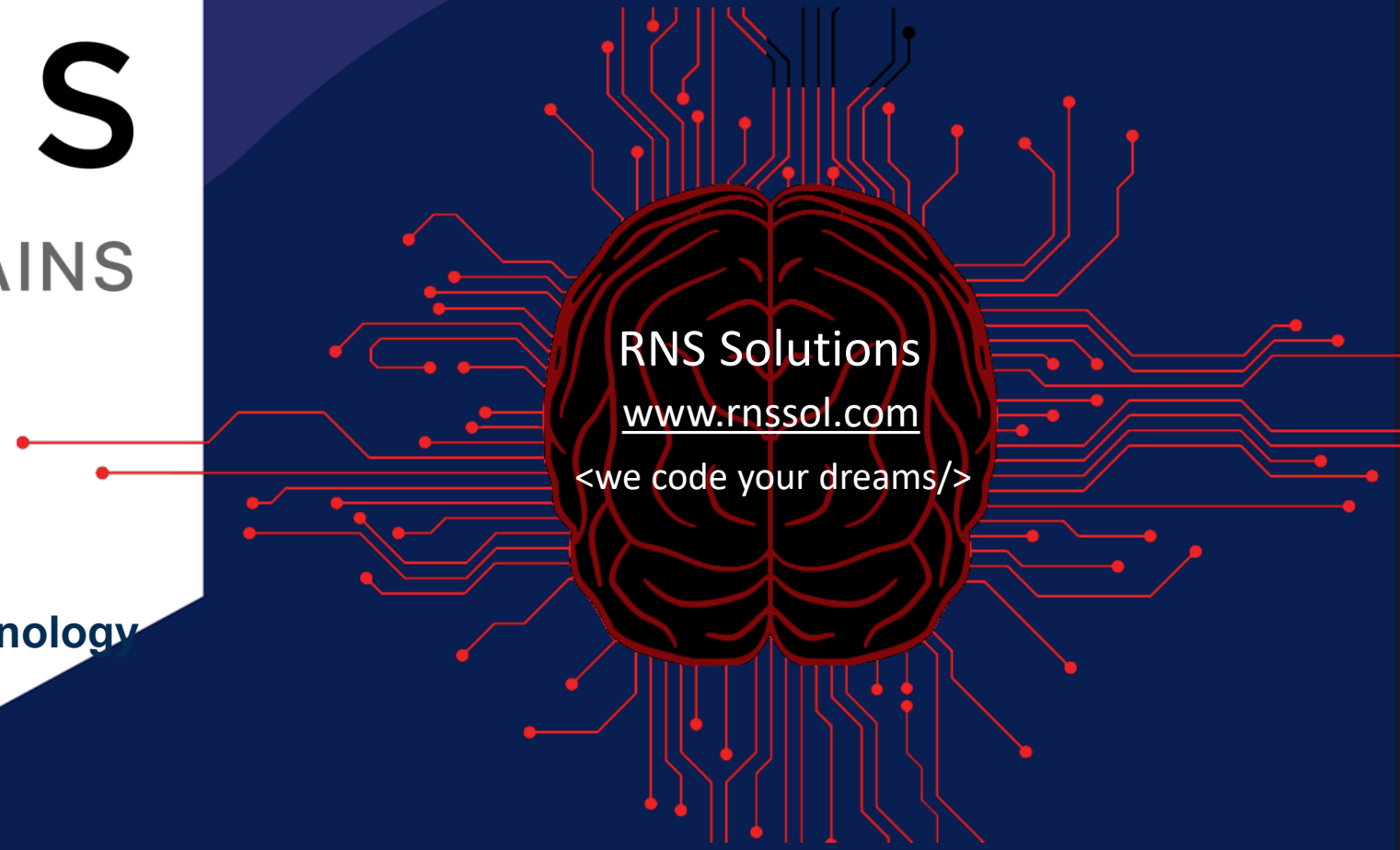
COSMOS

INTERNET OF BLOCKCHAINS

Shakil Muhammad

ITU Workshop on Distributed Ledger Technology
Scalability and Interoperability



Generation 3

Generation 1

- ✓ Sovereignty
- ✓ Efficient state machines
- ✓ Customizability

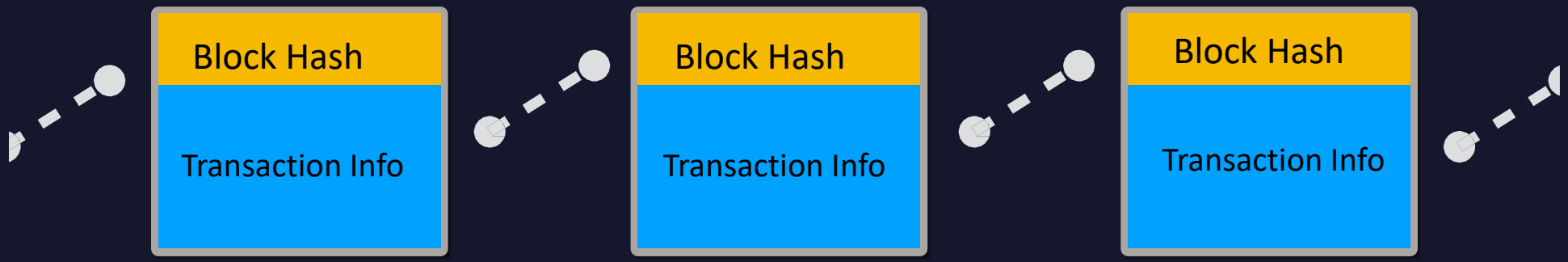
Generation 2

- ✓ Interoperability of Dapps
- ✓ Easier to develop
- ✓ “1 click” deploy

- ✓ Scalability
- ✓ Fault Tolerance
- ✓ Sustainable
- ✓ Interoperability
- ? Privacy

Scalability Problem

Transaction verification and consensus building take longer as more participants joins in the network



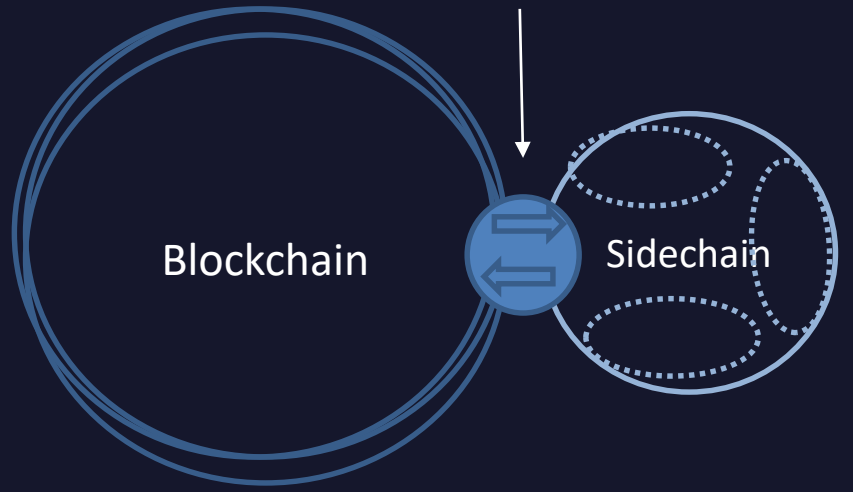
Scalability issues need to be solved to put blockchain into practical use

Solutions

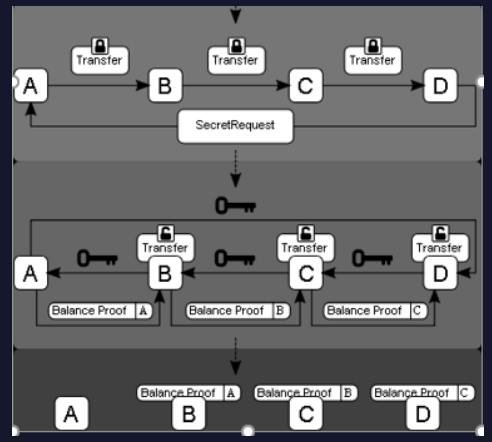
- Layer2
  - SideChains
  - Plasma, Cosmos
  - State Channels
  - Raiden, Lightning
- Layer1
  - Sharding
  - Ethereum Sharding, Ziliqa
  - Consensus Solution by
  - Casper, Tendermint, …

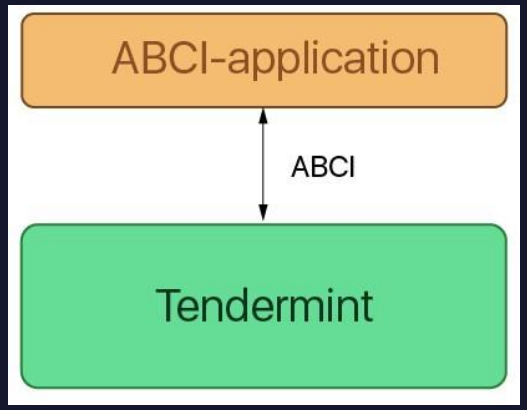
Anyway in the main chain, Let's reduce things to do!

Two way peg



Let's speed up Tx processing





11 tools

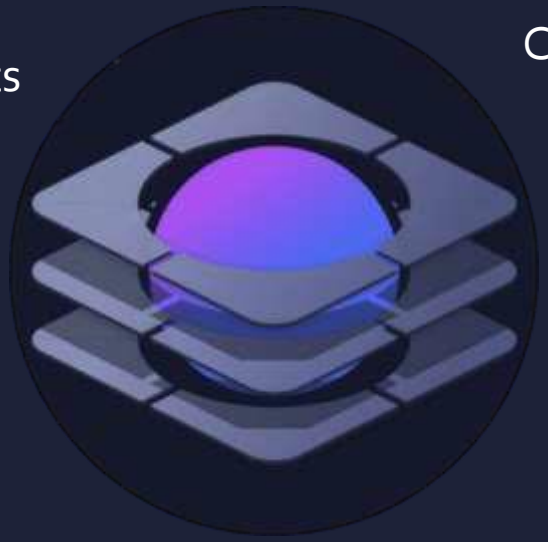
Ethermint

Monetary Experiments

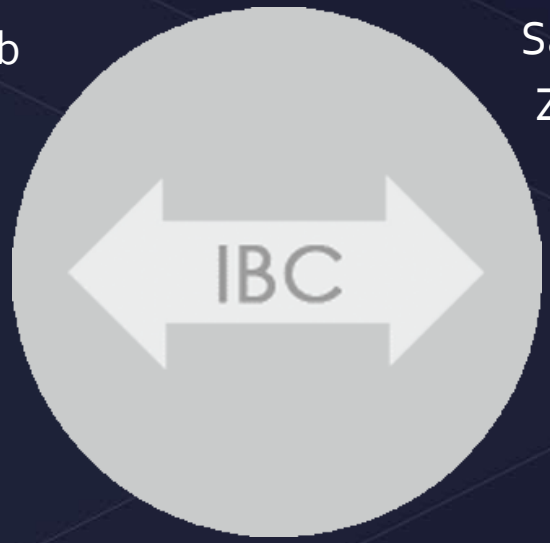
Cosmos Hub

Sample Zones

Tendermint



Cosmos SDK



IBC

Voyager

Proof of Stake

Alternative Frameworks

Low Level Libs

Tendermint

Tendermint Core

CGI

APPLICATION PLATFORMS

SOCKET PROTOCOLS

SECURITY & NETWORKING PLATFORMS

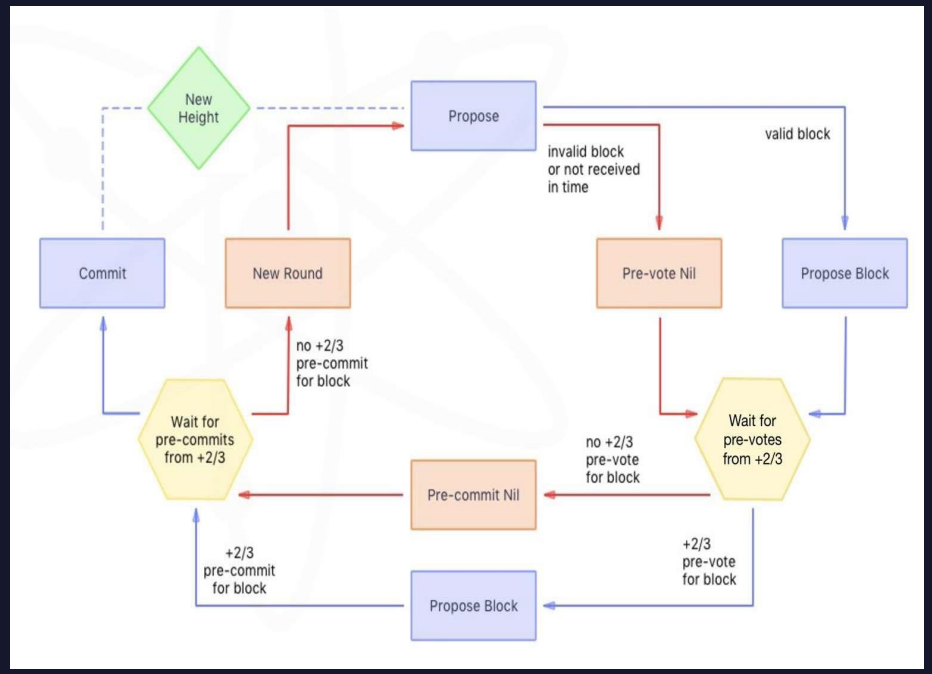
Your State Machine

ABCI

Tendermint

Tendermint BFT

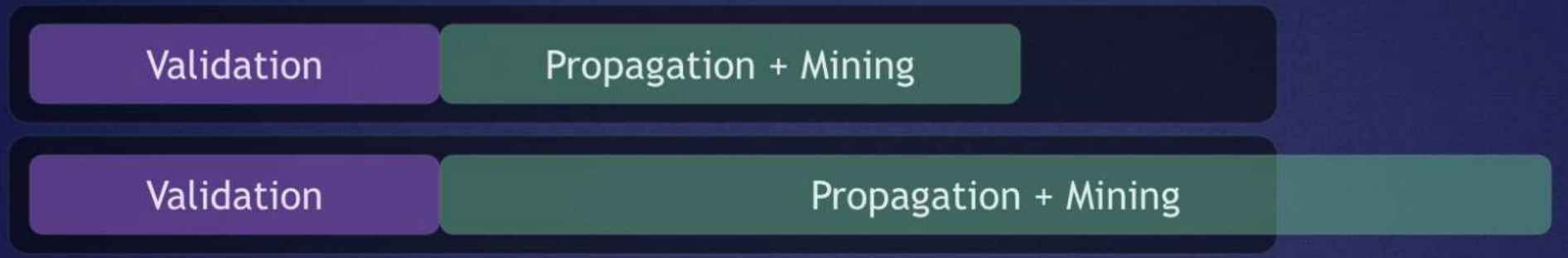
- Simplified and improved PBFT
- Provable liveness in partially synchronous network
- **Safety threshold:** ⅓ of validators' power
- **1-block finality**
- Consistency-prioritizing
- Rotating proposer
- Tendermint 2.0 in progress

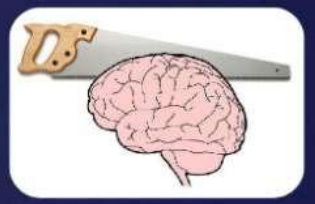


BLS aggregator, pipelining

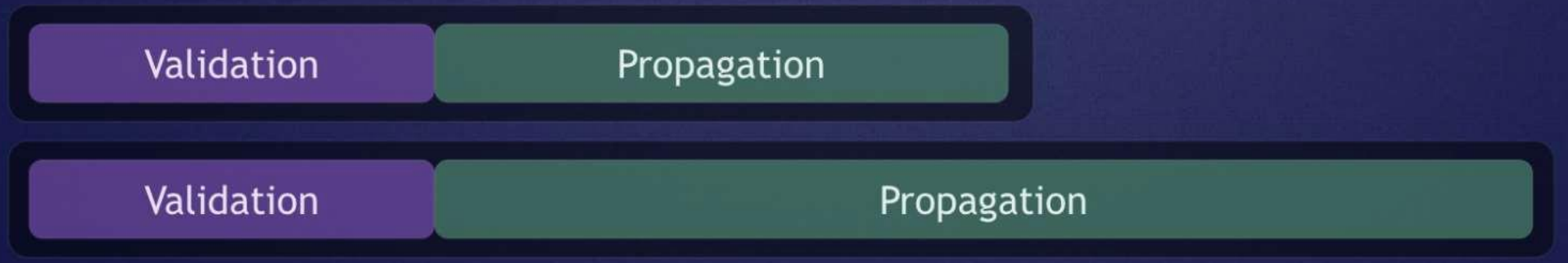
Tendermint BFT

Nakamoto Consensus



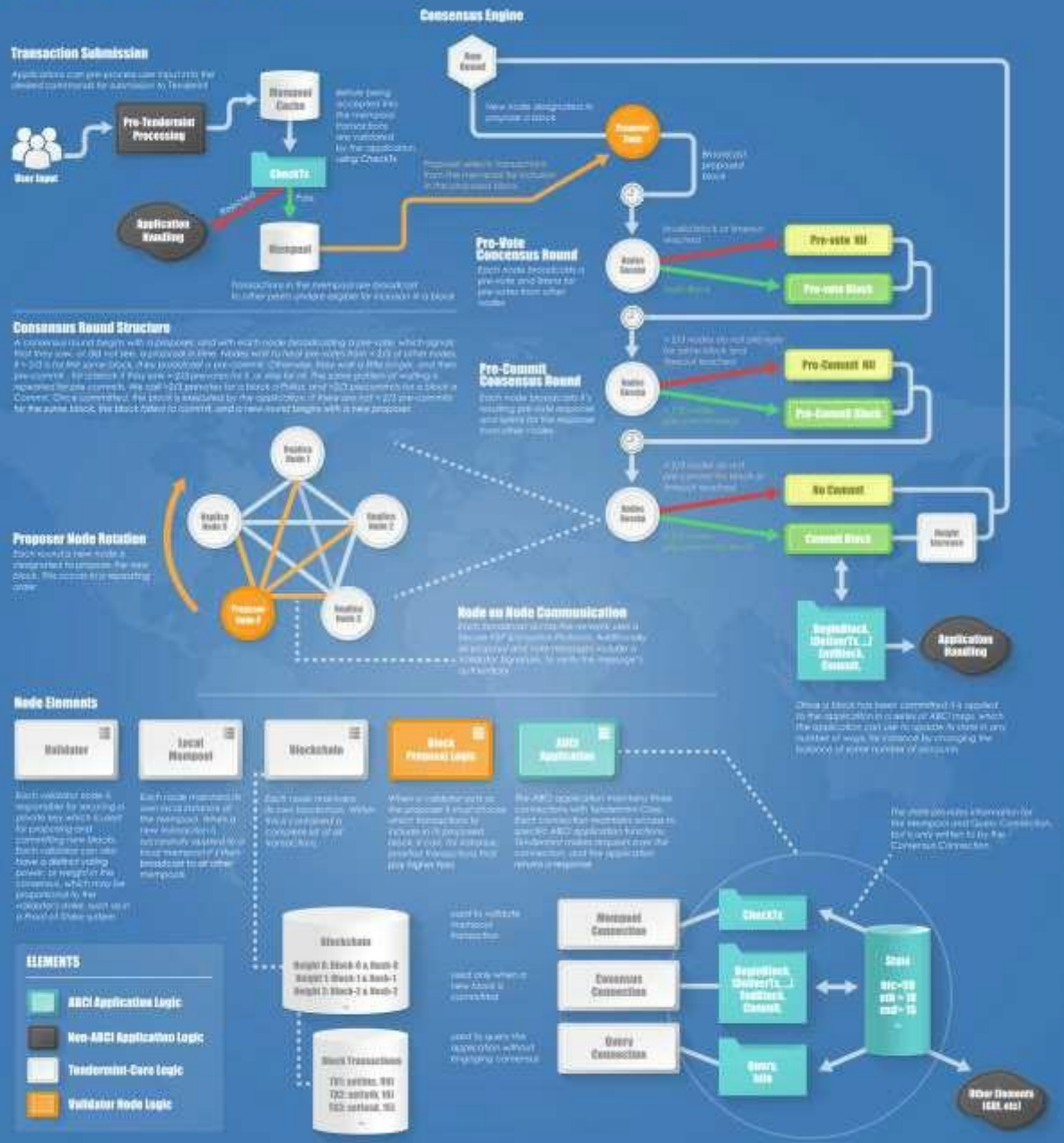


BFT Consensus



10 mins for btc, not lower than 15 sec for ethereum but tenermint can stretch as much as possible

