stake
- Equivocation
- Selfish Mining

Quantifiable risk

Presence multiple notarizations, chain score, allow est. statistical risk

SPV

Prove to a light client that only has Merkle root of groups

Waiting for Casper... PoW On Ethereum

- **Currently 50+% of blocks mined are empty...**
- **Proof-of-work's "Poission distribution" is cause**
- **Sooner you publish, greater chance of being "confirmed"**
- Publishing an empty block without delaying to validate transactions is more profitable...
- Building on an empty block that does not involve validation delay is more profitable...
- **To combat problem, per block gas limit set to tiny levels**

Bitcoin Could Consume as Much Electricity as Denmark by 2020, Motherboard 3/29/2016

Ridiculous!!!

Relative Performance

Threshold power!

			
Block time	Average 10 mins *varies wildly*	Average 20 secs *varies wildly*	Average 5 secs *low variance*
“TX finality” (speed)	6 confirmations avg. 1 hr	25 confirmations avg. 6 mins	2 confirmations ***avg. 7.5 secs***
Gas available	- - -	Low due to Poisson distribution	**25-50X Ethereum** ***(Unlimited scale-out achieved by applying randomness in following techniques…)***

3

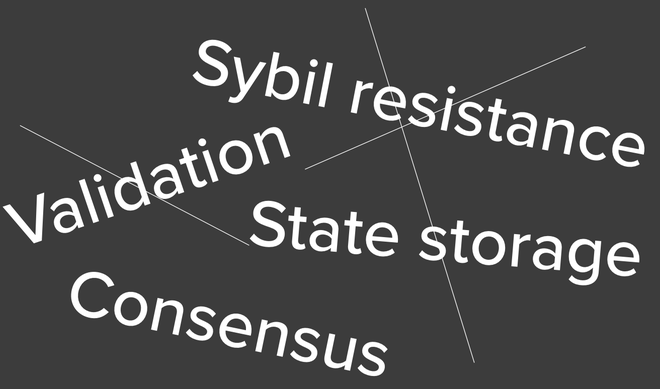
Applications of Randomness

Scale-out virtual computer

Handling any volume of computation and state

Separate and decouple concerns

Proof-of-Work Blockchain



DFINITY

Consensus

Validation

State storage

Sybil resistance

Good old-fashioned Computer Science!

