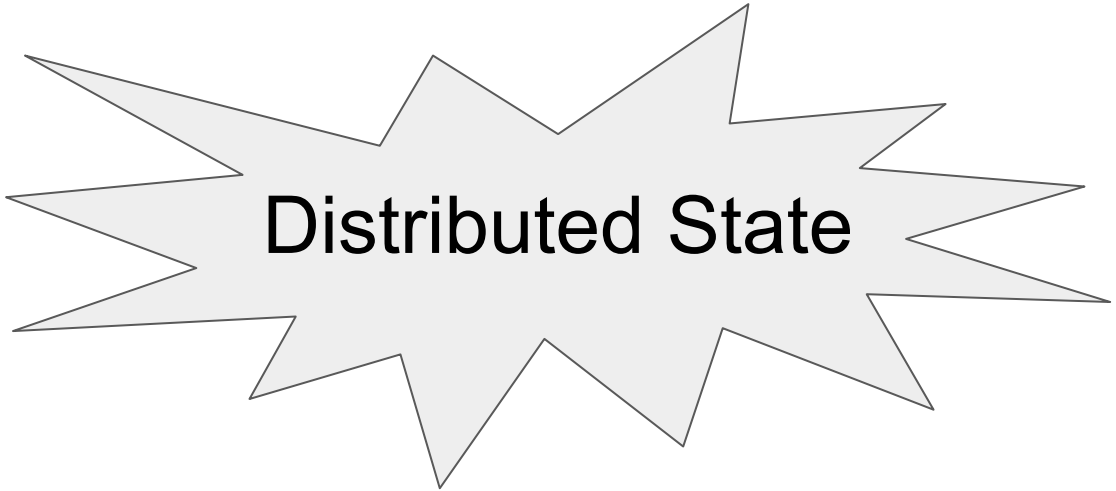
A Brief History of Distributed State

Ethan Buchman
Cosmos & Tendermint
Berlin

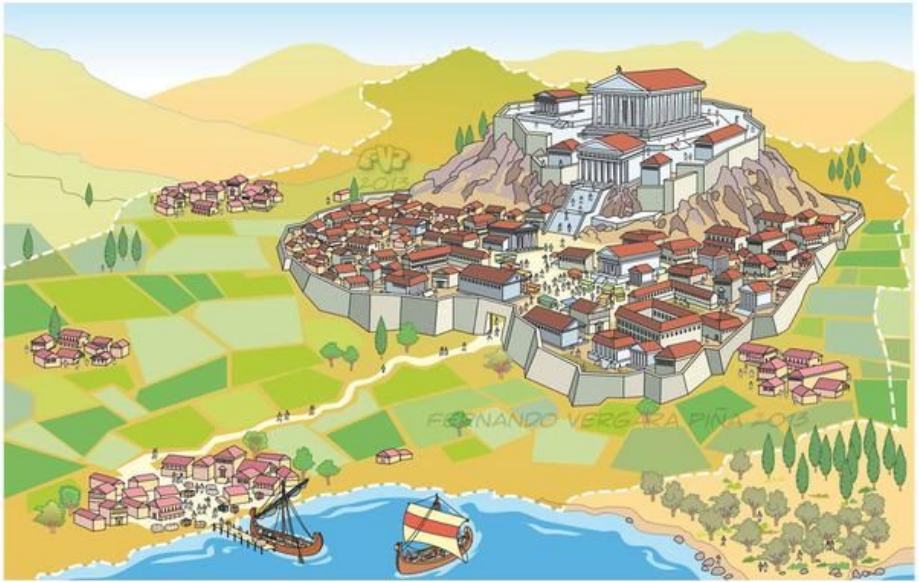
COSMOS
INTERNET OF BLOCKCHAINS

The Problem

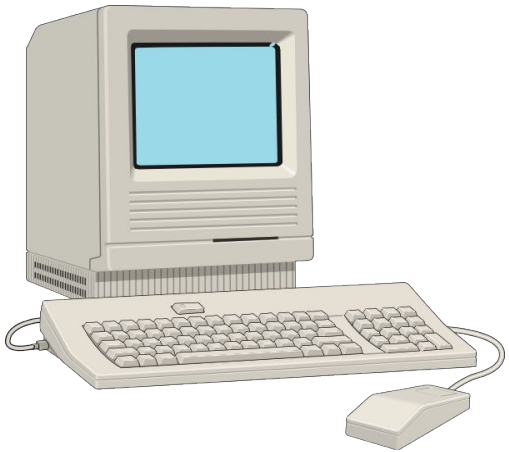


Villages and City States

Villages and City States



My Computer and Intranet



Empires

Somebody Else's Government

Somebody Else's Computers

amazon web services™

Sovereign Nations

GeoPolitical Sovereignty

Digital Network Sovereignty

Villages and City States

EcoVillage and Metropolis

Network of Community Currencies

Blockchain Applications

Somebody Else's State Machine

- Zookeeper, etcd, consul
  - Fancy key-value store
  - Emphasis on distributed systems tasks (dynamic config, locking, etc.)