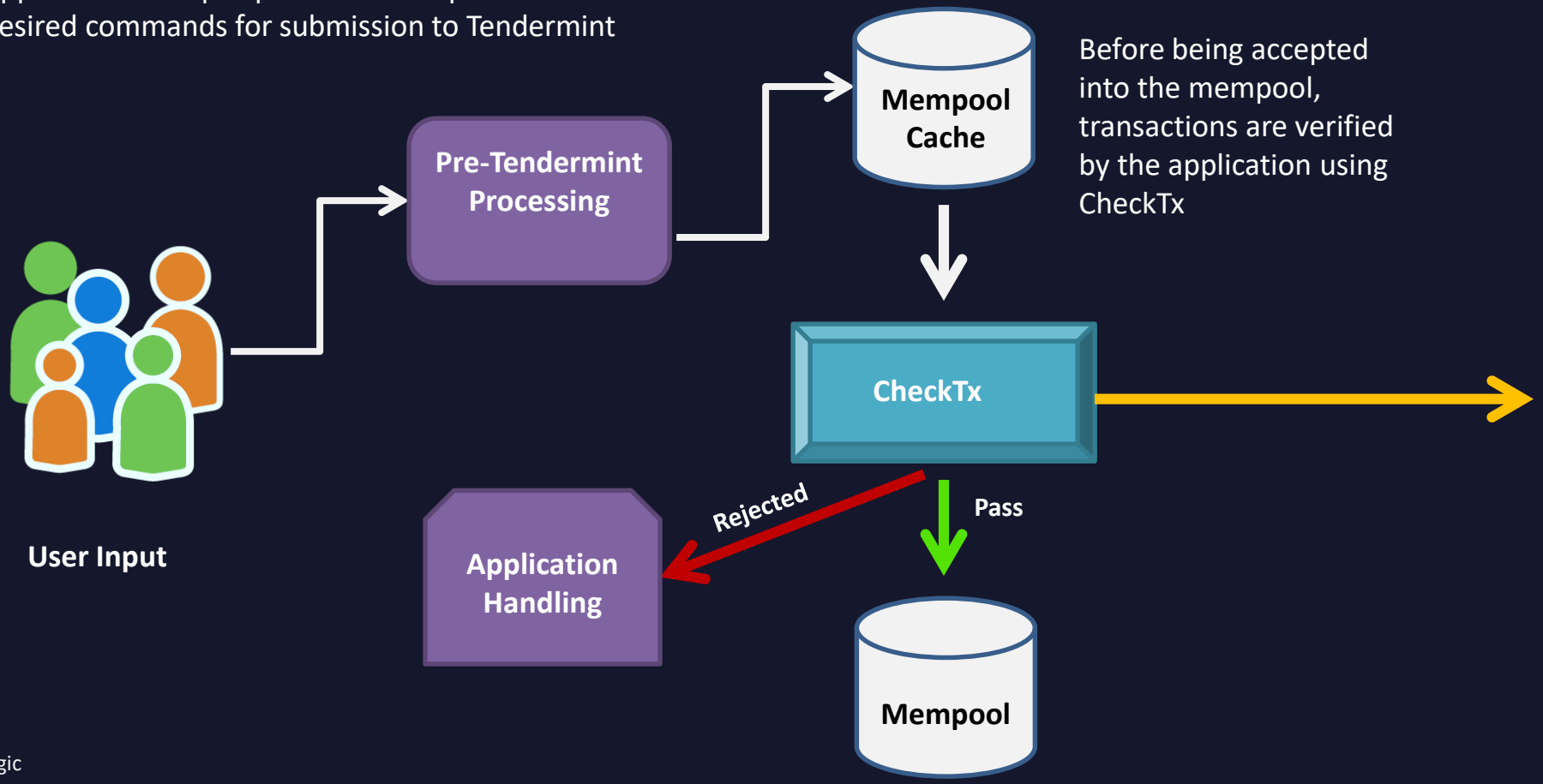
Tendermint in a Nutshell



1) Propose

Transaction Submission

Applications can pre-process user input into the desired commands for submission to Tendermint



Transactions in the mempool are broadcasted to other peers and are eligible for inclusion in a block

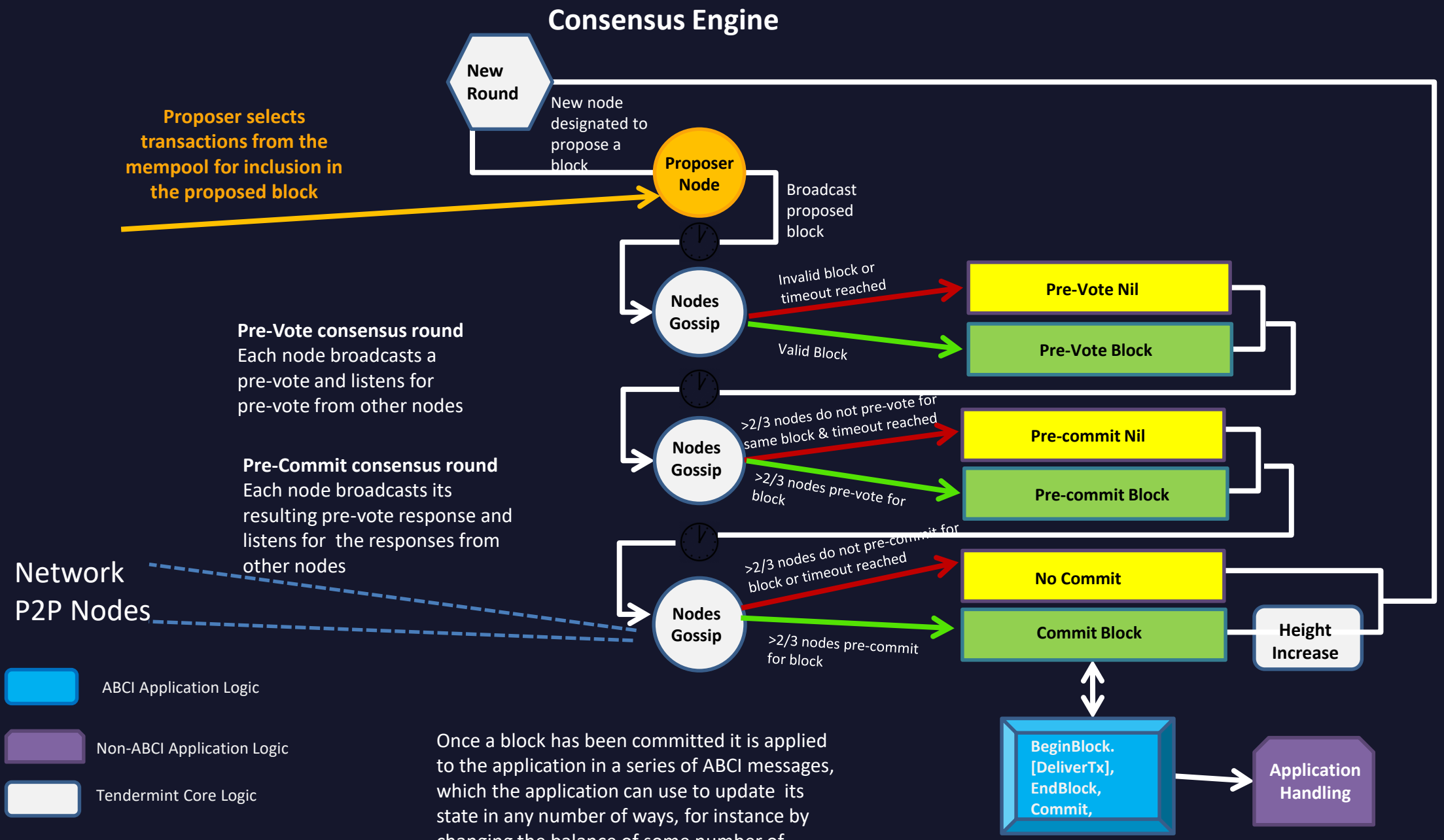
ABCI Application Logic

Non-ABCI Application Logic

Tendermint Core Logic

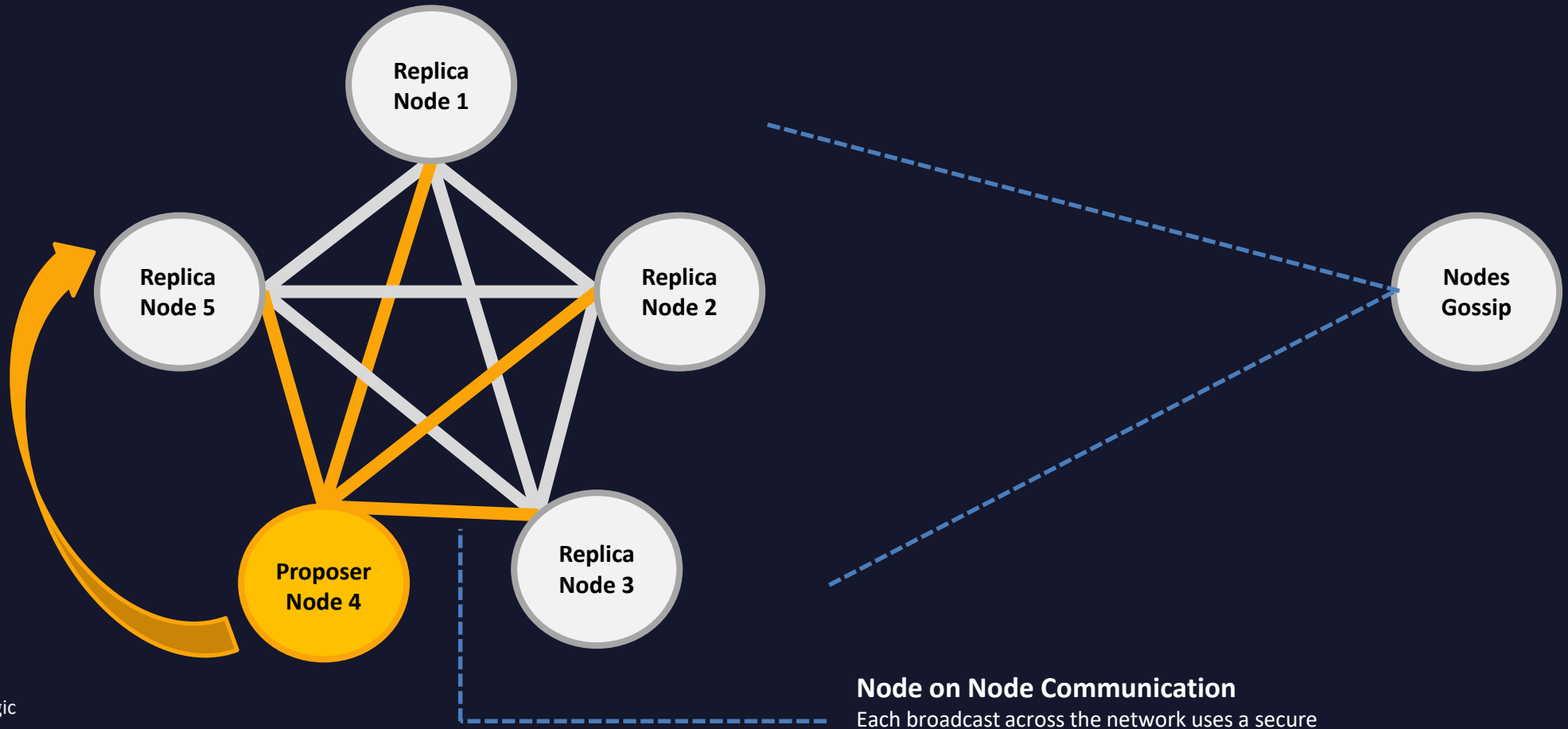
Validator Node Logic

1) Propose


Consensus Engine
New Round
New node designated to propose a block
Proposer Node
Broadcast proposed block
Proposer selects transactions from the mempool for inclusion in the proposed block
Nodes Gossip
Invalid block or timeout reached
Valid Block
Pre-Vote Nil
Pre-Vote Block
Pre-Vote consensus round
Each node broadcasts a pre-vote and listens for pre-vote from other nodes
Nodes Gossip
>2/3 nodes do not pre-vote for same block & timeout reached
>2/3 nodes pre-vote for block
Pre-commit Nil
Pre-commit Block
Pre-Commit consensus round
Each node broadcasts its resulting pre-vote response and listens for the responses from other nodes
Nodes Gossip
>2/3 nodes do not pre-commit for block or timeout reached
>2/3 nodes pre-commit for block
No Commit
Commit Block
Height Increase
Network
P2P Nodes
ABCI Application Logic
Non-ABCI Application Logic
Tendermint Core Logic
Validator Node Logic
BeginBlock. [DeliverTx], EndBlock, Commit,
Application Handling
Once a block has been committed it is applied to the application in a series of ABCI messages, which the application can use to update its state in any number of ways, for instance by changing the balance of some number of accounts

Consensus Round Structure

A consensus round begins with a proposer, and with each node broadcasting a pre-vote, which signal that they saw or did not see, a proposal in time. Nodes wait to hear pre-votes from >2/3 of other nodes. If >2/3 is for the same block, they broadcast a pre-commit. Otherwise, they wait a little longer, and then pre-commit for a block if they saw >2/3 pre-votes for it, or else for nil. The same pattern of waiting is repeated for pre-commits. We call >2/3 pre-votes for a block Polka, and >2/3 pre-commits for a block Commit. Once committed, the block is executed by the application. If there are not >2/3 pre-commits for the same block, the block failed to commit, and a new round begins with a new proposer.



ABCI Application Logic

Non-ABCI Application Logic

Tendermint Core Logic

Validator Node Logic

Proposer Node Rotation

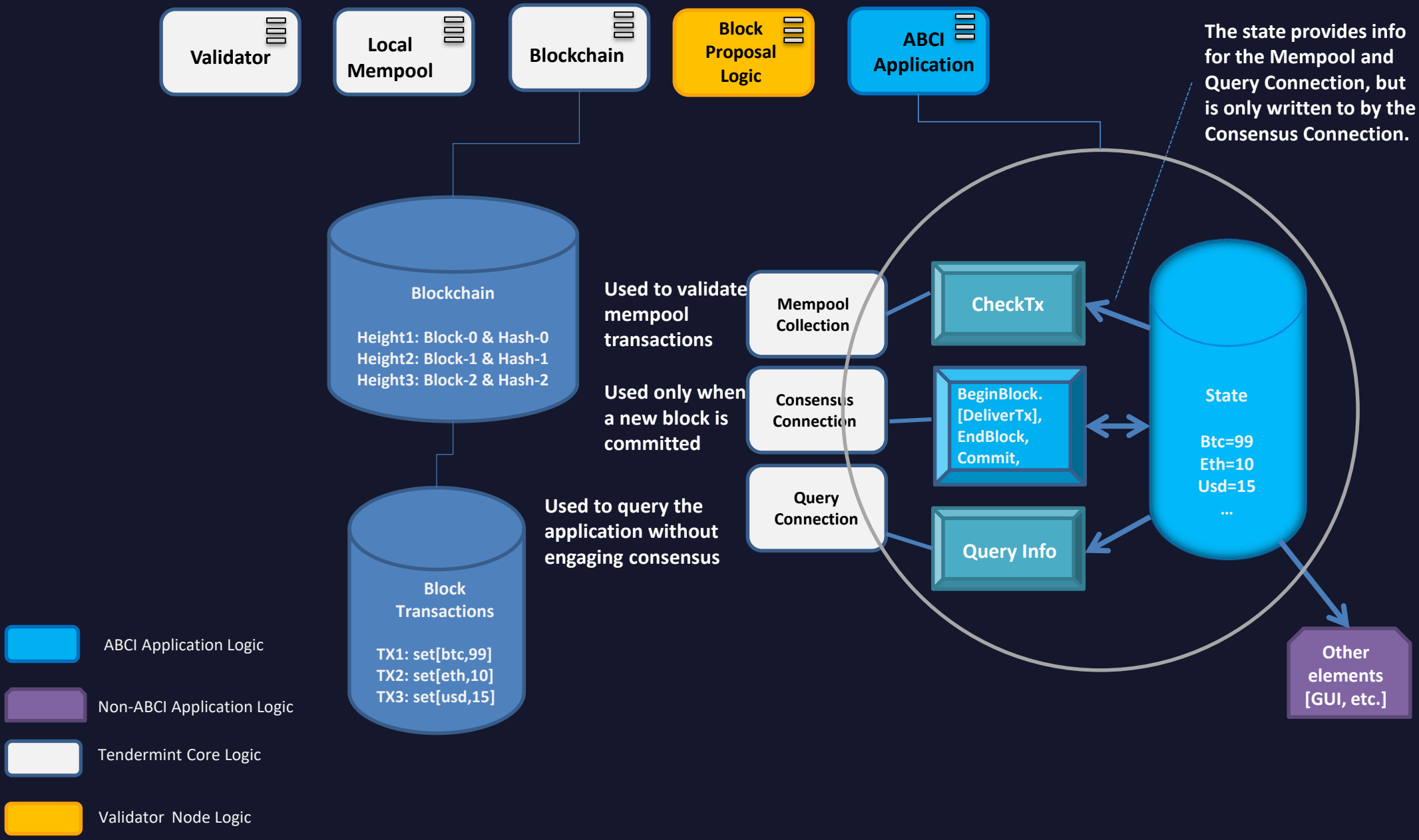
Each round a new node is designated to propose a new block. This occurs in a repeating order.

Node on Node Communication

Each broadcast across the network uses a secure P2P encryption protocol. Additionally, all vote and proposal messages include a Validator Signature to verify the message's authenticity.