TCP/IP

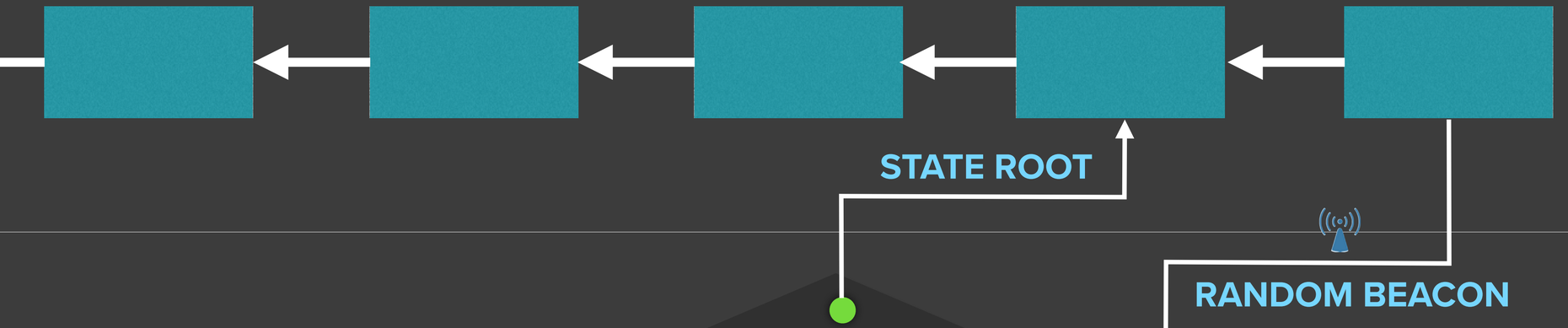
Application

Transport

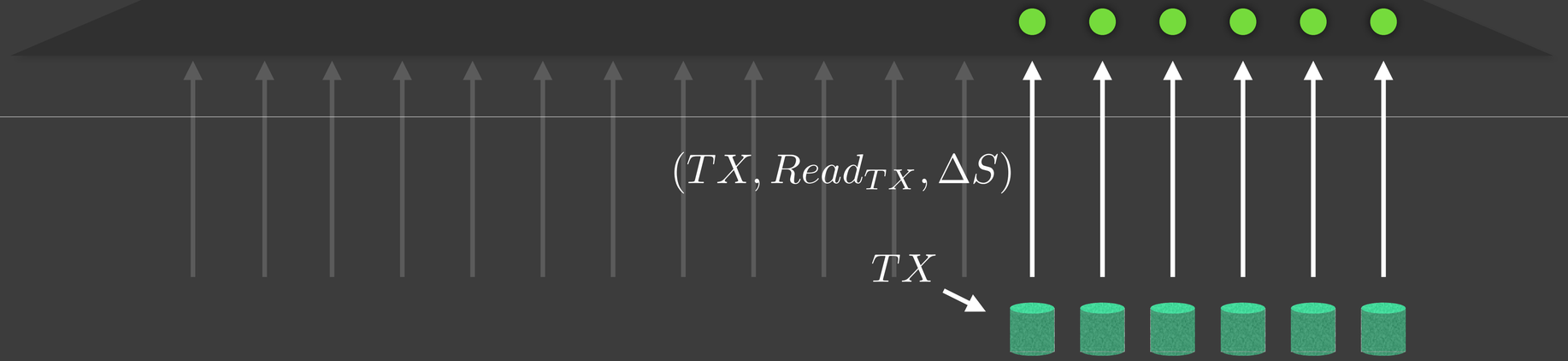
Internet

Network Access

Three Layer “Scale-out” Architecture



RANDOM BEACON
DRIVES TREE



CONSENSUS

Threshold relay chain generates randomness, records network metadata & validation tree “state root”.

VALIDATION

Scalable “Validation Tree” composed “Validation Towers”. Does for validation what Merkle does for data.

STORAGE

State and updates to state stored on shards. State transitions passed to Validation Tree.

Example Applications Of Randomness

1

SCALING

VALIDATION TOWER

Validate network state changes using few processes

Each additional level of the tower validates new state transitions applied to some storage shard, and is built by processes selected by the random beacon. These processes must also validate levels beneath them to some validation depth d. Once a level has been added, a process becomes economically inactive until it has been buried by d further levels. It cannot predict who will build these levels, and thus it become computationally infeasible to collude with shards and have bad transitions validated.

2

USER EXPERIENCE

LAZY VALIDATION

Create instant Web search results

Some queries of data on the virtual computer are relatively inexpensive and do not warrant the cost of a validation in a Validation Tower. For example, a search of a Web index is a low value operation. Nonetheless, validation is still necessary, since otherwise miners might insert advertising into search results. To address this situation, query results are validated 1. only occasionally upon direction of the random beacon and 2. after the query has been returned to the user.

3

COST REDUCTION

LOTTERY CHARGING

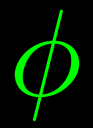
Create inexpensive Web search results

While we can use lazy validation to reduce the cost of low value operations such as search queries, any operation that involves currency (in this case “dfinities”) must necessarily be fully validated - after all, lots of small frauds make a big fraud!!! A problem thus exists, because the computer cannot make any operation free or it will be DOSed. The solution is to multiply charges by e.g. 1000, and then only apply them 1/1000 times as directed by the random beacon.

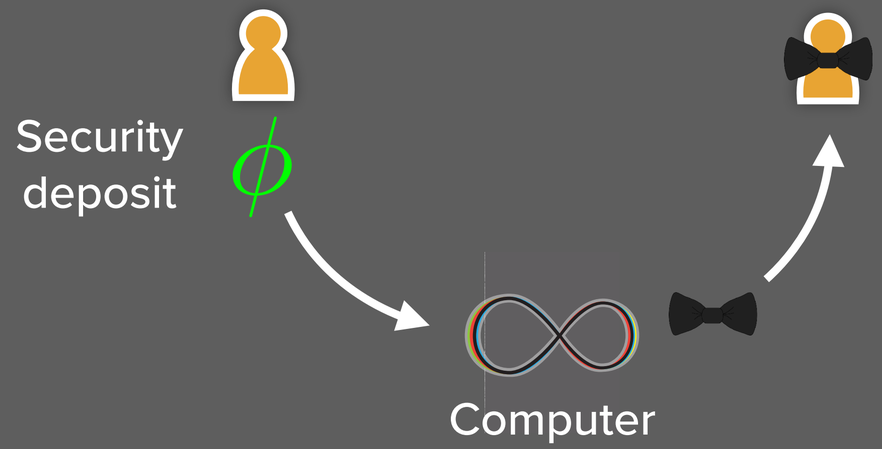
4. Extract from DEVCON2 PHI presentation demonstrating use of randomness to originate loans algorithmically