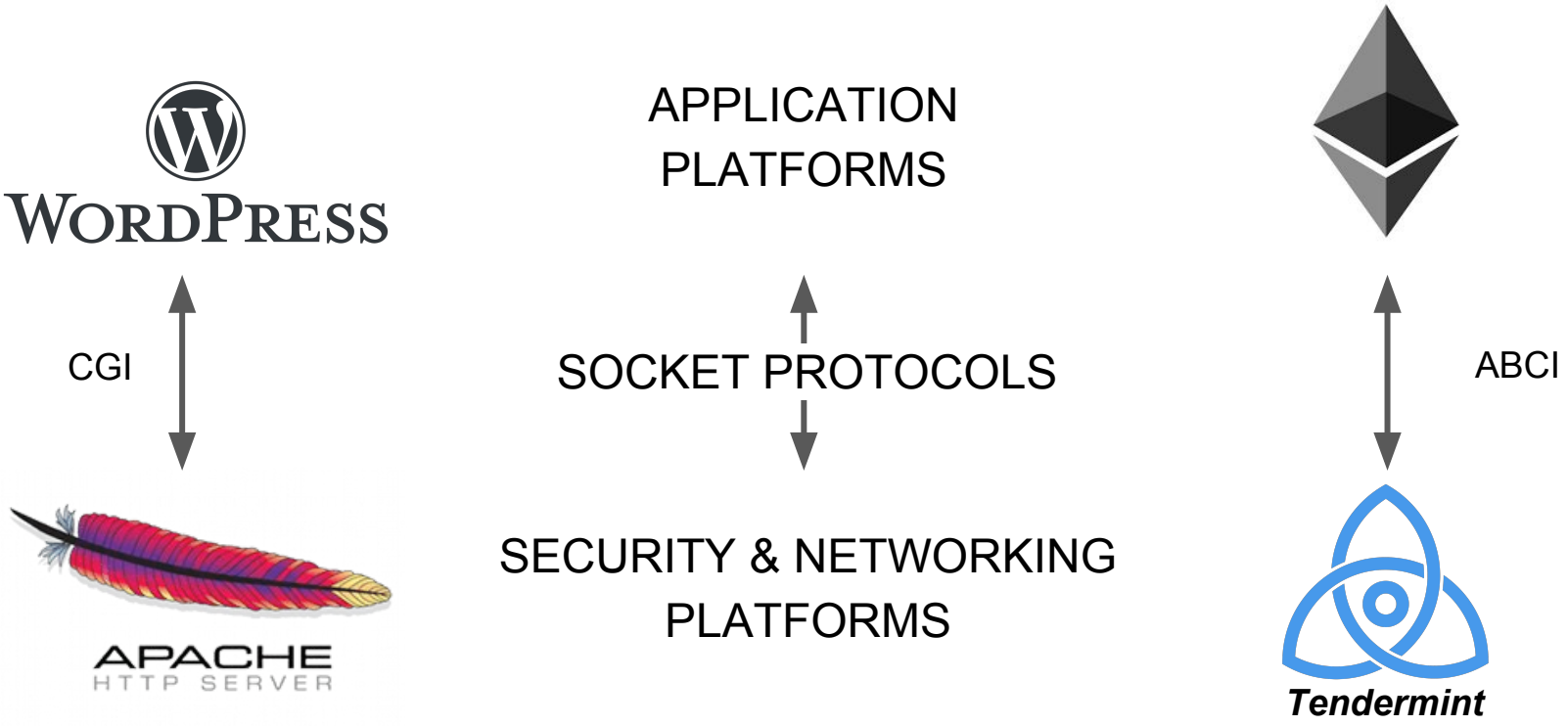
- Bitcoin
  - "Programmable money"
  - "Functional programming" - no state (!), contracts renewed every transaction
  - Forth like, purposefully not Turing-complete

- Ethereum
  - "Smart contracts"
  - "Contract-oriented" - stateful contracts live independently on the blockchain
  - Turing complete (Ethereum Virtual Machine)