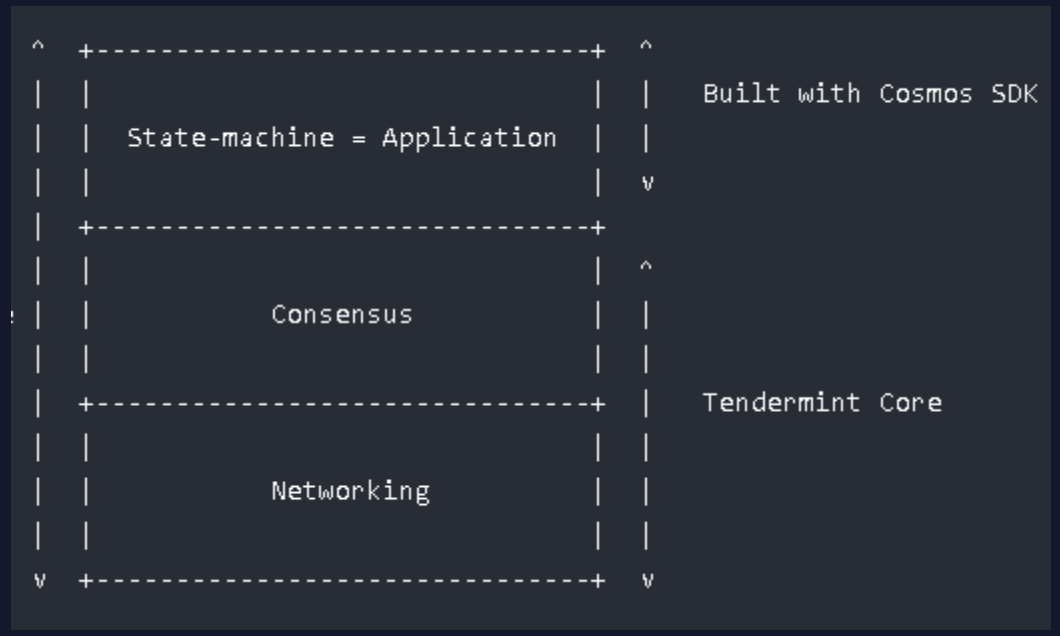
1) Propose

Node Elements

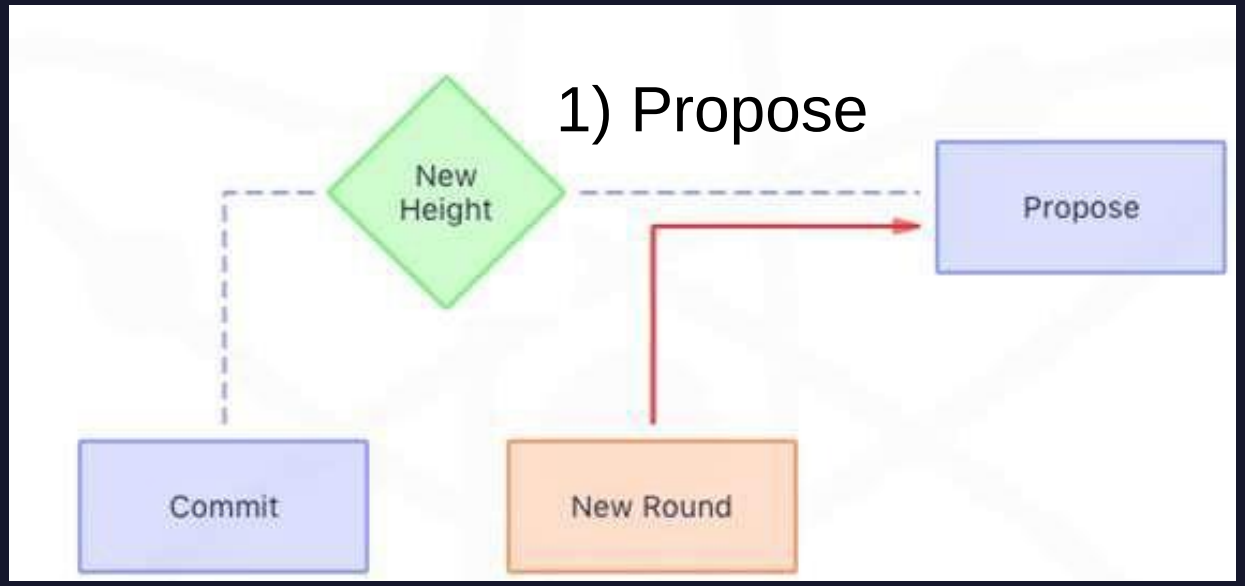


Tendermint State machine

- At each height of the blockchain a round-based protocol is run to determine the next block. Each round is composed of three steps (Propose, Prevote, and Precommit), along with two special steps Commit and NewHeight.
- The sequence (Propose -> Prevote -> Precommit) is called a round. There may be more than one round required to commit a block at a given height.

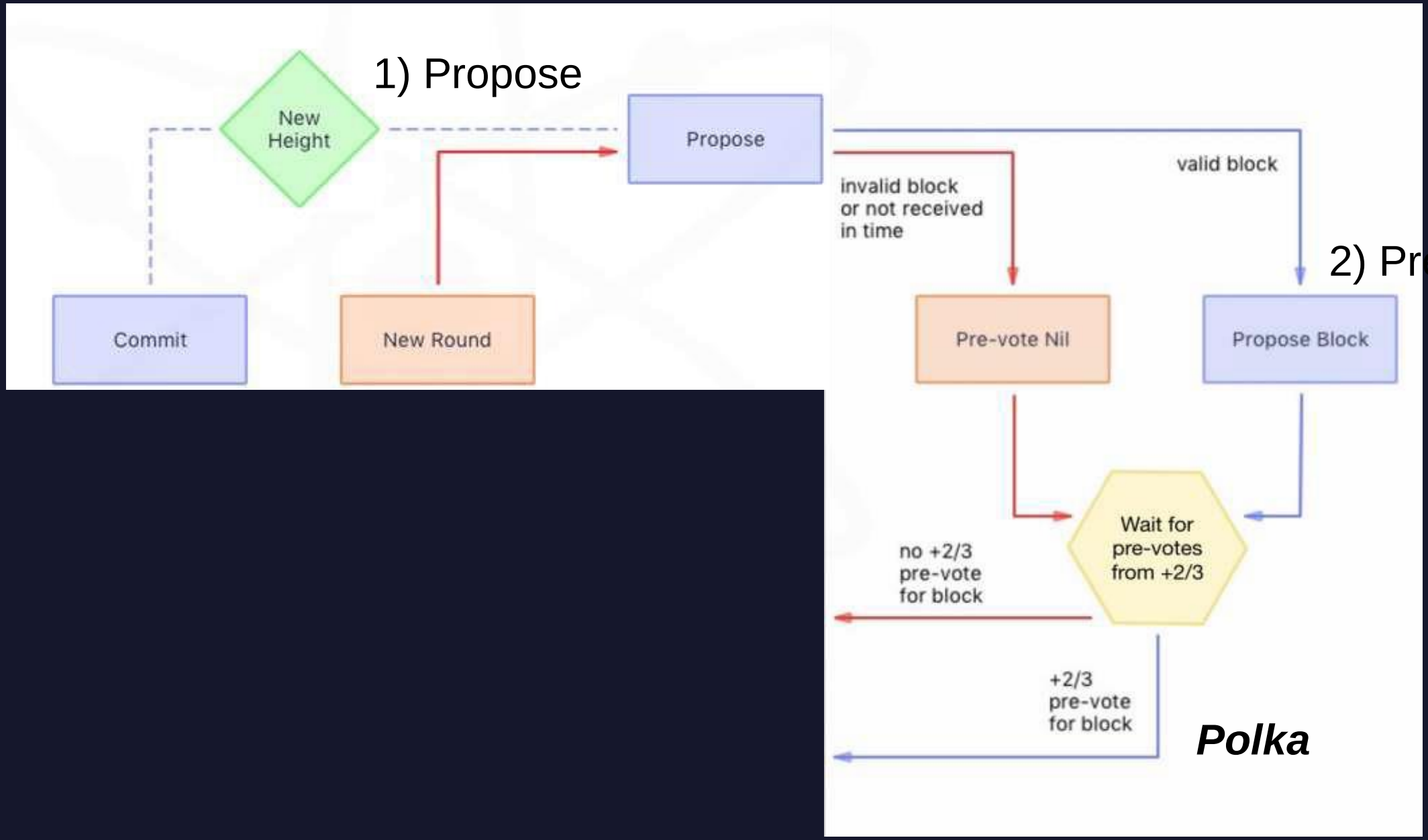


1) Propose

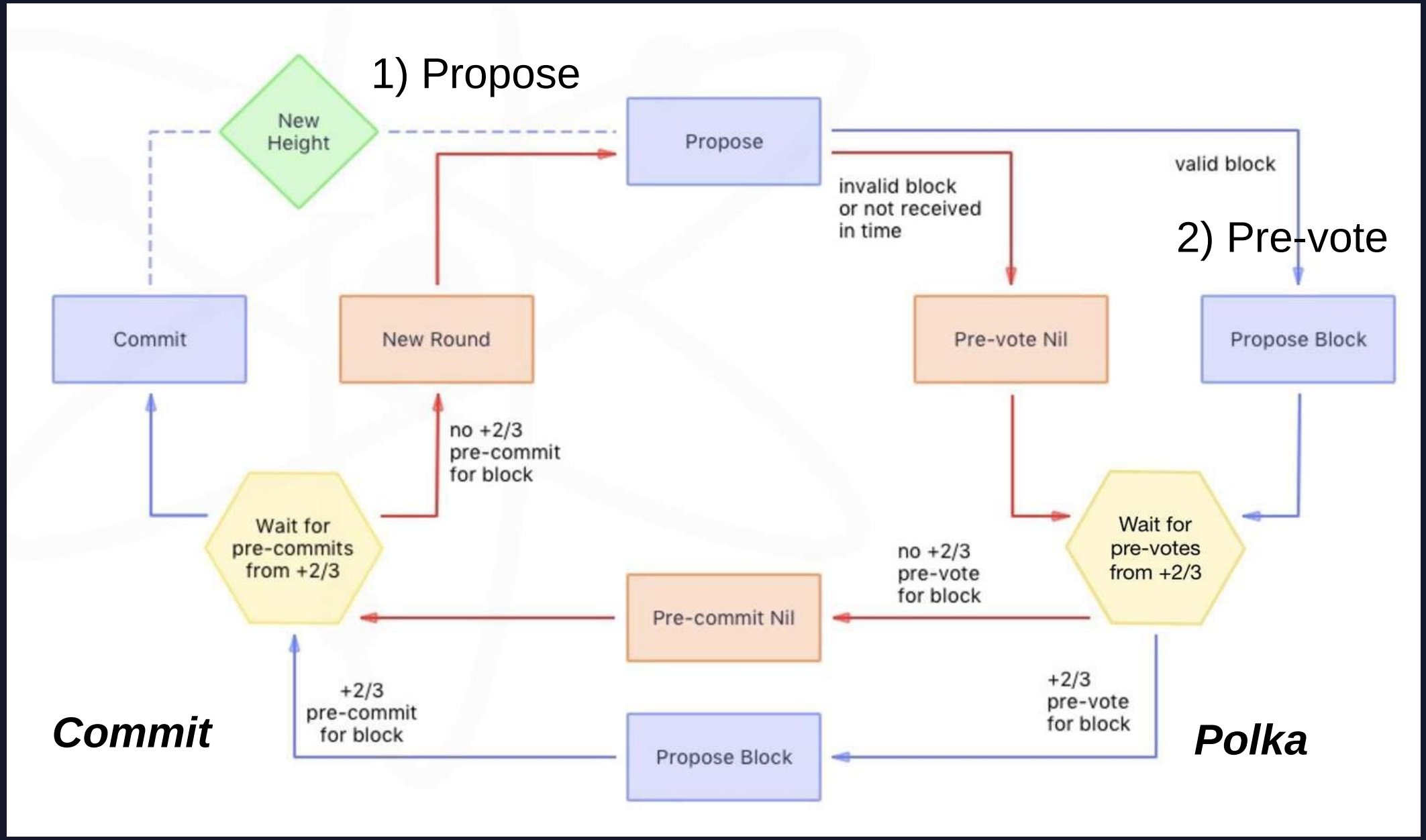


Tendermint BFT

1) Propose



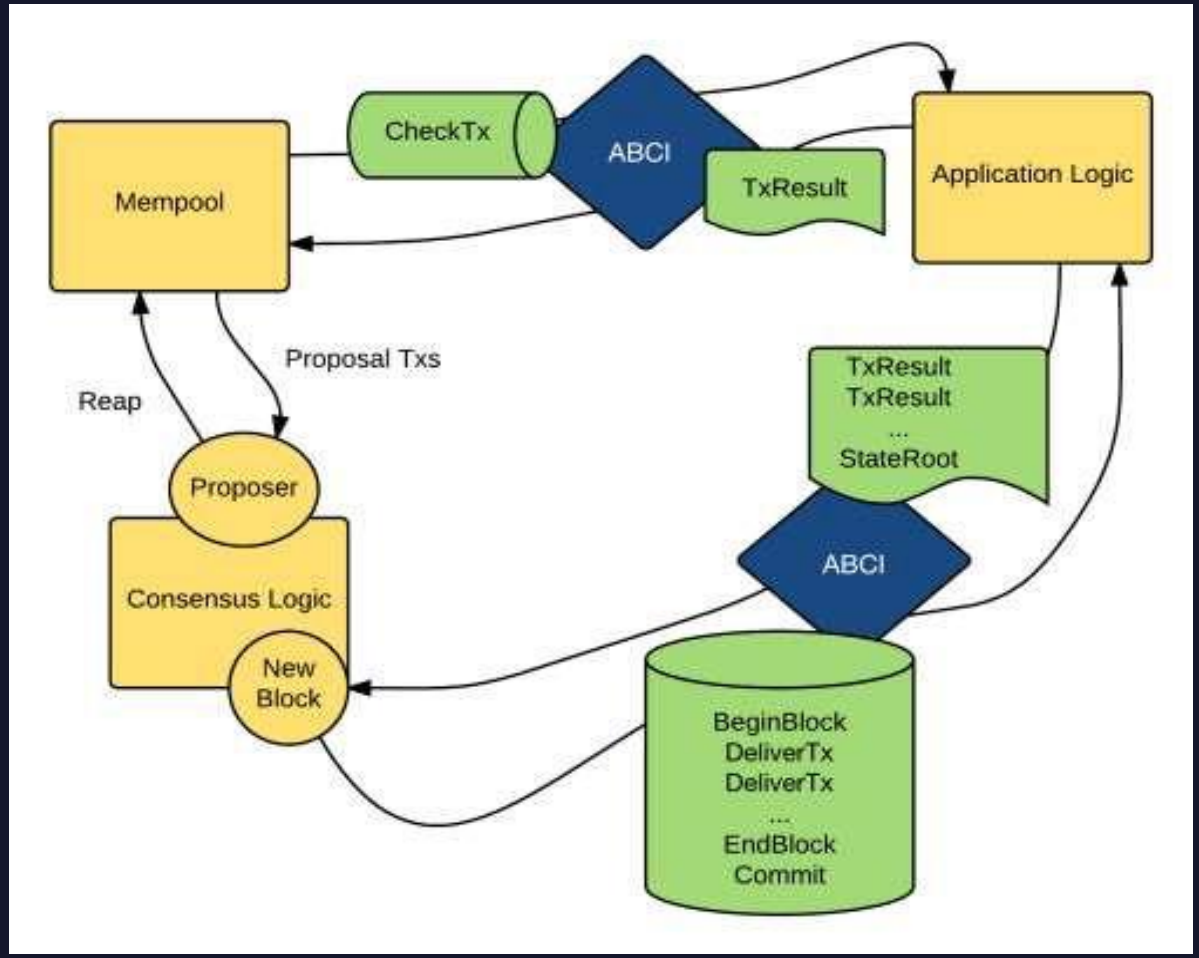
Tendermint BFT



Tendermint BFT

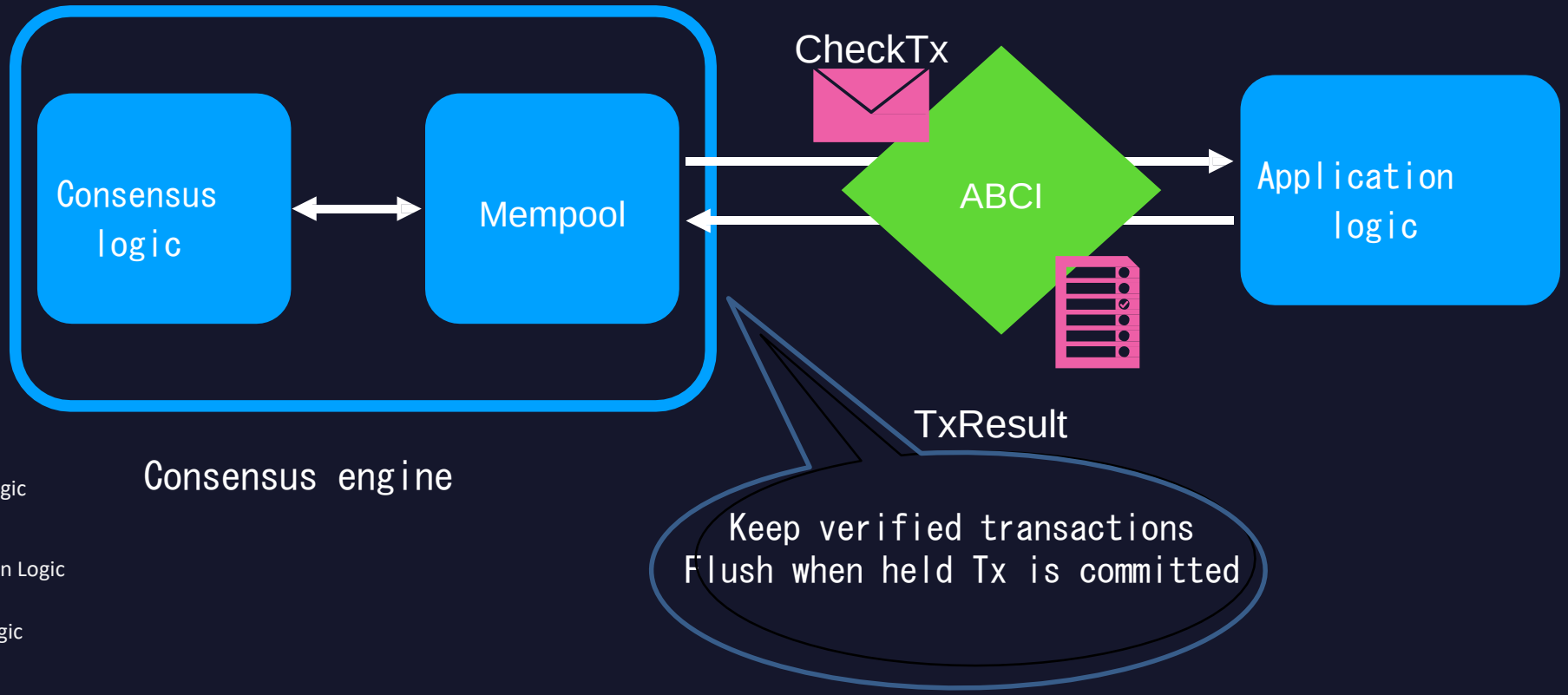
Tendermint Core

- First production grade BFT consensus engine
- Written in Go
- Handles all p2p and consensus logic
- Can handles 100s of validators at sub-5 second block times