4. Extract from DEVCON2 PHI presentation demonstrating use of randomness to originate loans algorithmically

Decentralized commercial banking and stable currency _



Anyone Can Be A **PHI Validator**

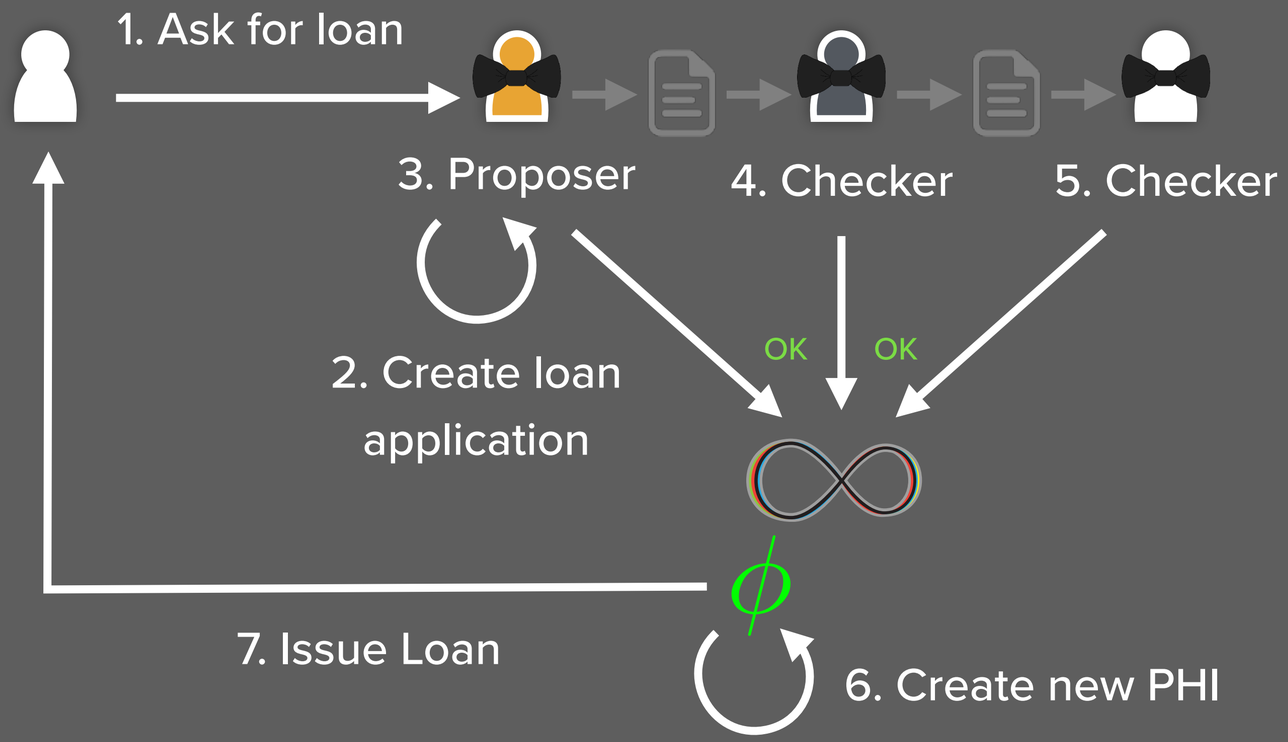


Example (paid in PHI)

Deposit	$50,000
Max Loan	$5,000
Max Loans	$500,000

Anyone can become a PHI Validator by making a security deposit to the computer. If a loan you approve becomes delinquent the computer takes compensation from your deposit.

How Computer **Issues Loans**



- Random sequence validators
  - Nobody knows who's next
  - Nobody knows length
- Choice validators
  - Size of their deposit
  - Reputation
- Loan application
  - Format is open standard

Validator **Incentives**

Proposer	Checker	Checker
60% loan interest	20% loan interest	20% loan interest
60% underwriter	20% underwriter	20% underwriter
reputation	reputation	reputation

EXAMPLE

Validator **Reputation**

Loans performing

Deposit size maintained

Incompetence & game playing are losing strategies

The lower your reputation falls the longer the validation chains and the harder it is to make returns from security deposit, which can get “frozen”

Loans have delinquent payments

Deposit size decreasing

Other validators reject loans

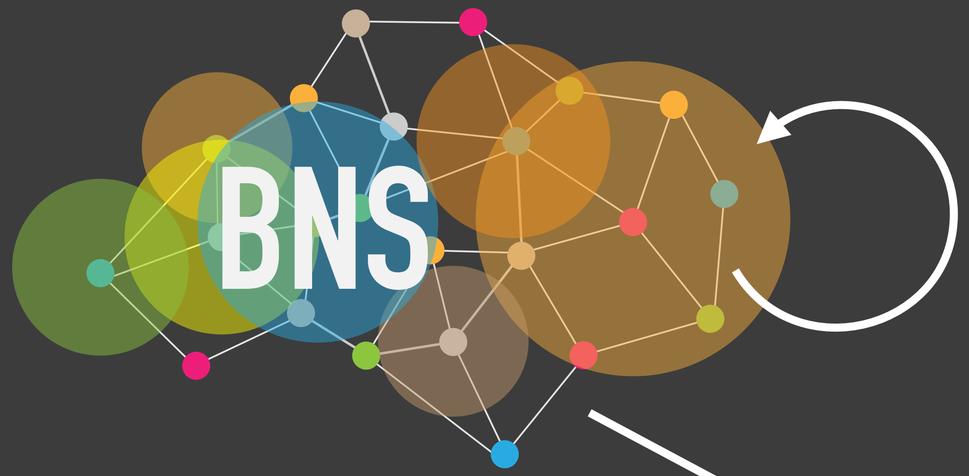
If validation decision rejected by others, can appeal and computer decides using new random sequence