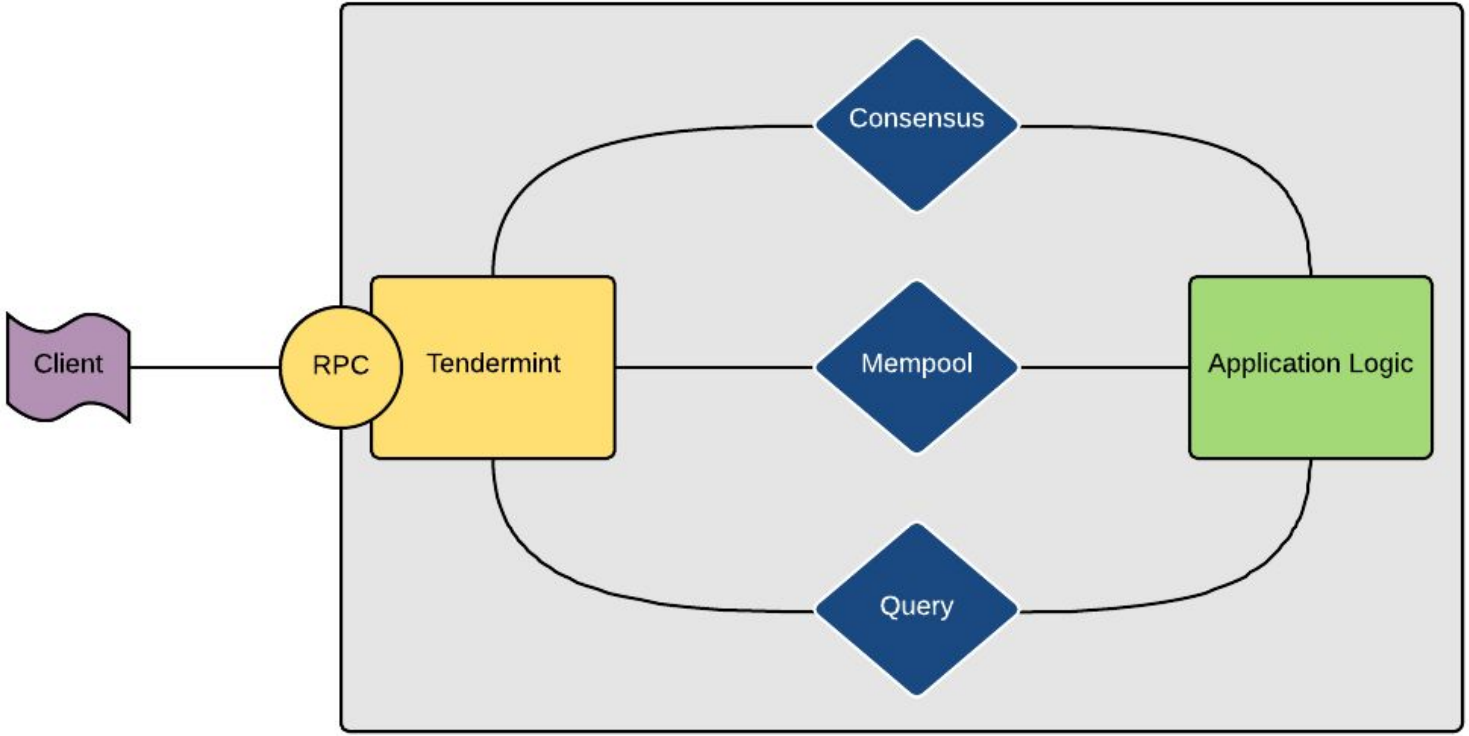
Application Blockchain Interface (ABCI)

State Machines
in
Any Programming Language

Application Blockchain Interface (ABCI)



Application Blockchain Interface (ABCI)