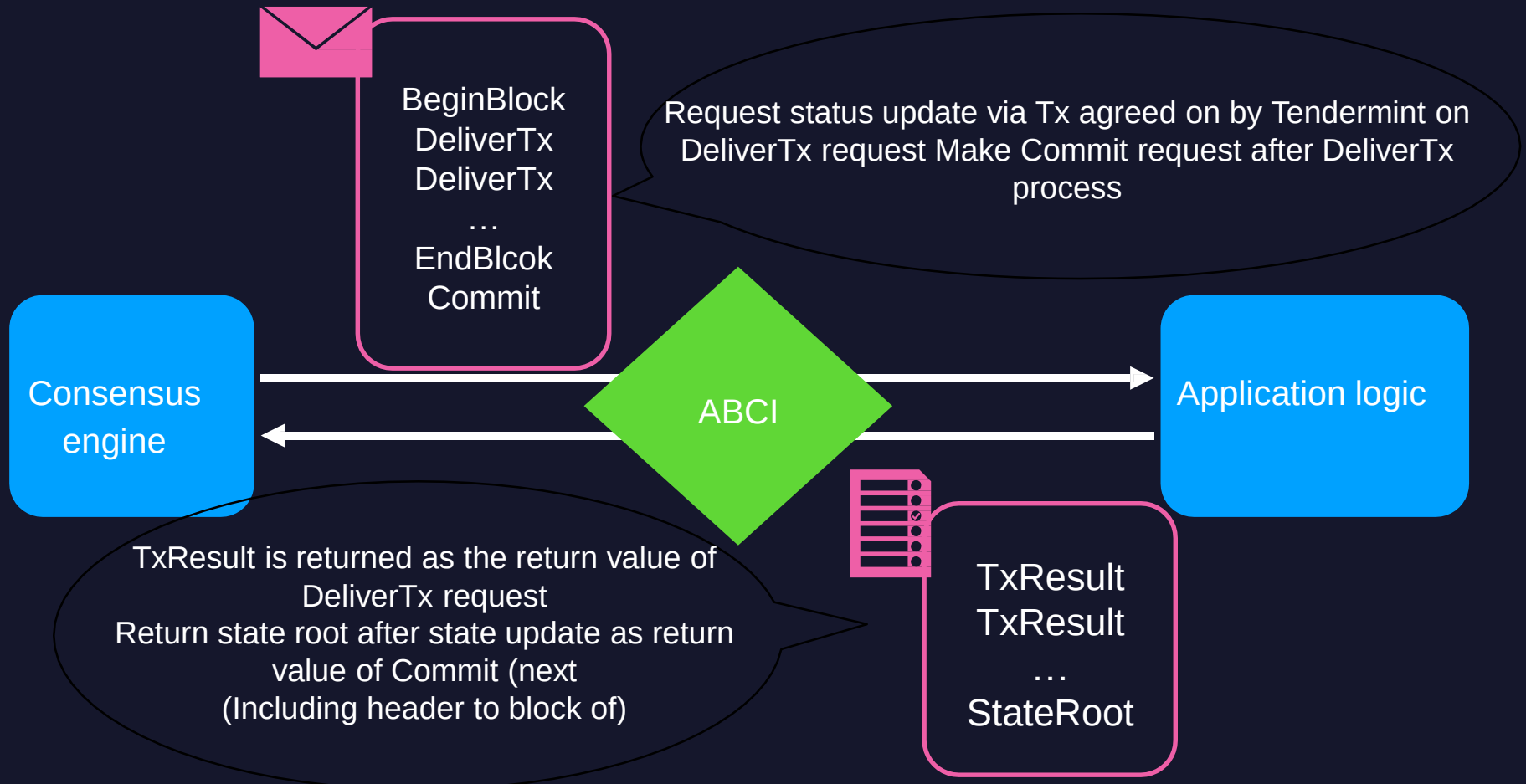
ABCI mempool connection

Ask the application to validate the transaction before committing,
Pool the Txs successfully verified



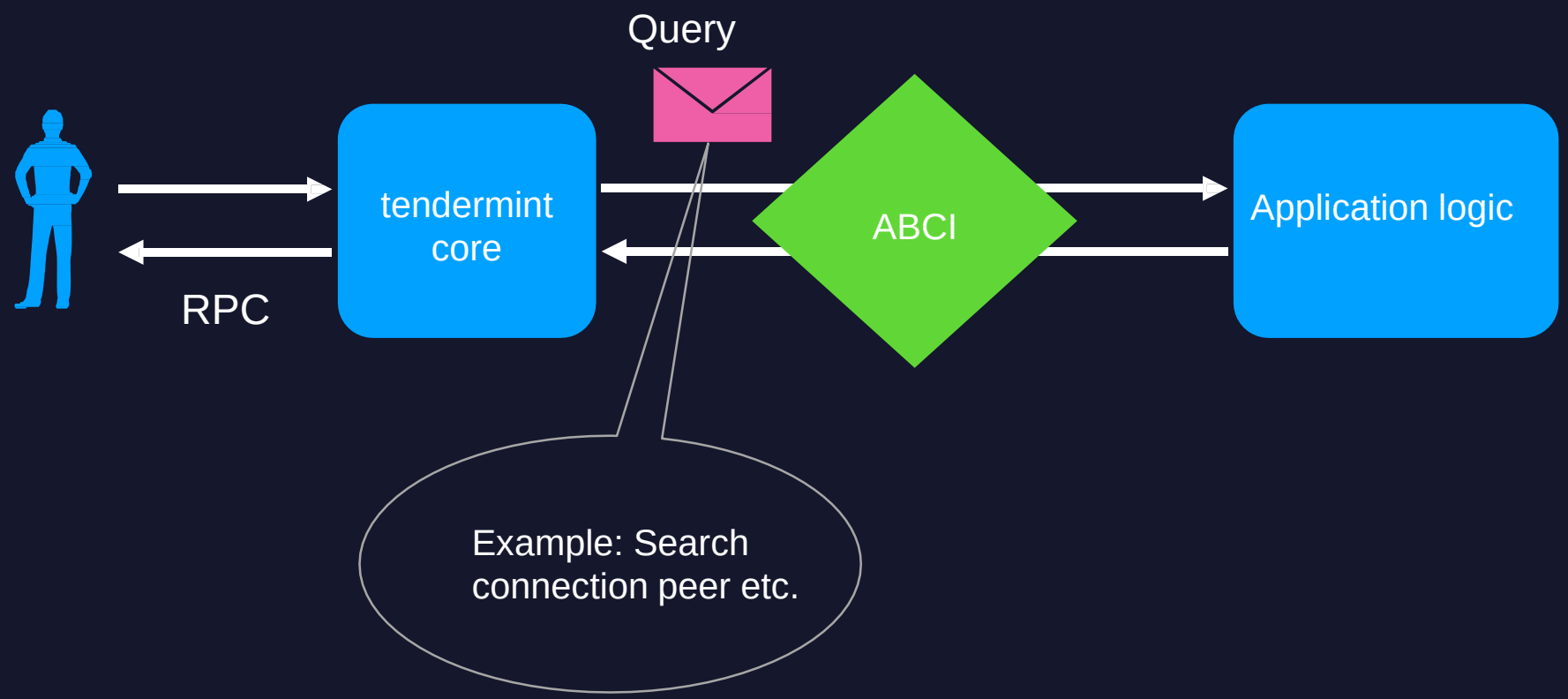
ABCI consensus connection

Perform transaction verification and state transition based on connection committed block information that is agreed upon and occurs when a new block is committed



ABCI query connection

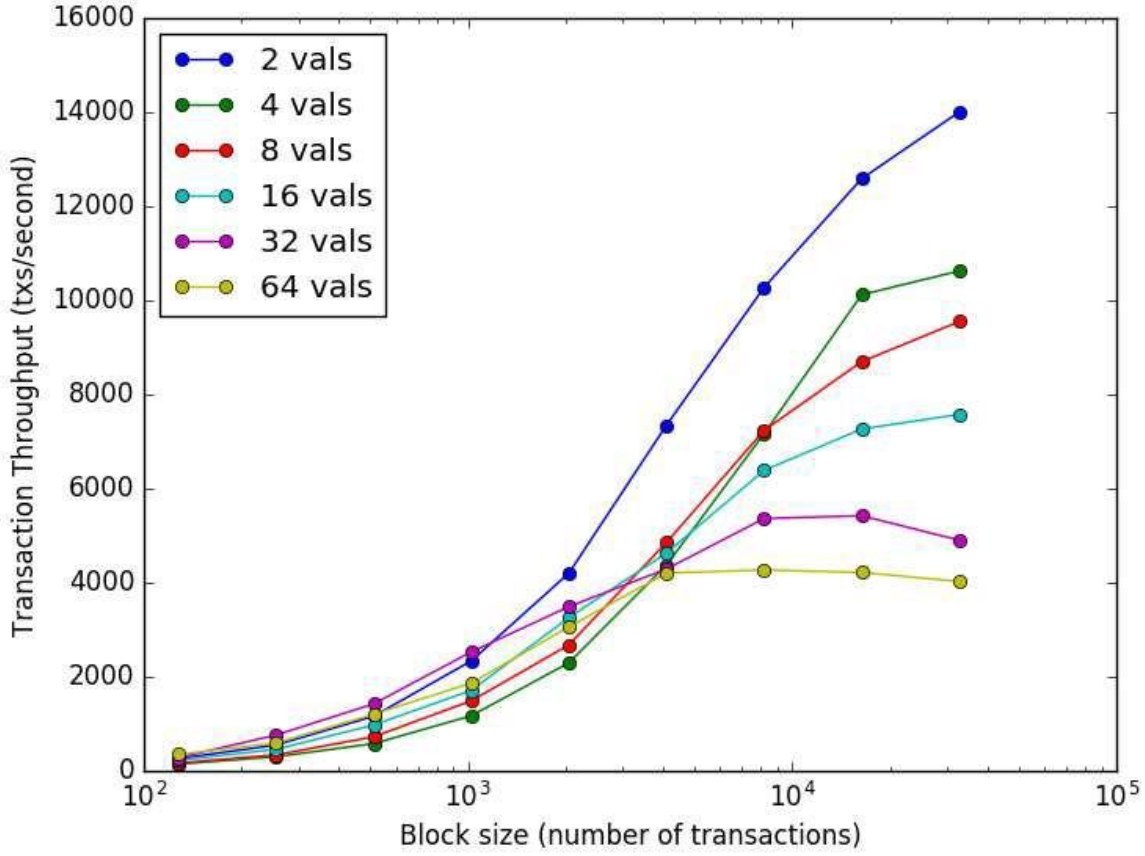
Connections that can always be queried for the application
Always available with RPC from tendermint core



Throughput

	Maximum throughput (*tps*)
Bitcoin	3,2
Ethereum	15
Ethermint	**200**
Tendermint	**~14.000***
Visa	56.000

* Depends on the number of validators and block size



Scalability

- **Vertical scalability**: How much *tps* can a single blockchain archive. Has a cap
- **Horizontal scalability**: Several separate and specialized chains that interact efficiently through a network

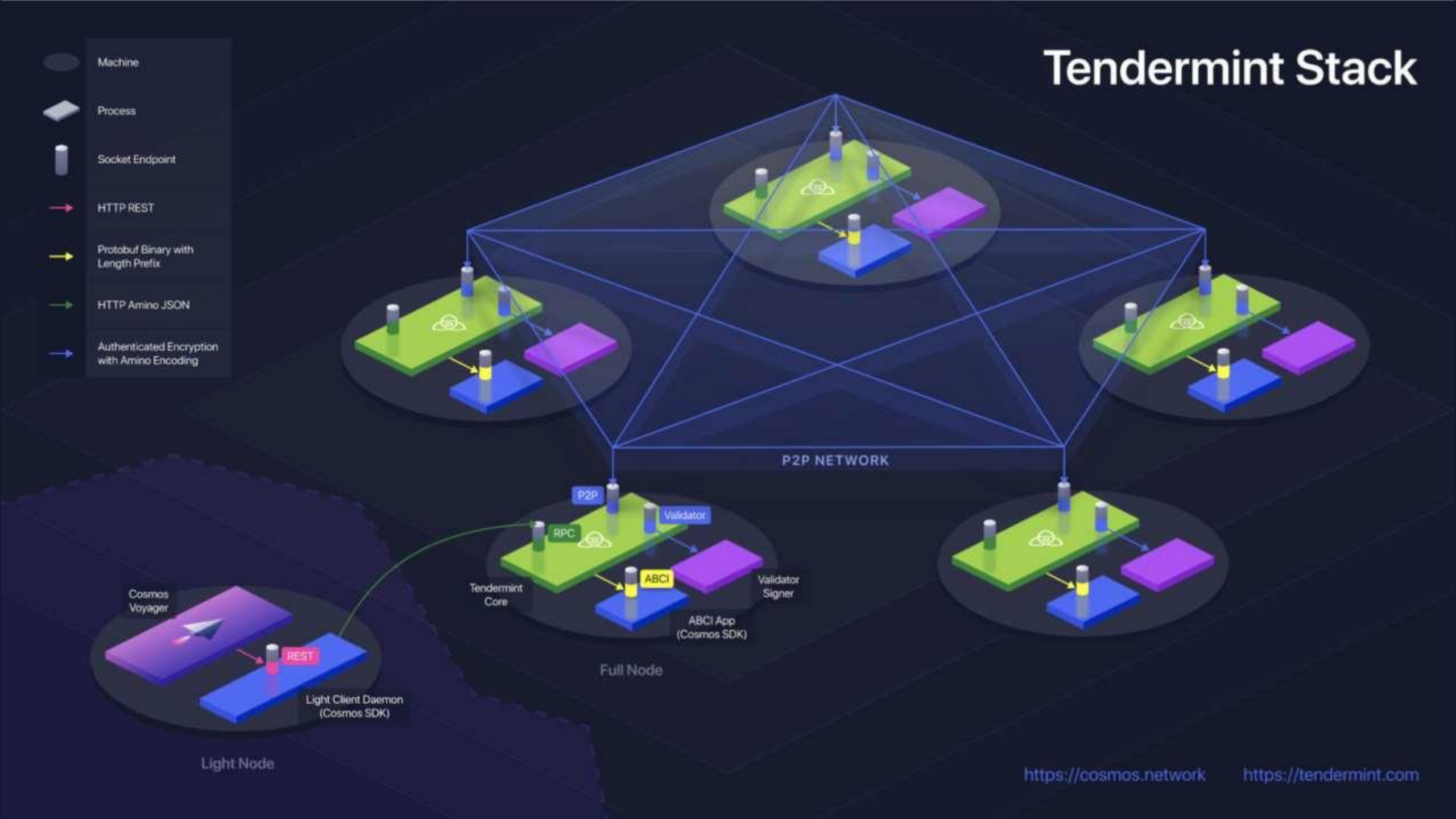
Tendermint BFT Learn More!

Formal specification with proofs of safety and liveness:
https://arxiv.org/abs/1807.04938

Tendermint Diagram:
http://bit.ly/2Nfl9Vb

Casper vs Tendermint:
https://bit.ly/2Iu4Uno

Tendermint Stack



Tendermint Core Learn More!

Tendermint Core Docs:
https://tendermint.com/docs/

Performance Testing Results:
https://bit.ly/2NKCW9n

Ethan Buchman’s Masters Thesis:
https://bit.ly/2S9PyoF

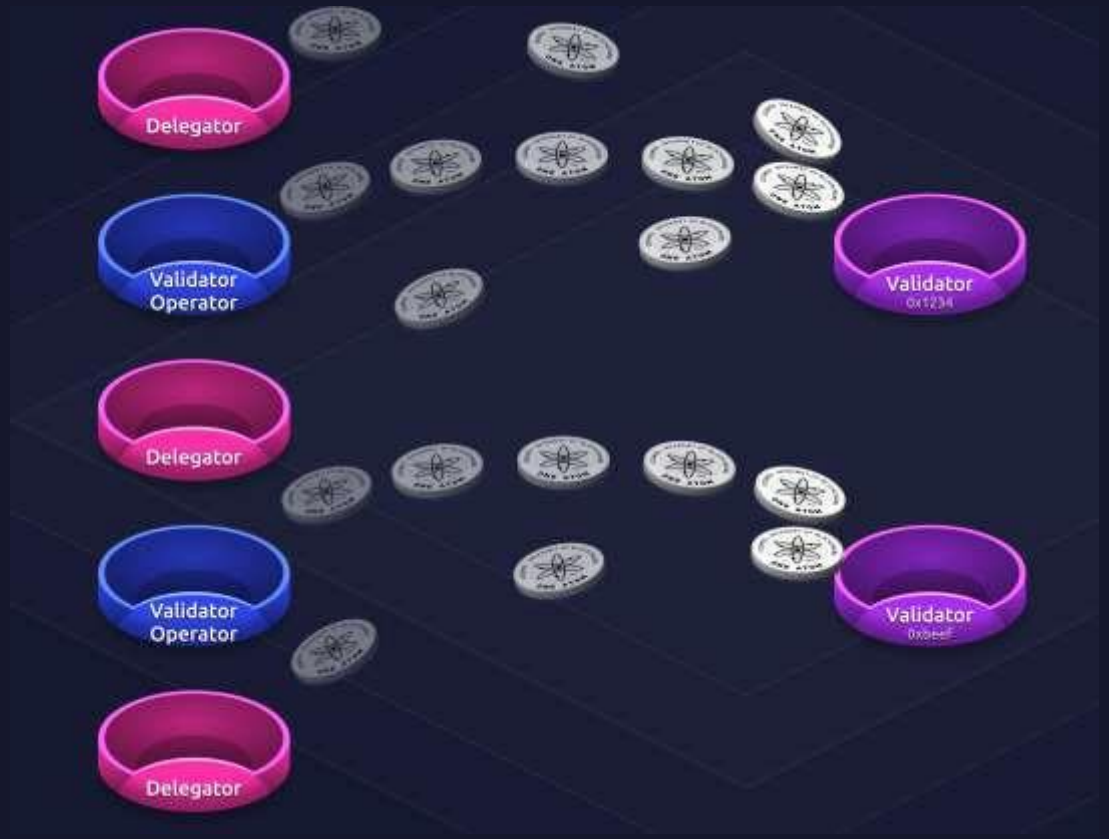
Bonded Proof of Stake

Proof of Stake Basics

- Use bonded tokens as resource limiter for determining voting power
- Eliminates wasteful energy consumption of Proof of Work
- Public permissionless system
- Solve nothing at stake problem through slashing and unbonding periods

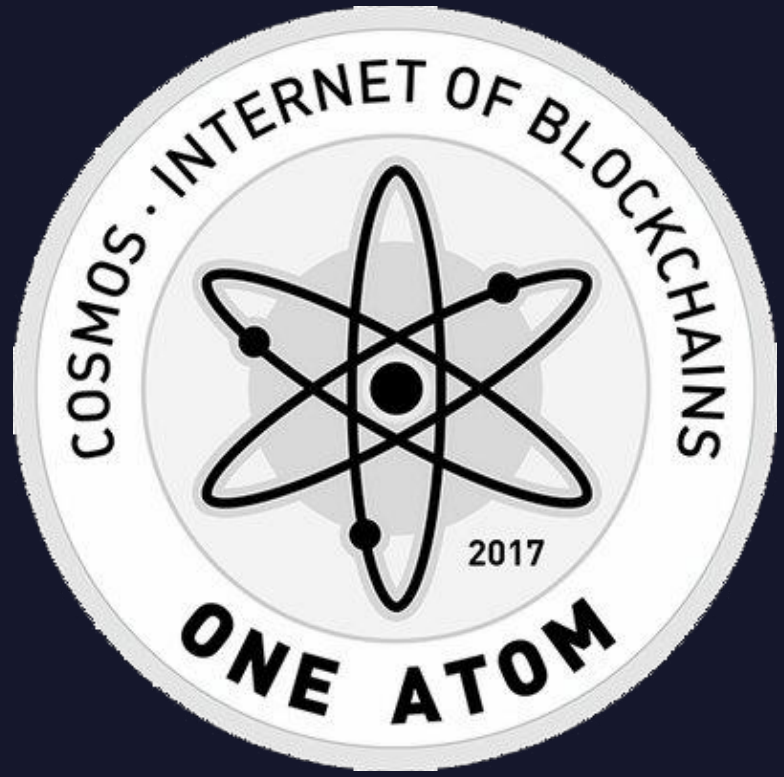
Delegation

- Allow any token holder to be a staker by delegating to a validator
- Skin in the game
- Automatic reward distribution
- Solve stickiness issues through features such as instant redelegation and validator commitments



Multi Token Model

- Specialized staking token for security
- Similar to ASIC security
- Allow fees to be paid in any token to massively improve user experience



Proof of Stake Learn More!