4

Blockchain Nervous System

Decentralized governance

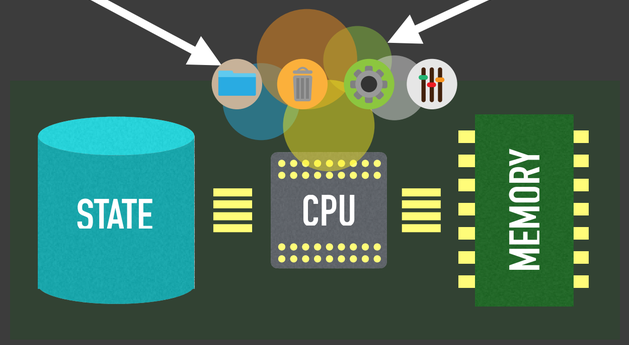
Proposal Processing



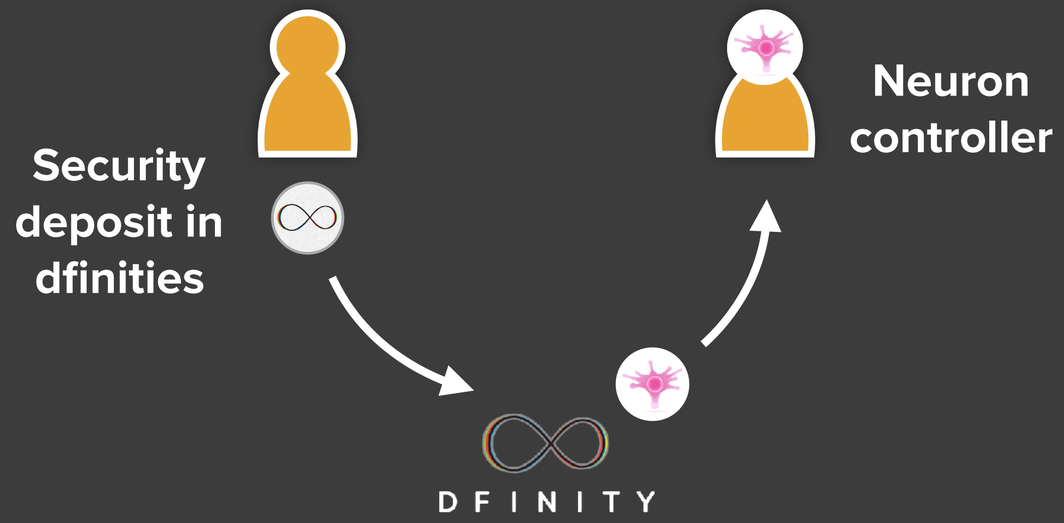
2. BNS evaluates proposal
- Process is type dependent
- Output is adopt | reject

1. Submit proposal
- Standard types of proposal
- A fee is paid in dfinities
- Deposit refunded on adoption

3. Voting triggers decision
- Use privileged EVM op codes
- Freeze contracts, move tokens
- Run arbitrary privileged code
- Configure platform