ABCI Ecosystem

node JS™
LotionJS

Java

C++

python™

elixir

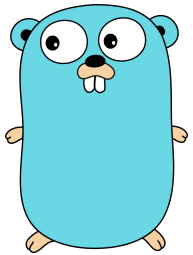
Haskell

ERLANG

Cosmos SDK

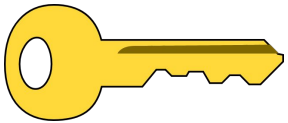
Cosmos SDK

- Abstract away low-level ABCI concerns



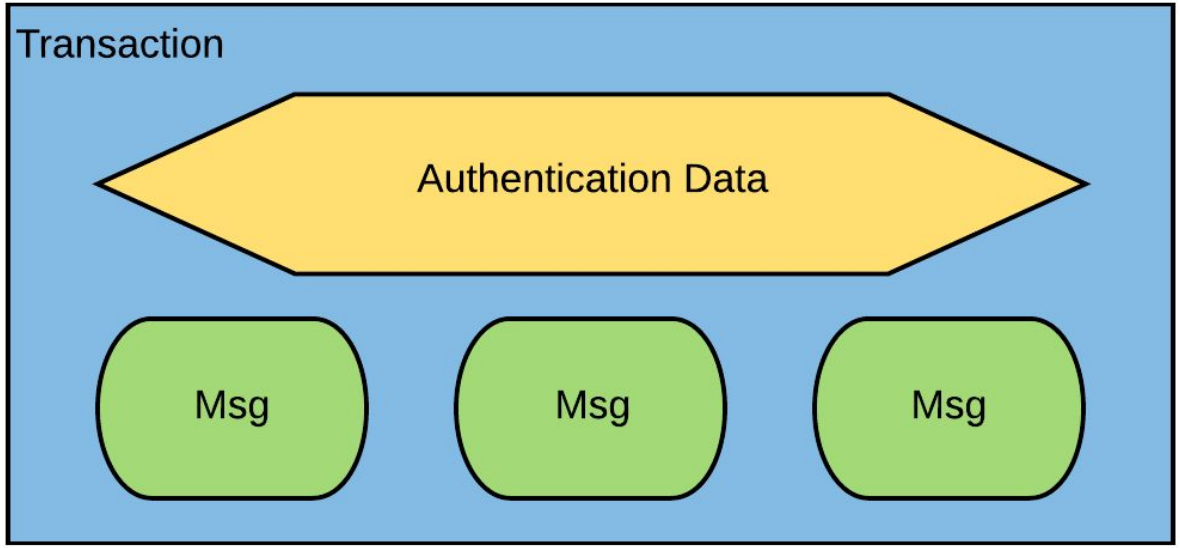
- Golang
  - Tiny language, static and interface types, high performance, compiles everywhere, standardized formatting, built-in testing, etc.

- Composable modules

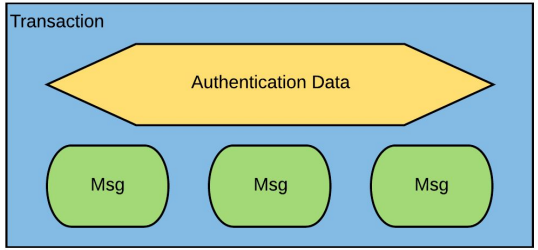


- Capability-based security

Txs and Msgs

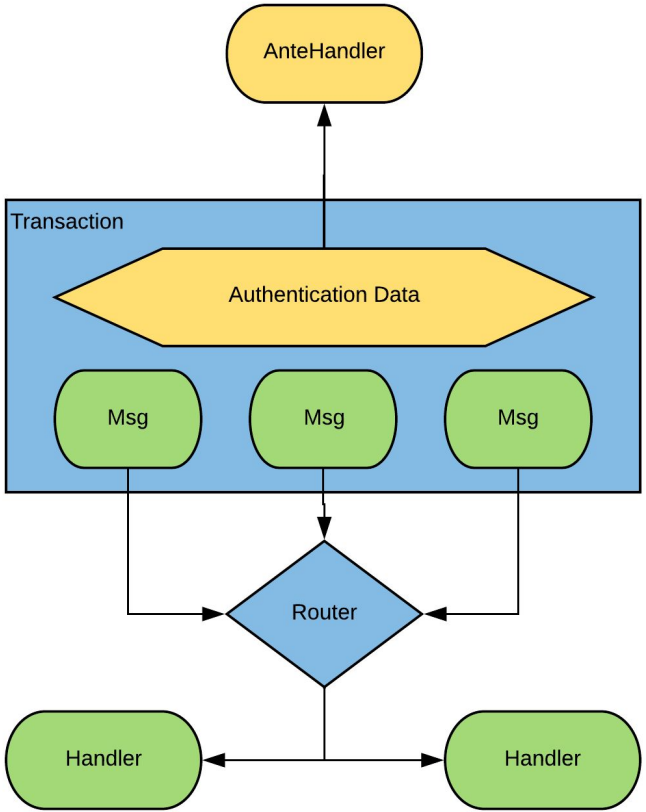


Txs and Msgs

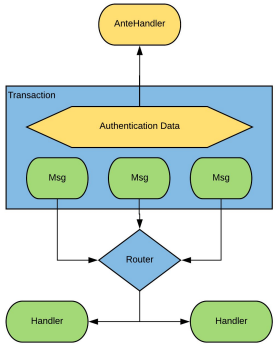



```
// A standard way to wrap Msgs
// with a Fee and Signatures.
type StdTx struct {
    Msgs       []sdk.Msg       `json:"msg"`
    Fee        StdFee          `json:"fee"`
    Signatures []StdSignature `json:"signatures"`
    Memo       string          `json:"memo"`
}
```