Cosmos Proof of Stake Deep Dive:
https://youtu.be/XxZ04w2x4nk

Multi Token Model Paper:
http://bit.ly/2V6YZXI

Efficient Token Distribution Paper:
http://bit.ly/2SReAhO

Cosmos SDK

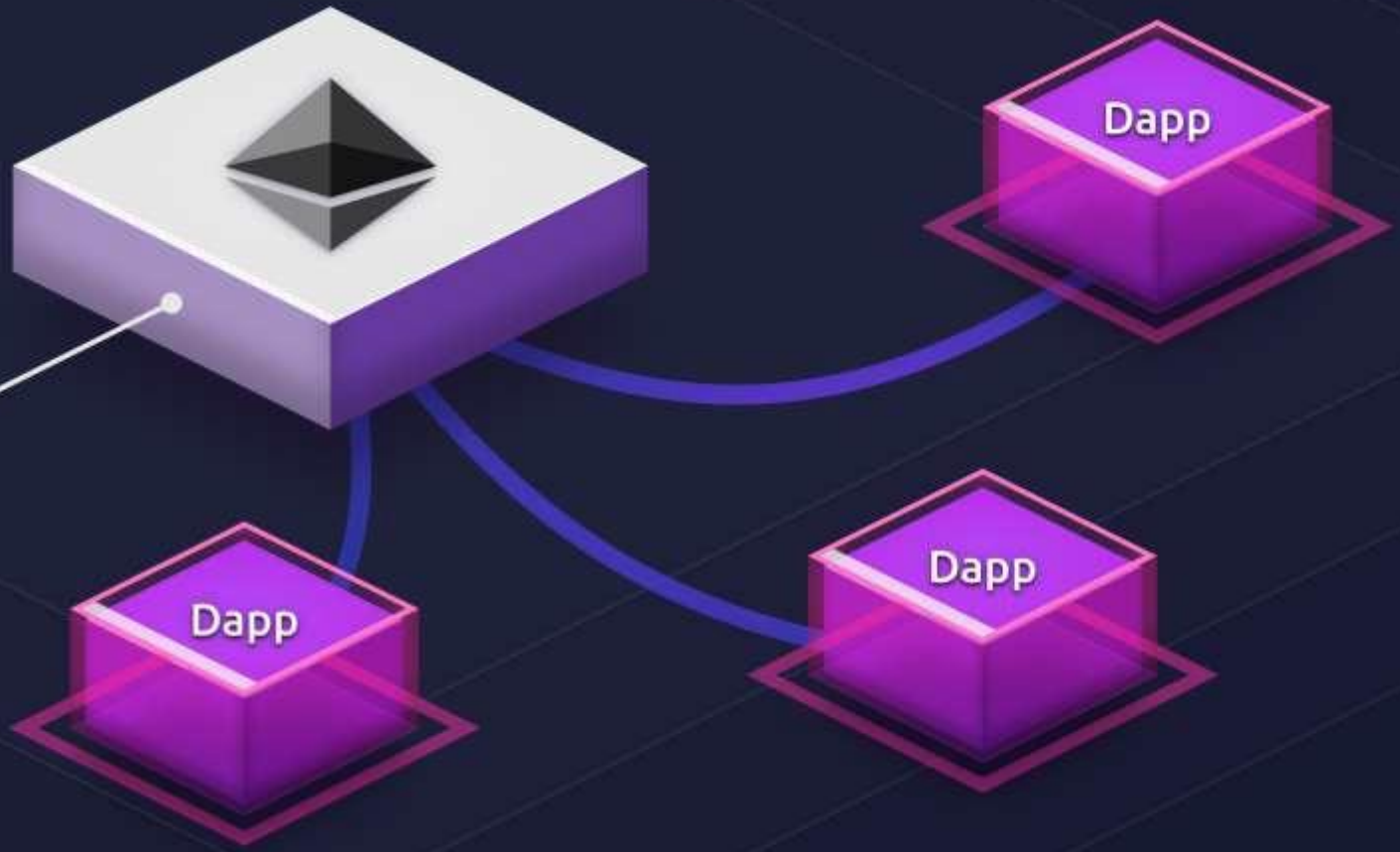
Gen 1: Forked Bitcoin Codebase

Payments
UTXO
Fee by tx size
Bitcoin script
Proof of work

App Developer's
"Zone of Control"

Gen 2: Ethereum Smart Contracts

Ether fee coin
Account model
Patricia tries
EVM
Proof of work



App Developer's
"Zone of Control"

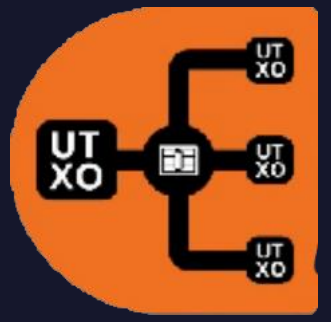
Gen 3: Cosmos SDK

Secure
Modular
Extensible
Golang
Proof of stake

App Developer's
"Zone of Control"

Application Specific Blockchains

- Reduces attack surface
- Efficiency gains due to lower computational overhead
- Fine tune to optimize for your application



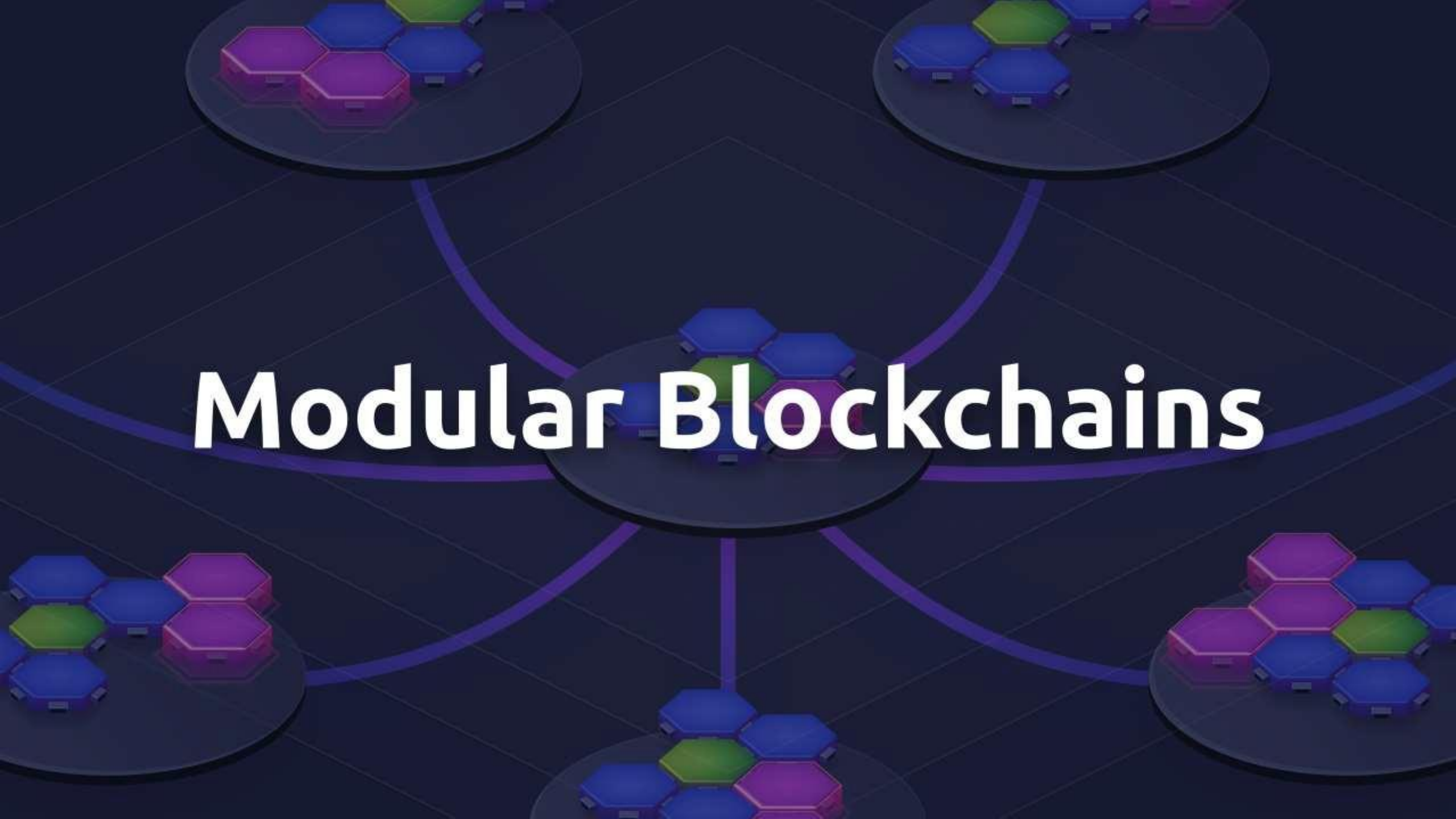
Set	0x01
Remove	0x02
Get	0x03
Compare and Set	0x04
Validator Set Change	0x05
Validator Set Read	0x06
Validator Set CAS	0x07

Cosmos SDK: Developer Friendly

- Written in Go
- Like Ruby-on-Rails for building blockchains
- Completely open source and available on GitHub
- Secured by the principal of least authority

Cosmos SDK: Modular & Extensible

- Modular architecture for plug-and-play development
- Simply plug ready-built modules to add features to your blockchain
- Build new modules and share them downstream to enrich and contribute to the Cosmos-SDK ecosystem


Modular Blockchains

SDK + Peggy == Cosmos Hub

EVM + Shared Security == Ethermint

github.com/cosmos/ethermint

Microservices == IRIS Network

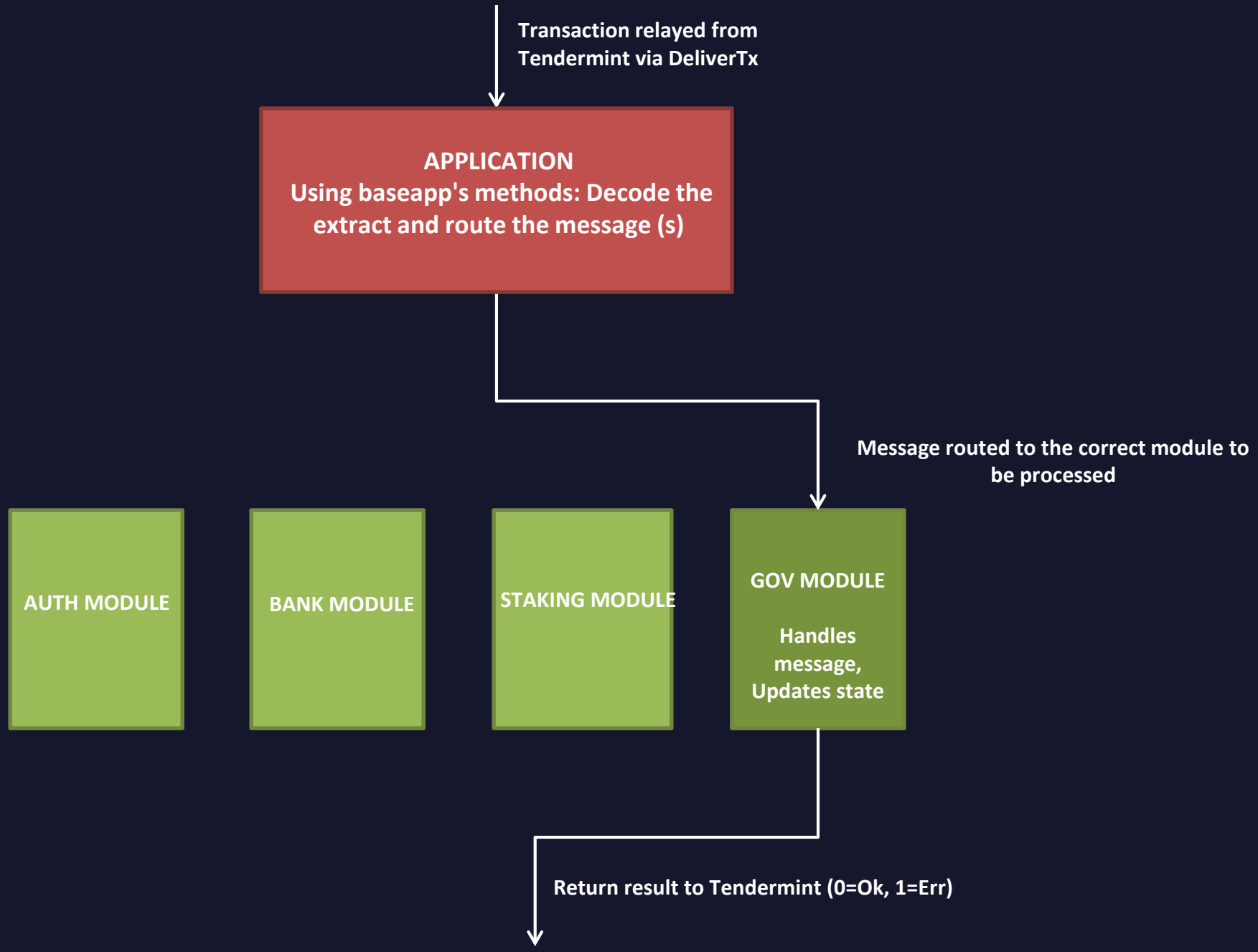
irisnet

Rep, Auctions, BW Fees == LINO

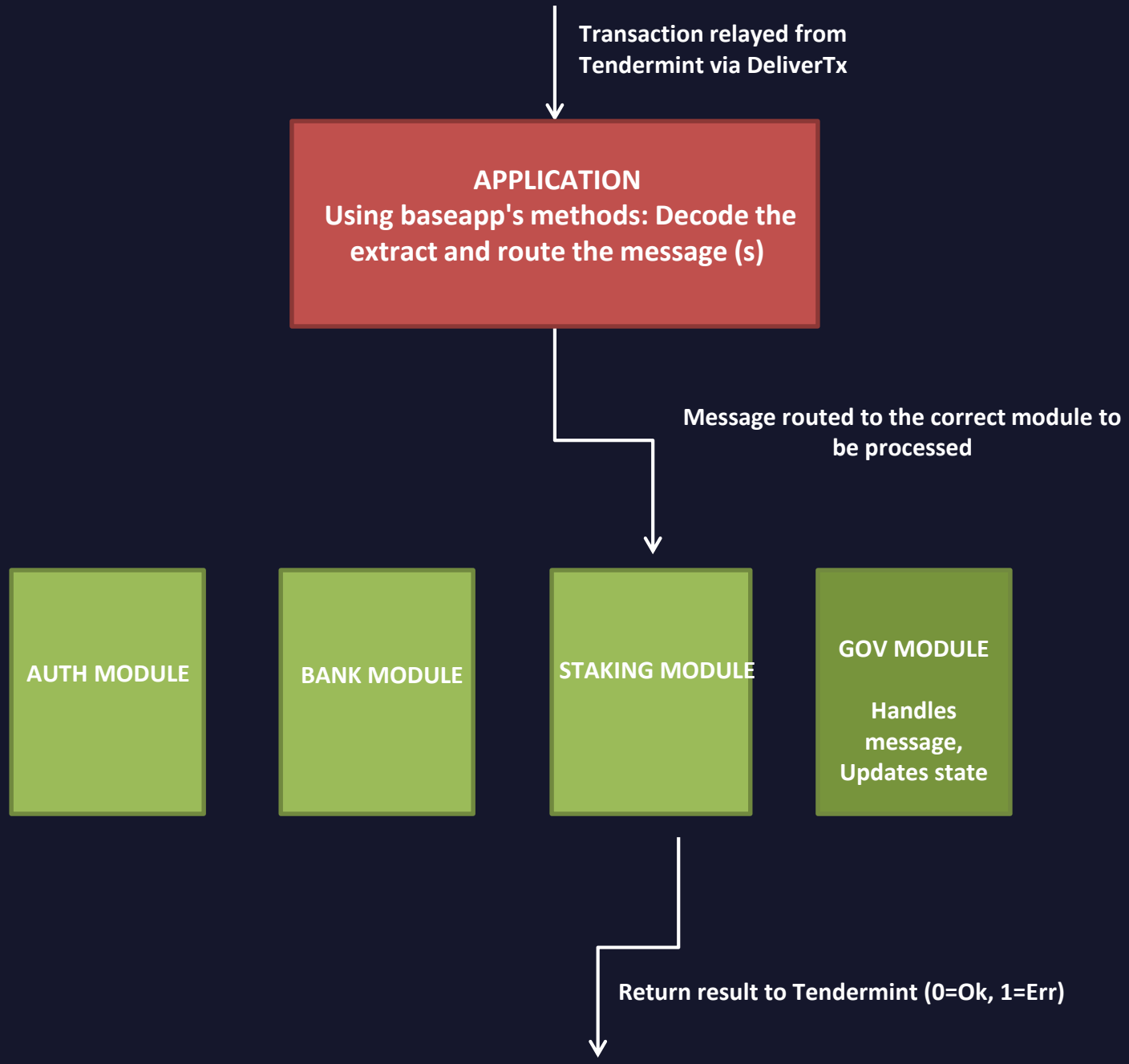
Peggy + Plasma == Fourth State

What Will You Build?

Own App


Transaction relayed from Tendermint via DeliverTx
APPLICATION
Using baseapp's methods: Decode the extract and route the message (s)
Message routed to the correct module to be processed
AUTH MODULE
BANK MODULE
STAKING MODULE
GOV MODULE
Handles message, Updates state
Return result to Tendermint (0=Ok, 1=Err)

Own App


Transaction relayed from Tendermint via DeliverTx
APPLICATION
Using baseapp's methods: Decode the extract and route the message (s)
Message routed to the correct module to be processed
AUTH MODULE
BANK MODULE
STAKING MODULE
GOV MODULE
Handles message, Updates state
Return result to Tendermint (0=Ok, 1=Err)

Own App

1. Design the application.
2. Begin the implementation of your application in ./app.go.
3. Start building your module by defining some basic Types.
4. Create the main core of the module using the Keeper.
5. Define state transitions through Msgs and Handlers.
    SetName
    BuyName
6. Make views on your state machine with Queriers.
7. Register your types in the encoding format using sdk.Codec.
8. Create CLI interactions for your module.
9. Create HTTP routes for clients to access your nameservice
10. Import your module and finish building your application!
11. Create the nsd and nscli entry points to your application.
12. Setup dependency management using dep.
13. Build and run the example.
14. Run REST routes.