People Create Neurons

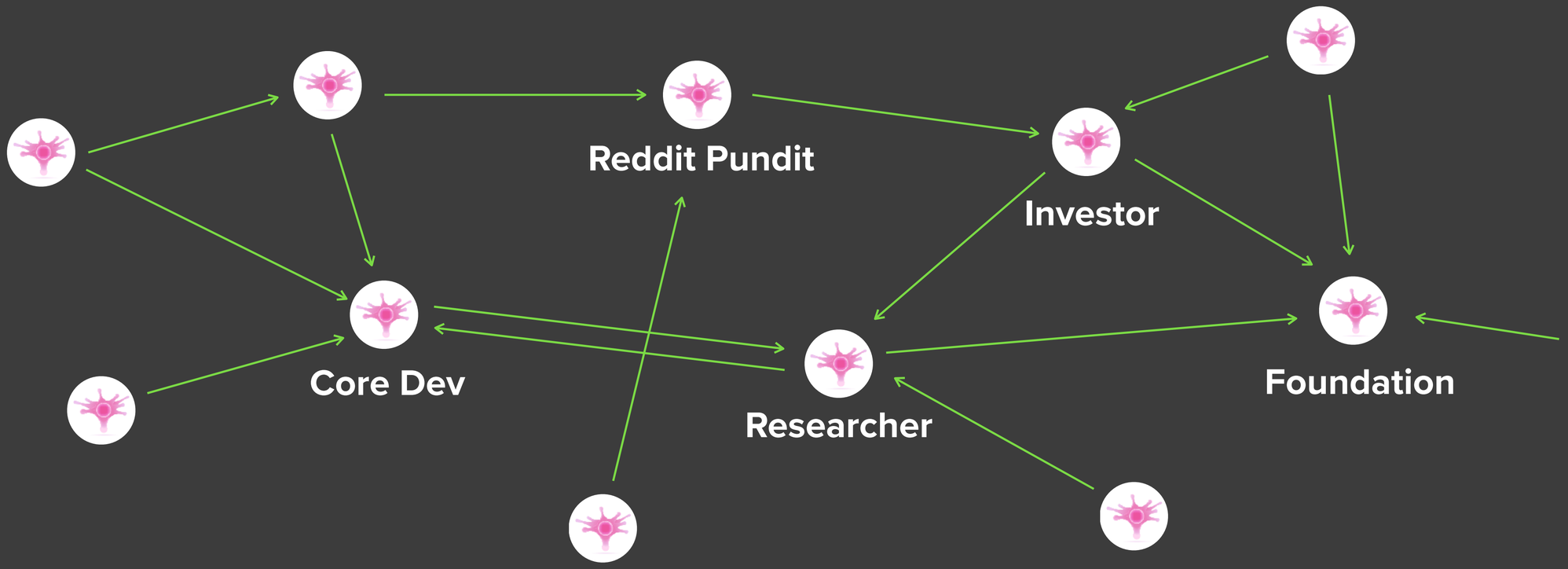


Neuron key pair configured into laptop or smartphone client

Neuron's voting power equals dfinities deposited

Neurons are created by making a security deposit to a special DFINITY smart contract.
It takes 3 months to dissolve a neuron and retrieve the deposit, incentivizing good decision making

Neurons Follow Neurons...



Neurons follow other neurons. This enables them to make decisions without benefiting from the input of their direct human controller. People can advertise the address of their neuron

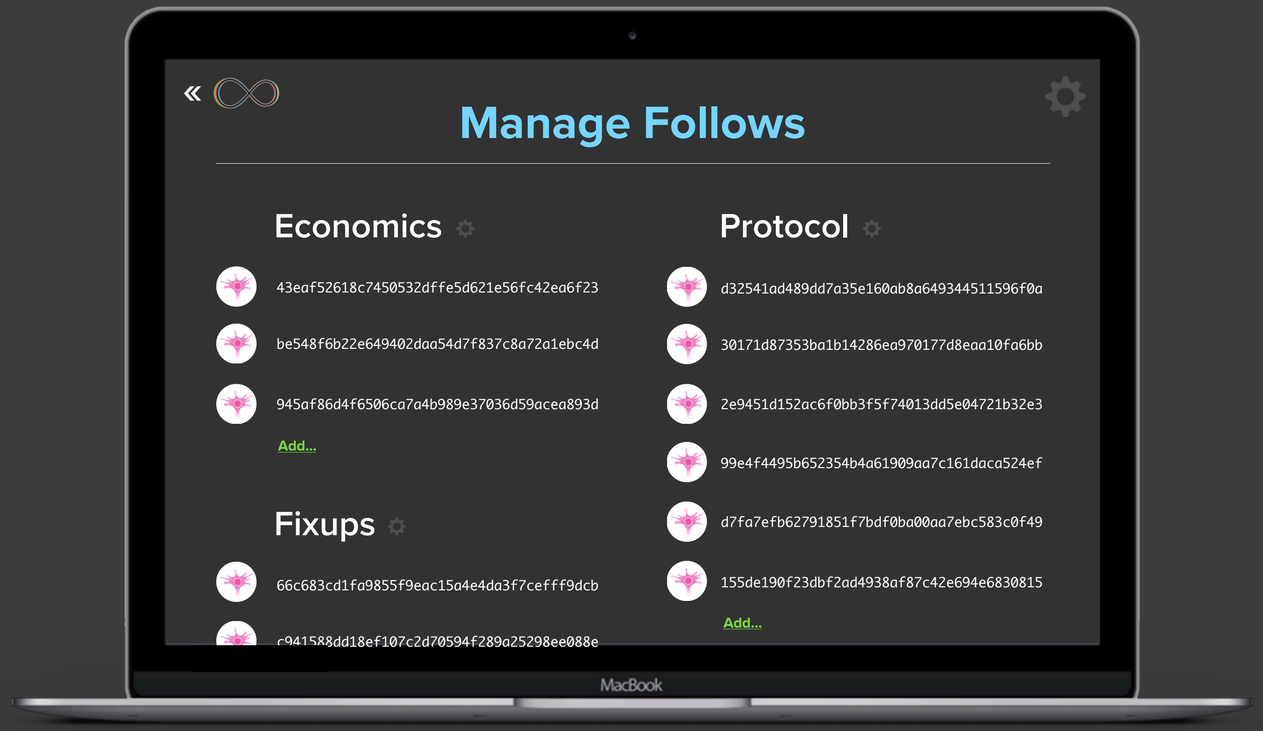
DFINITY

Neuron Client Software

Proposals are submitted with a non-refundable fee and deposit that is refunded if the proposal is adopted. Before a timeout, the neuron controller (owner) can manually vote on proposals…

DFINITY

Neuron Client Software



When the neuron's controller (owner) doesn't vote on a proposal before the timeout, the neuron client software examines the follow list configured for the proposal's type...