Cosmos SDK Learn More!

Cosmos SDK Tutorial:
https://cosmos.network/docs/tutorial/

Cosmos SDK Repo:
https://github.com/cosmos/cosmos-sdk

The Case for Application Specific Blockchains:
http://bit.ly/2SMiCI7

Low Level Libs

Amino

- Improvement to Protobuf standard
- Naturally support interfaces instead of OneOf
- Deterministic
- Generate proto files from Go code


```
amino.RegisterInterface((*MyInterface1)(nil), nil)
amino.RegisterInterface((*MyInterface2)(nil), nil)
amino.RegisterConcrete(MyStruct1{}, "com.tendermint/MyStruct1", nil)
amino.RegisterConcrete(MyStruct2{}, "com.tendermint/MyStruct2", nil)
amino.RegisterConcrete(&MyStruct3{}, "anythingcangoinhereifitsunique", nil)
```

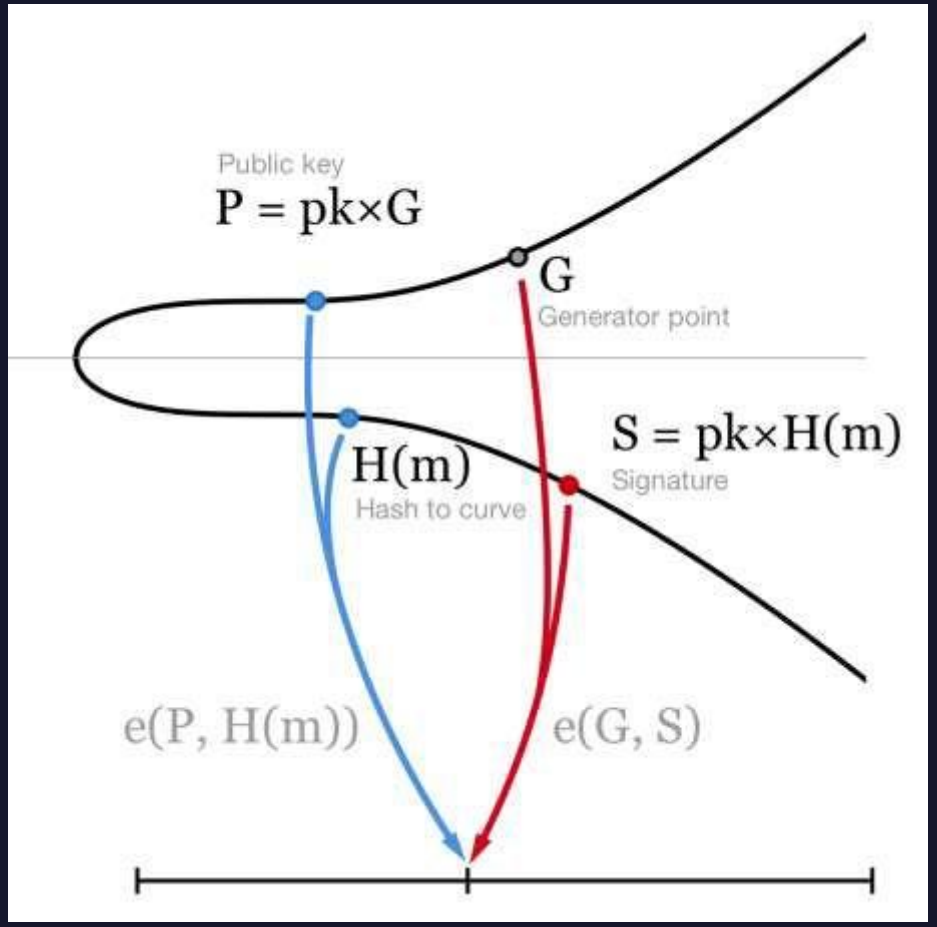
IAVL+ Tree

- Self-balancing AVL tree
- All values are stored at leaves
- Immutable with snapshots and caching
- All operations log(N)
- No hashing keys required

```
writing down, my checksum
waiting for the, data to come
no need to pray for integrity
thats cuz I use, a merkle tree

grab the root, with a quick hash run
if the hash works out,
it must have been done

theres no need, for trust to arise
thanks to the crypto
now that I can merkleyes

take that data, merklize
ye, I merklize ...

then the truth, begins to shine
the inverse of a hash, you will never find
and as I watch, the dataset grow
producing a proof, is never slow

Where do I find, the will to hash
How do I teach it?
It doesn't pay in cash
Bitcoin, here, I've realized
Thats what I need now,
cuz real currencies merklize
-EB
```

Crypto

- Cryptography library with built-in Amino support
- Abstracted multisignature pubkeys
- BGLS Aggregate Signature implementations
- BGLS verifier in EVM



Low Level Libs Learn More!

Go-Amino Repo:
https://github.com/tendermint/go-amino

IAVL+ Repo:
https://github.com/tendermint/iavl

BGLS Repo:
https://github.com/Project-Arda/bgls

Alternative Frameworks

for state machine

1. Install

```
$ npm install lotion
```

2. Write your state machine

```
// app.js
let lotion = require('lotion')

let app = lotion({
    initialState: {
        count: 0
    }
})

app.use(function(state, tx) {
    if (state.count === tx.nonce) {
        state.count++
    }
})

app.start().then(function(appInfo) {
    console.log(`app started. gci: ${appInfo.GCI}`)
})
```

3. Run it and query the state

```
$ node app

# in another terminal:

$ npx lotion-cli state <GCI>
```

IOV Weave

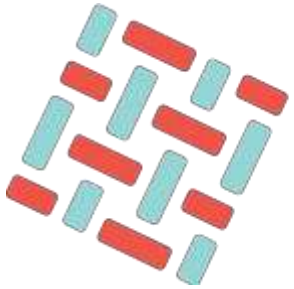
- Fork of the Cosmos SDK maintained by IOV
- Simpler version of the SDK with more limited features
- Second Go Framework

Potential Future Frameworks

Chainmint

parity substrate

NERVOS CKB



HYPERLEDGER FABRIC

Ethermint

Alternative Frameworks Learn More!

Lotion JS Repo:
https://github.com/nomic-io/lotion

Weave Repo:
https://github.com/iov-one/weave

Ethermint

Module: Ethereum Virtual Machine

EVM in the Cosmos SDK module interface

- Account database, state tree
- EVM module can run Ethereum txs
- EVM module calls into Cosmos SDK modules
- Shared state view - one token

Ethermint: Cosmos SDK + EVM

Best of both worlds

- Scalability of Tendermint
- Power of the Cosmos SDK
- Existing ecosystem of Ethereum contracts, dev tooling

EVM module from TurboGeth

- DB performance improvements
- Flexible SDK module interface

Two Ways to Use EVM Module

Ethermint as a blockchain

- Cosmos PoS chain for smart contracts

EVM as a library

- Deploy your own Cosmos chain with EVM support
- Add in other SDK modules or write your own
- Flexibility in token choice & economic model

Ethermint as a Blockchain

One chain for many EVM applications

- Hard spoon of account balances
- Sovereign chain, own token
- Governance, staking, slashing
- Fully web3 compatible
- IBC connections to other chains

EVM Module for SDK Zones

EVM module for your
Tendermint/Cosmos chain

- Full EVM functionality set
- Include other Cosmos SDK modules
- Utilize existing Solidity contracts
- Gradually port parts of logic to native code

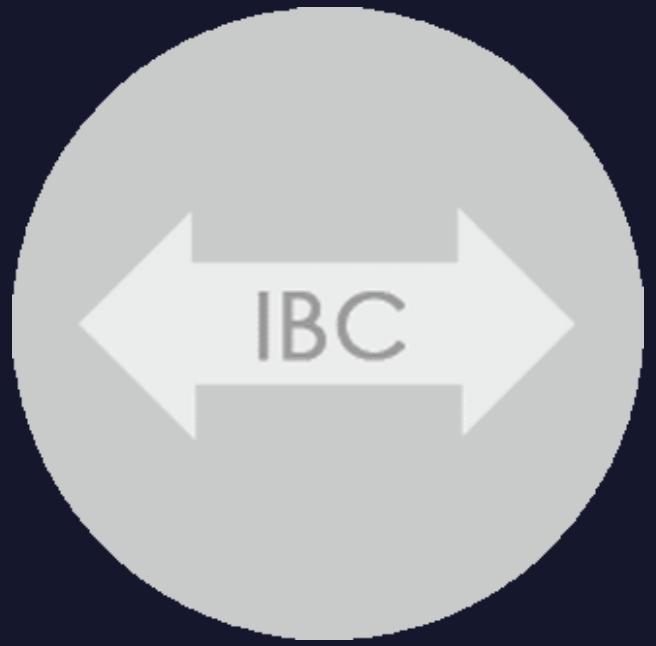
Ethermint 2.0

- EVM client built using the Cosmos SDK
- Will be fully Web3 compatible
- Can deploy existing Ethereum dapps / smart contracts
- Can add your own precompiles
- Working with TurboGeth team to build and optimize

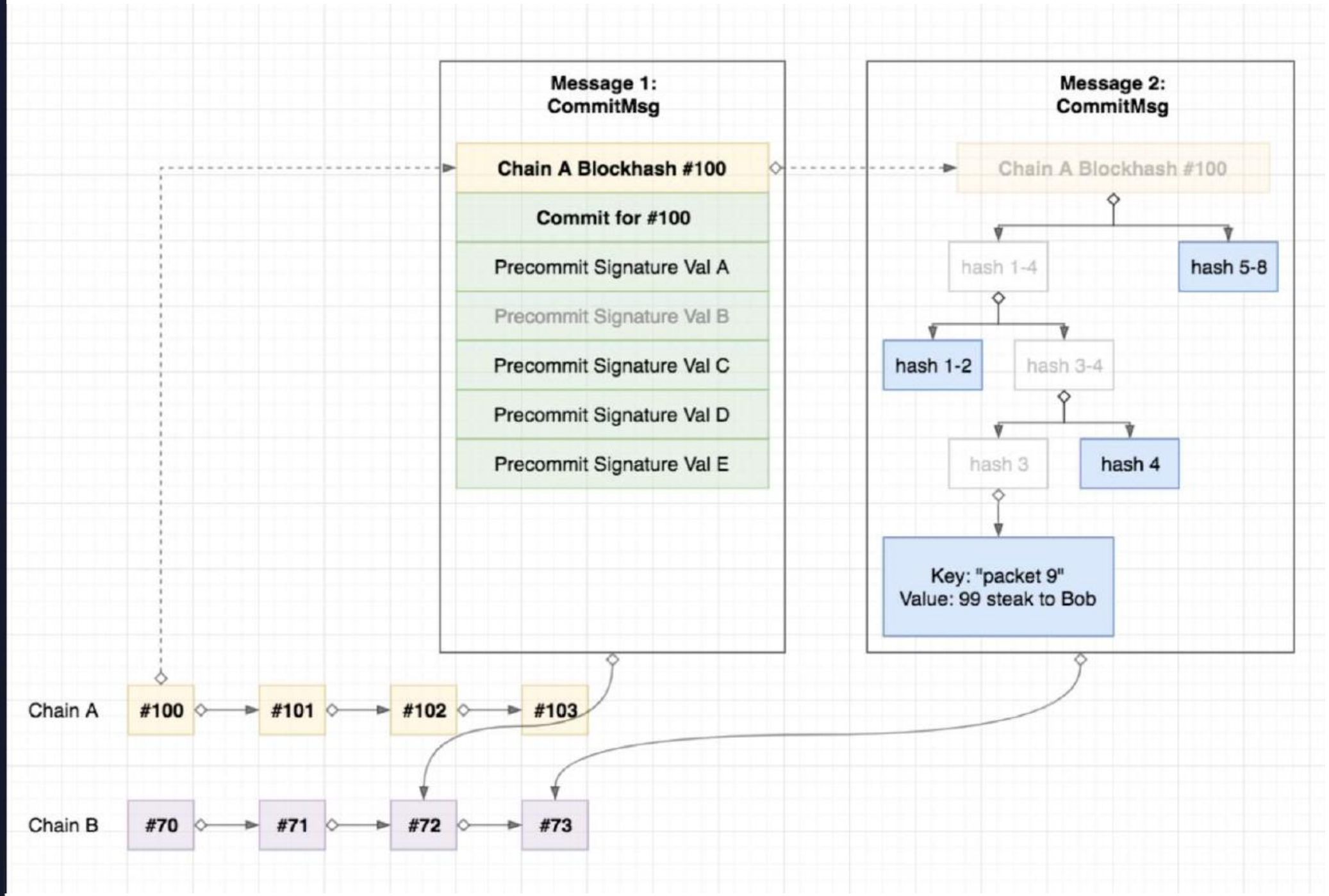
Ethermint Learn More!

Ethermint DevCon Presentation:
https://youtu.be/VCLbS1Oks8A

Ethermint Repo:
https://github.com/cosmos/ethermint



Inter Blockchain Communication


Message 1:
CommitMsg
Chain A Blockhash #100
Commit for #100
Precommit Signature Val A
Precommit Signature Val B
Precommit Signature Val C
Precommit Signature Val D
Precommit Signature Val E
Message 2:
CommitMsg
Chain A Blockhash #100
hash 1-4
hash 5-8
hash 1-2
hash 3-4
hash 3
hash 4
Key: "packet 9"
Value: 99 steak to Bob
Chain A
#100
#101
#102
#103
Chain B
#70
#71
#72
#73

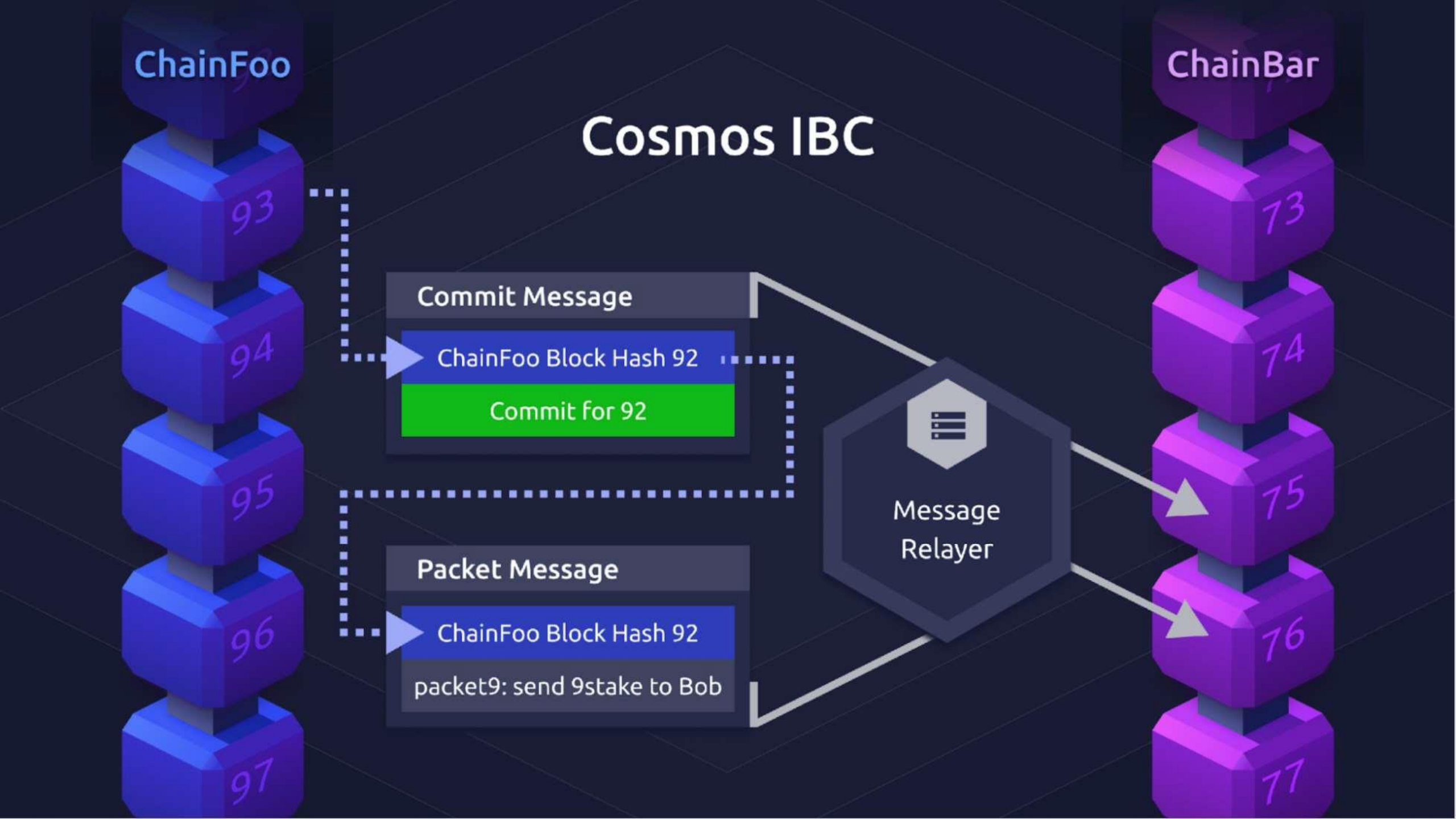
Cosmos IBC

Chain A and chain B are light clients of each other.

IBC Packet (kinda like TCP/IP):

1. Prove the hash commit w/ signatures
2. Prove the packet w/ Merkle proof

Or, IBC State proves state.


ChainFoo
ChainBar
Cosmos IBC
93
94
95
96
97
73
74
75
76
77
Commit Message
ChainFoo Block Hash 92
Commit for 92
Packet Message
ChainFoo Block Hash 92
packet9: send 9stake to Bob
Message
Relayer

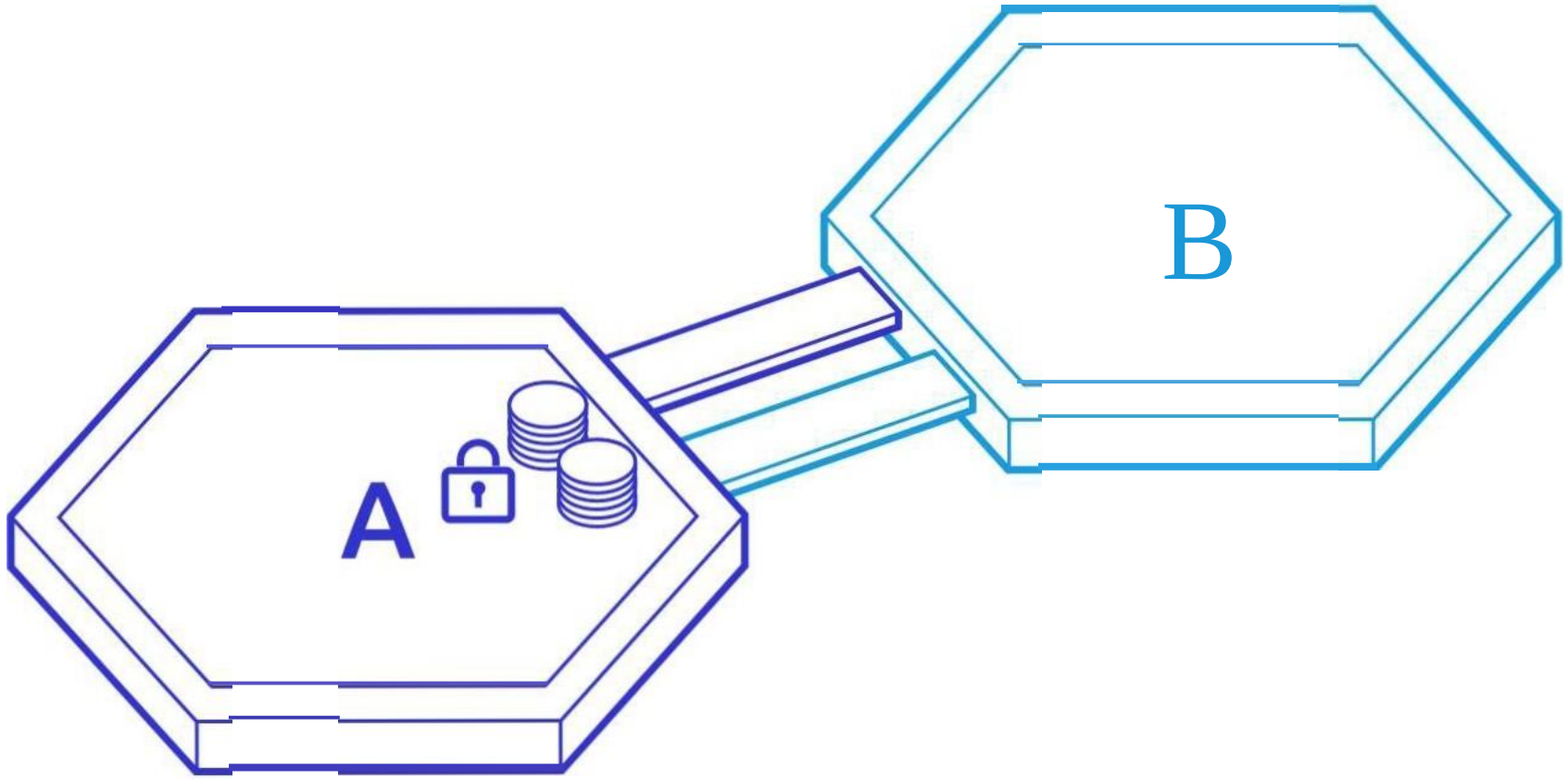
Horizontal Scalability

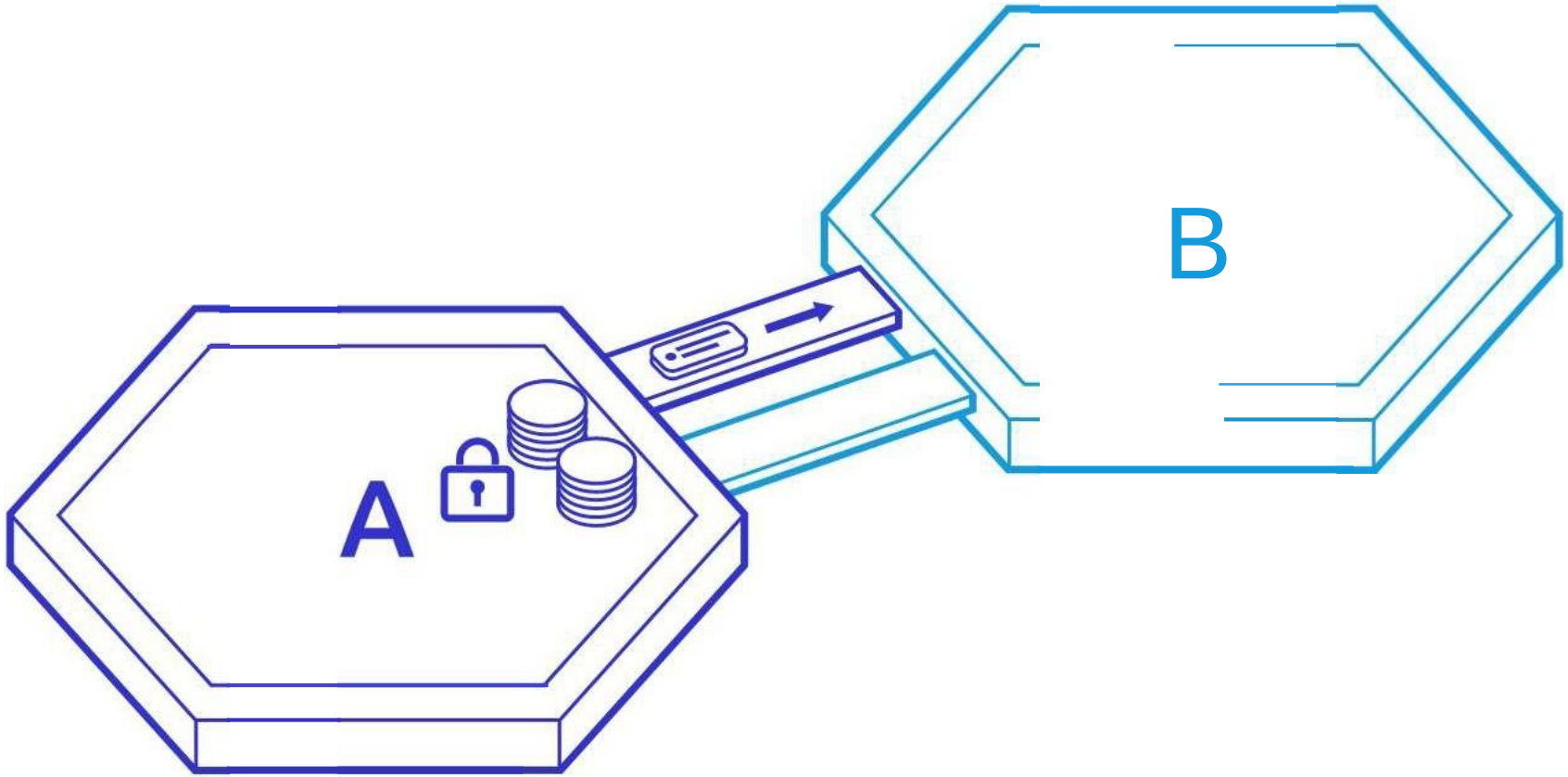
- Application-based sharding is logical as it minimizes bottleneck
- You only have to be a full node for applications you care about

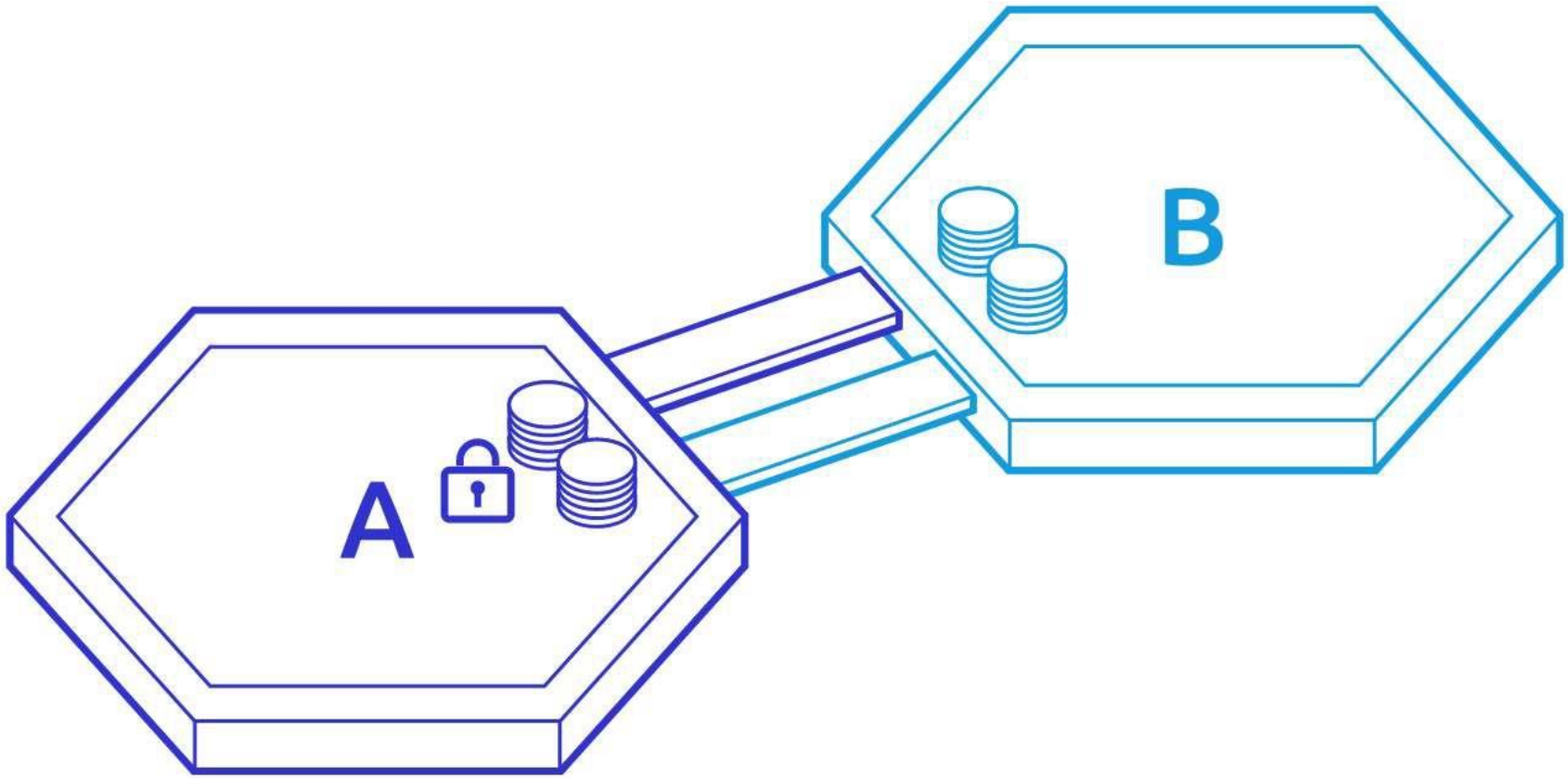
Packet Types

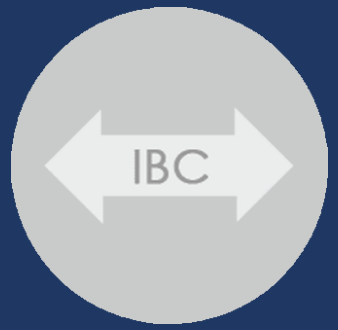
- Equivalent to Application Layer in Internet stack

- Different types of packet structures/handling protocols
- Token Transfers
- Non Fungible Assets
- Data
- Agoric ERTP


A
B


A
B


A
B

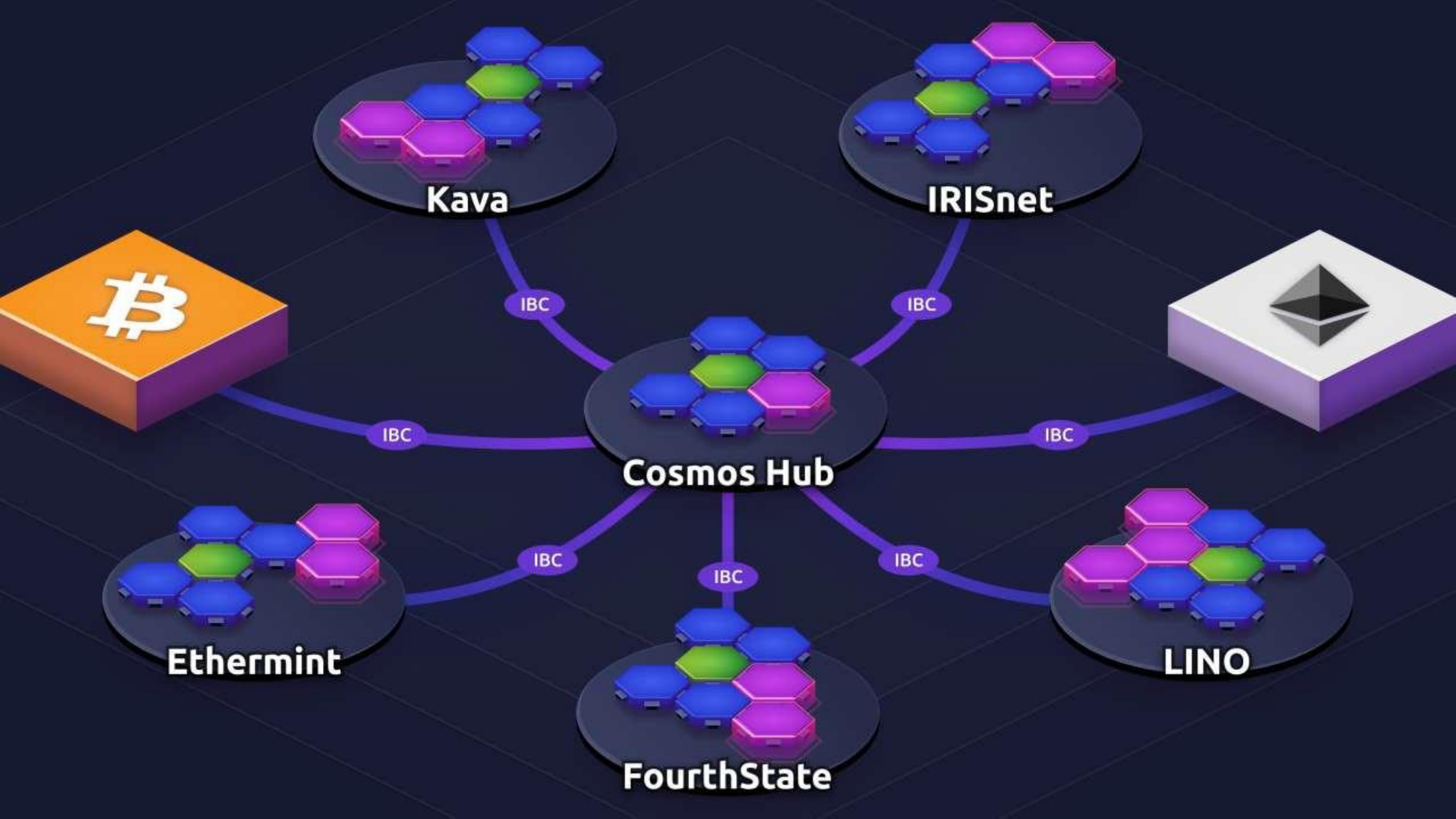


IBC Learn More!

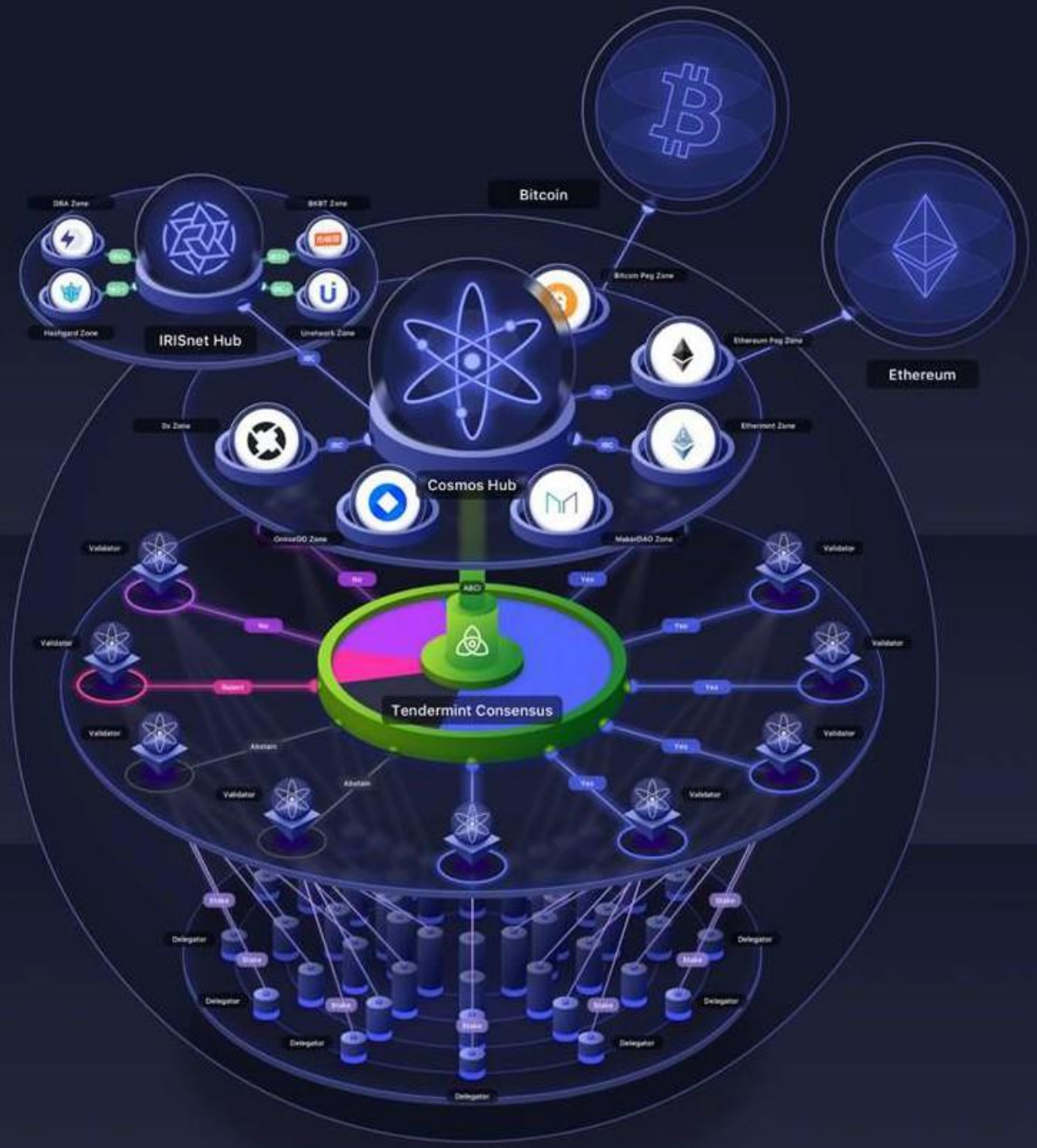
ZK Summit IBC Presentation:
https://youtu.be/cjfYThAk06w