DFINITY

BNS Properties

Decentralized intelligence

Captures wisdom of crowds

- Community expertise incorporated
- Opaque liquid democracy

Neurons cascade to make decisions on proposals

- Non-deterministic (depends timing)

Highly resilient

- Trust graph (follow relationships) exist on edges and are unknowable...
- Difficult to kidnap, extort or influence “key holders”
- Government, or “frozen” cos cannot capture, sue etc...
- Good incentives system

The BNS learns

- Follow relationships are dynamic
- Improves over time

Thought mining

- Neurons earn money
- Factor down by proportion votes missed

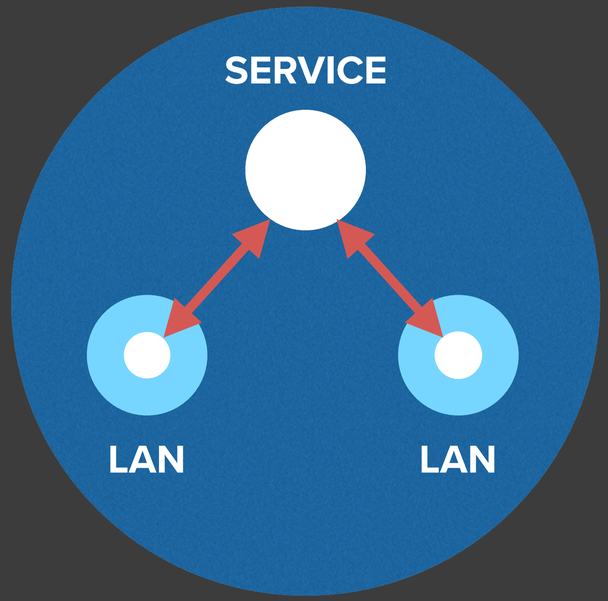
5

Public - Private Interoperability

Possible using high consistency of crypto:3 networks

Internet Network

PRIVATE LANS CONNECTED *

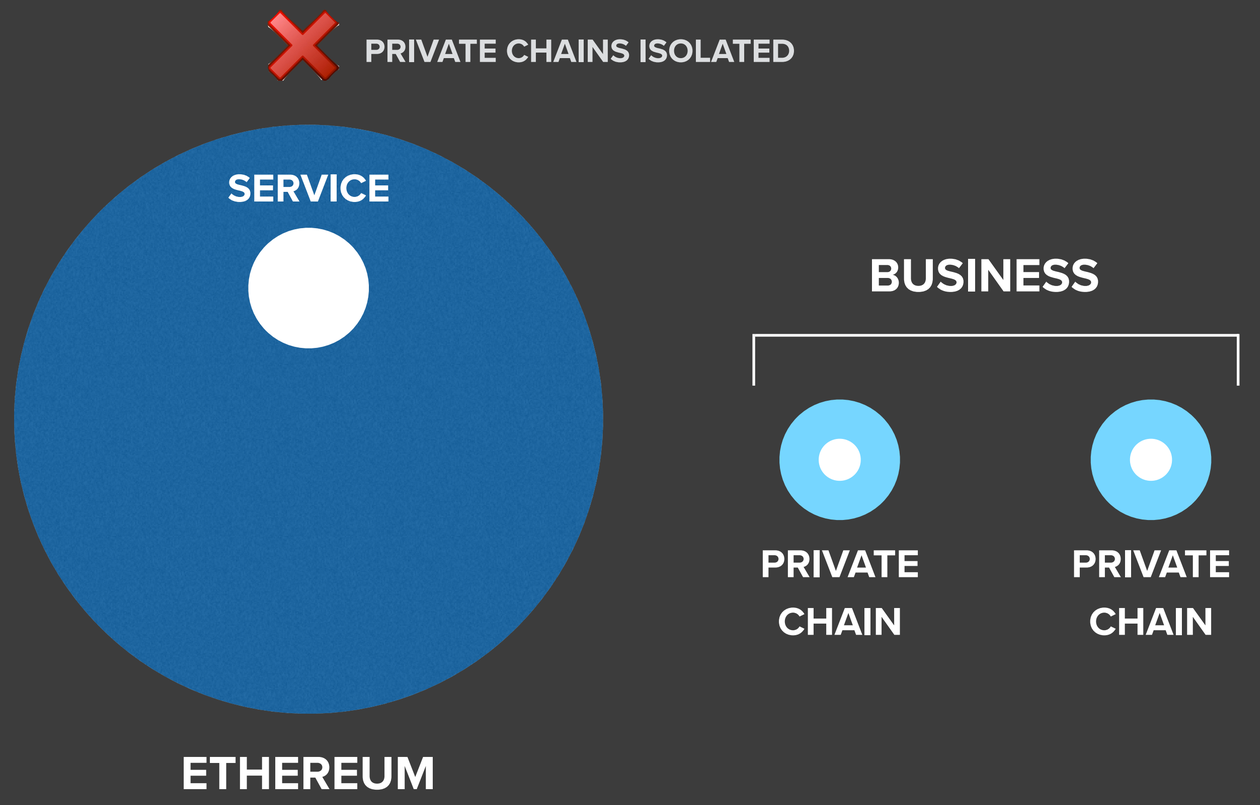


INTERNET

* LAN = Local Area Network, run in office or home

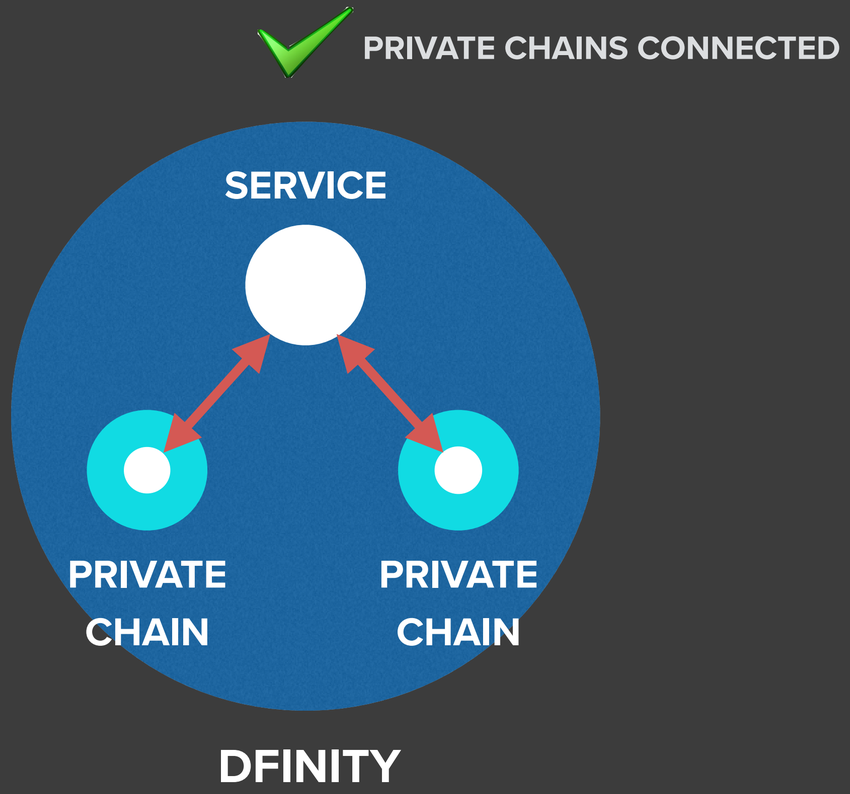
DFINITY

Ethereum Network



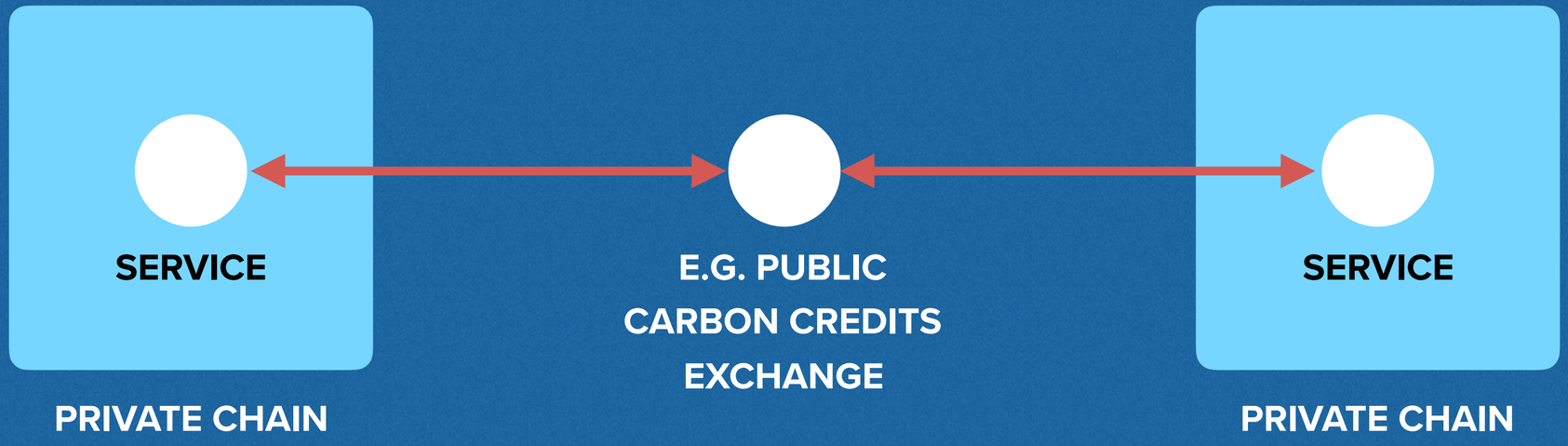
DFINITY

DFINITY Network



Private Chains Interact

EXAMPLE



DFINITY

Public Systems Are Building Blocks

EXAMPLE

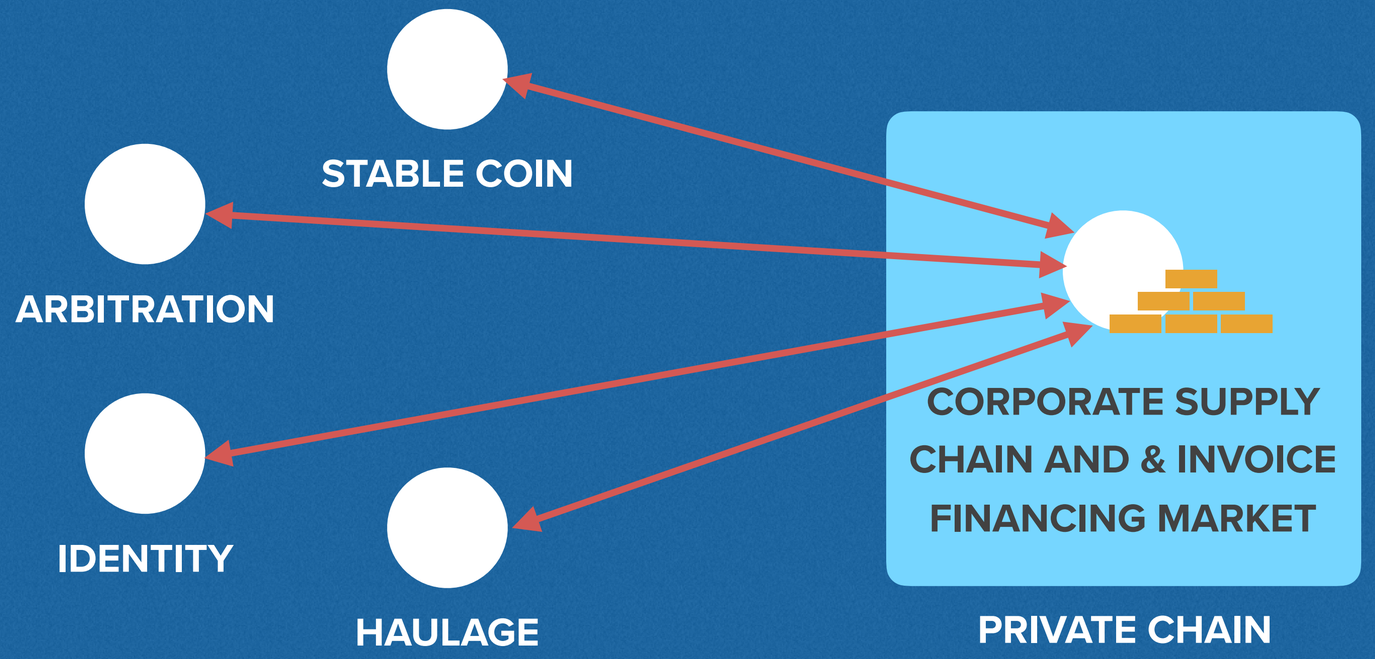
```
Include “dfinity:StableCoin.sol”
Include “dfinity:Arbitration.sol”
Include “dfinity:Identity.sol”
Include “dfinity:Haulage.sol”


_
```

PRIVATE SMART CONTRACT CODE

DFINITY

Public Systems Are Building Blocks

EXAMPLE



DFINITY

6

Release Schedule...

Release Schedule

Objective: race to Tungsten < 1 year

1| COPPER

- Threshold Relay Chain
- Blockchain Nervous System (BNS)
- Security deposits
- State-root-only-chain

2| ZINC

- Private - Public IOP

3| TUNGSTEN

- State sharding (basic)
- Validation Towers (basic)
- Asynchronous model for cross-shard programming
- USCIDs (Unique State Copy IDs)
- Advancements in BNS

4| LITHIUM

- Full “Validation Tree” architecture
- Micro-sharding
- Advanced economic models
- zkSNARKs, and other privacy enhancements

More TBA...