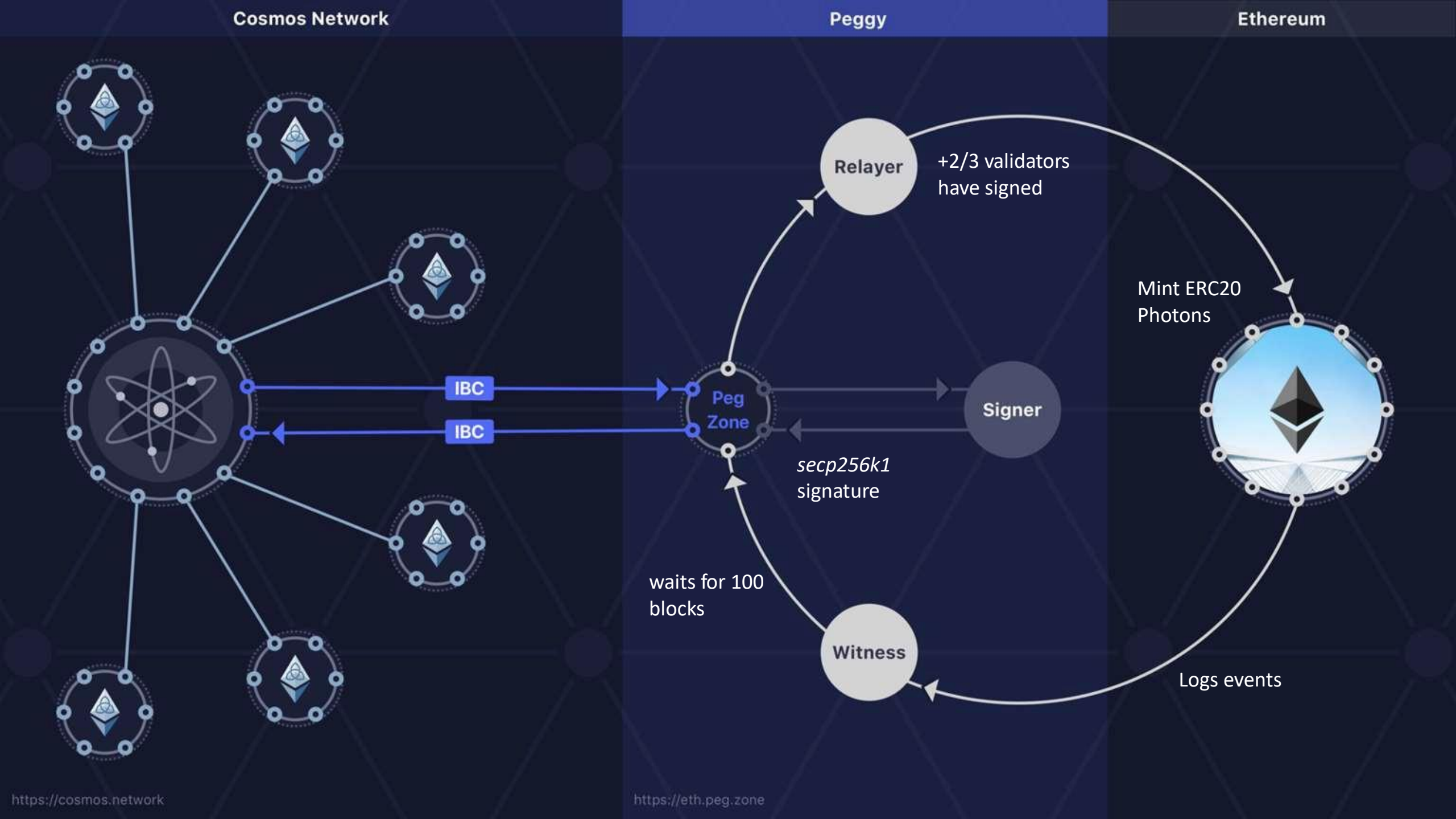
EdCon IBC Presentation:
https://youtu.be/enPetIum0d0

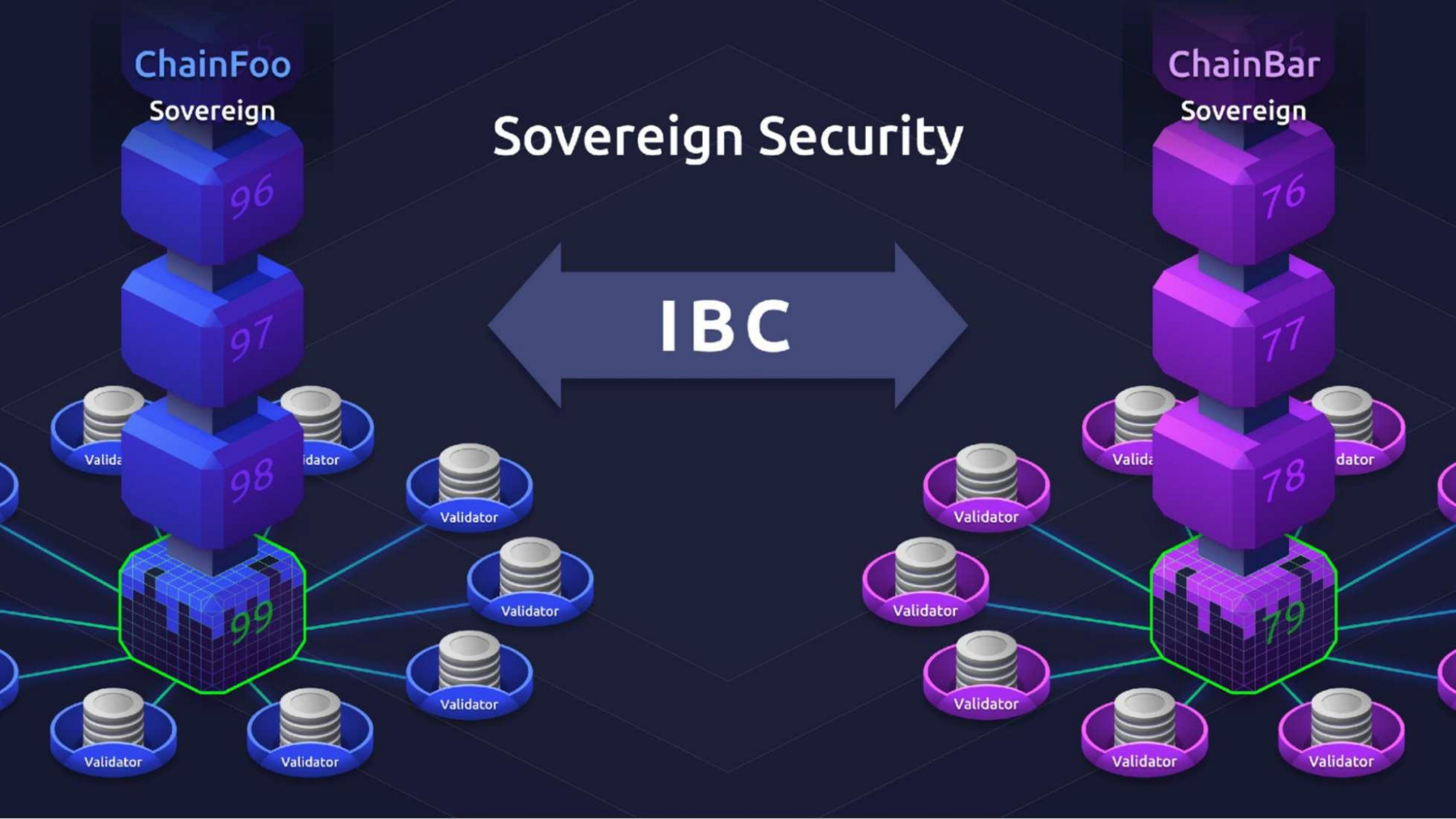
IBC Webinar:
https://youtu.be/m_b_Noe70Vc

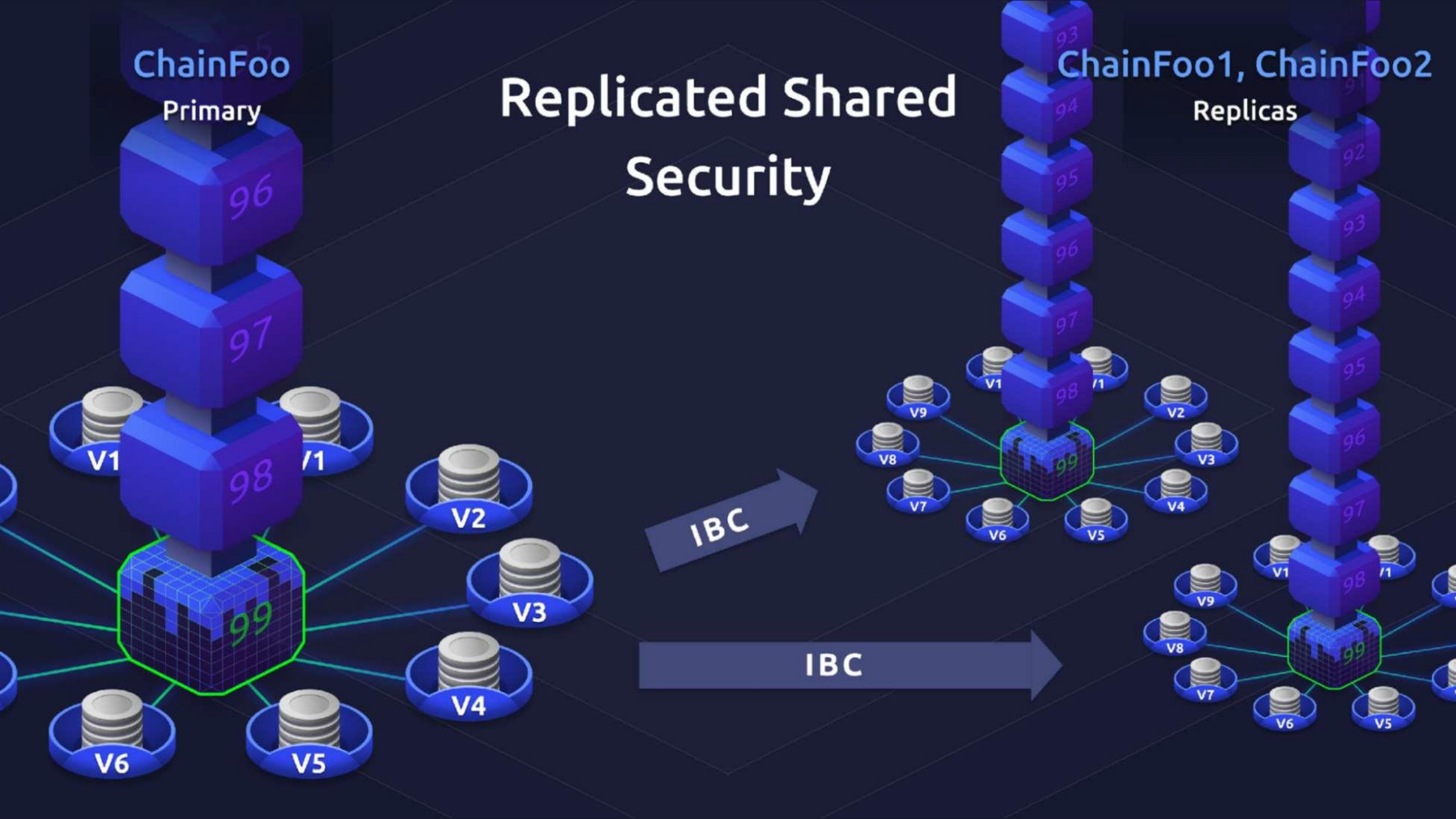

Kava
IRISnet
IBC
IBC
IBC
IBC
Cosmos Hub
IBC
IBC
IBC
Ethermint
LINO
FourthState

@cosmos


Bitcoin
Ethereum
IRISnet Hub
Cosmos Hub
Tendermint Consensus
Zones
Validators
Delegators


Cosmos Network
Peggy
Ethereum
Relayer
+2/3 validators
have signed
Mint ERC20
Photons
IBC
IBC
Peg
Zone
Signer
secp256k1
signature
waits for 100
blocks
Witness
Logs events
https://cosmos.network
https://eth.peg.zone


ChainFoo
Sovereign
ChainBar
Sovereign
Sovereign Security
IBC
96
97
98
99
76
77
78
79
Validator


ChainFoo
Primary
Replicated Shared Security
ChainFoo1, ChainFoo2
Replicas
96
97
98
99
V1
V2
V3
V4
V5
V6
V7
V8
V9
IBC
IBC

Cosmos Hub Learn More!

CESC Interchain Scaling Presentation:
https://youtu.be/D4Q-gA_kPrU

PolkaDot vs Cosmos:
https://forum.cosmos.network/t/polkadot-vs-cosmos/1397

Cosmos Intro:
https://cosmos.network/intro

Learn more

tendermint.com

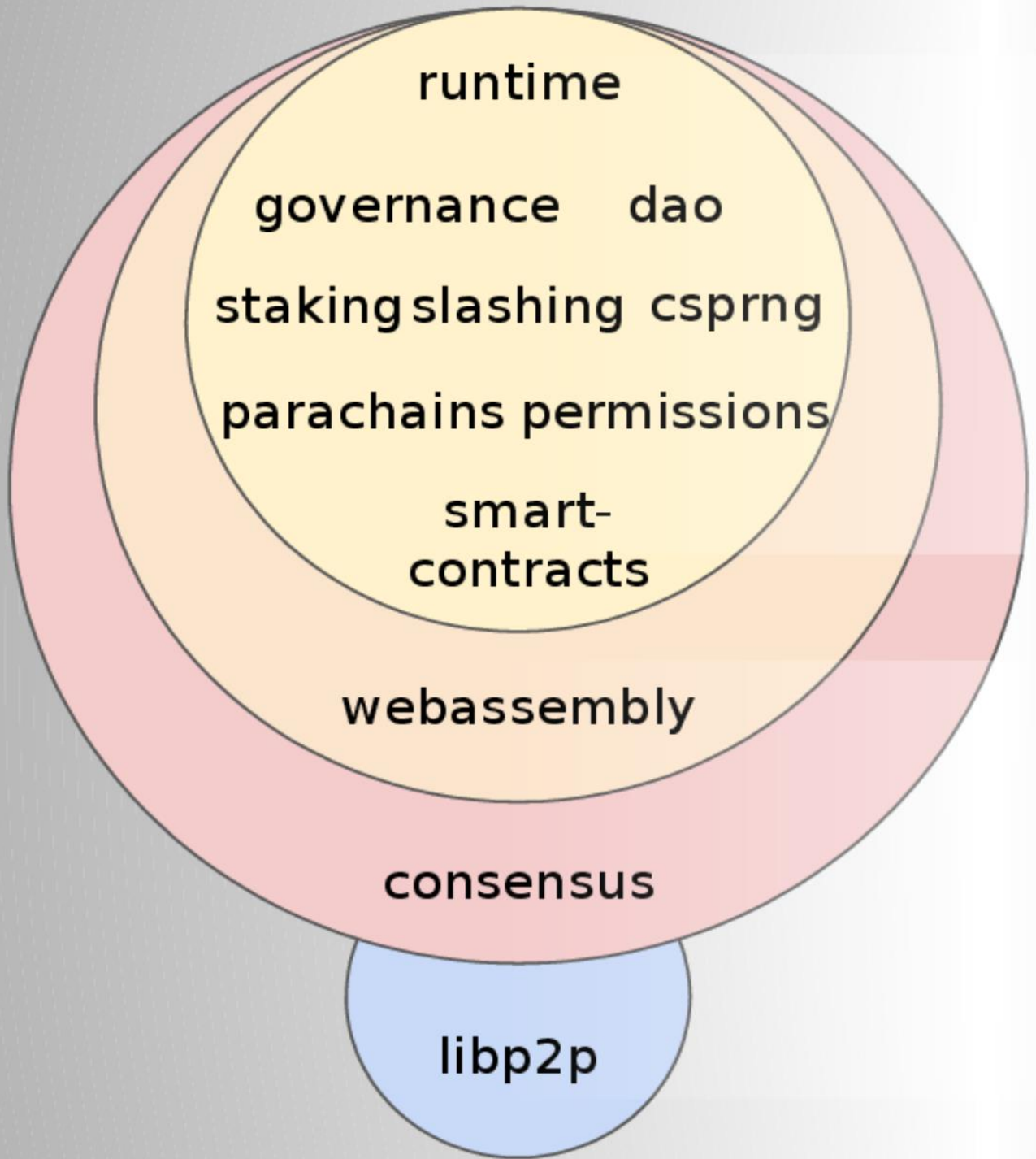
cosmos.network

Comparison

	COSMOS	POLKADOT
Consensus	Tendermint (BFT)	GRANDPA/BABE
Governance	Validator/Delegator Vote	Referendum and Council representing stakeholders
Models	Hub and Zones	Relay chain and parachains
Security	Each zone has its own security	Pooled security
Native token	Atom	Dot

Sovereignty*

*Polkadot substrate can be used as library in Cosmos base Chain



Written in Rust

Built into both Wasm & native

Wasm stored on-chain

Written in Rust

Built into native

Contact Us

Shakil Muhammad

Chairman

+82-10-3347-7860

shakil@rnssol.com

shakilphd@kaist.ac.kr

www.rnssol.com

Shehroze Rao

Cheif Executive Officer

+92-323-5222-321

shehroze@rnssol.com

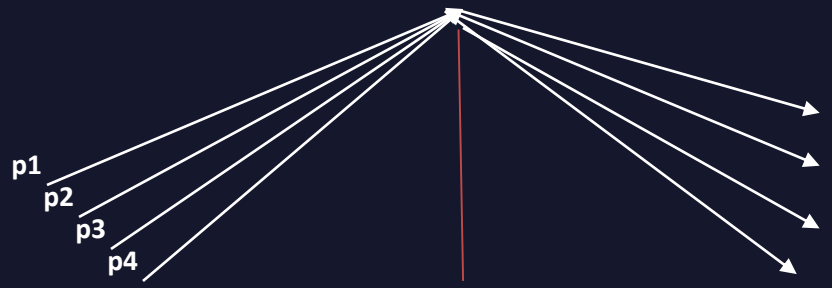
contact@rnssol.com

PAKISTAN OFFICE

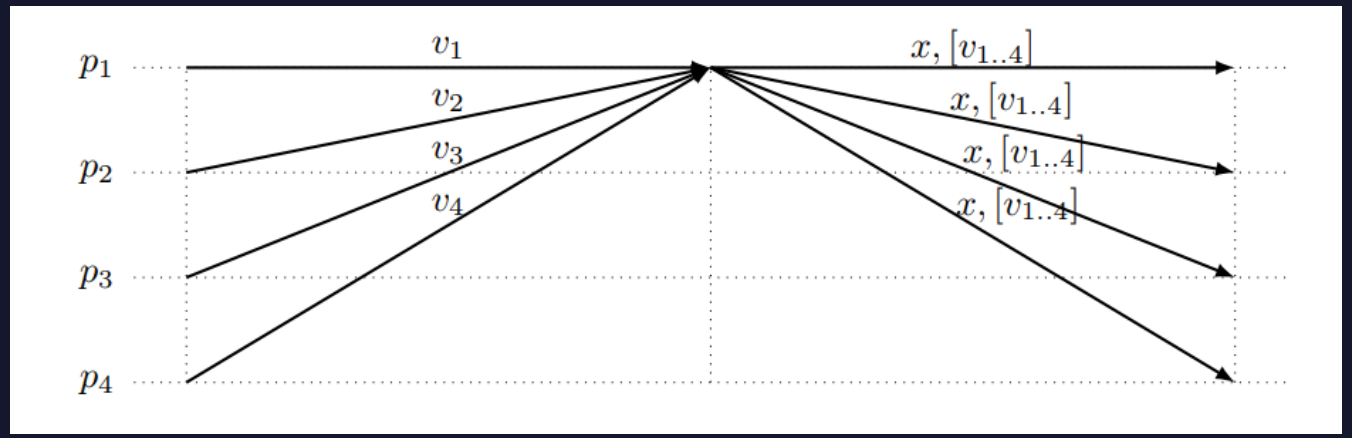
Office #7, 3rd floor, Satellite plaza, 6th road Rawalpindi, Pakistan

SINGAPORE OFFICE

531 A Upper Cross Street #04-95, Hong Lim Complex, Singapore



Global Stabilization Time, GST



t ≥ GST

Validity Predicate-based Byzantine consensus