Handlers and AnteHandler

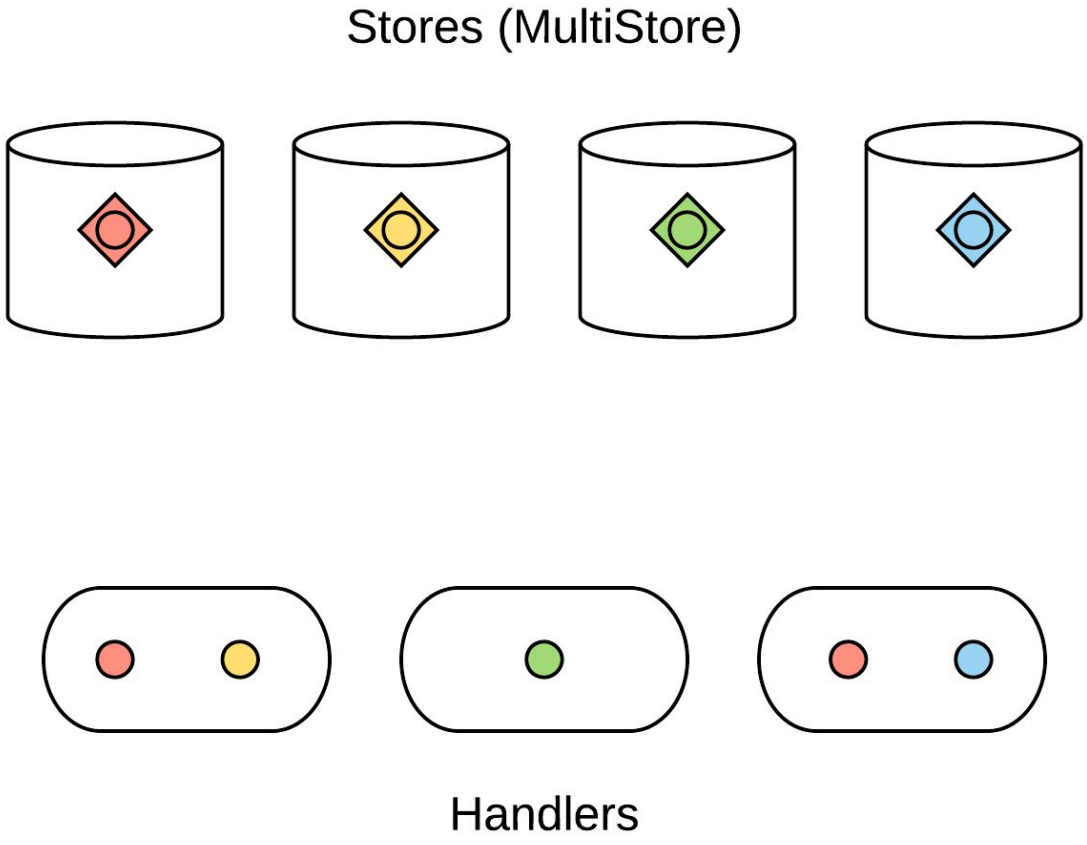


Handlers, AnteHandler, Result




```go
// Handler defines the core of the application's
// state transition function
type Handler func(ctx Context, msg Msg) Result

// AnteHandler authenticates transactions before
// their internal messages are handled.
type AnteHandler func(ctx Context, tx Tx) (newCtx Context, result Result, abort bool)

// Result is the result of a transaction
type Result struct {
    Code ABCICodeType
    Data []byte
    Log  string

    GasWanted int64
    GasUsed   int64
    Fee       sdk.Coins

    Tags Tags
}
```

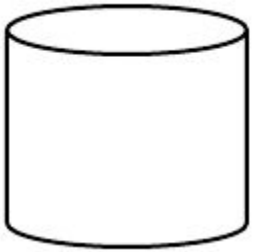
Capability-Based Store Access



Capability-Based Store Access

```
// Create keys, register handlers, mount stores
keyAccount := sdk.NewKVStoreKey("acc")
keyIssue := sdk.NewKVStoreKey("issue")
app.Router().
    AddRoute("send", handleMsgSend(keyAccount)).
    AddRoute("issue", handleMsgIssue(keyAccount, keyIssue))
app.MountStoresIAVL(keyAccount, keyIssue)
```

Simple Key-Value Store



```
// KVStore is a key-value store to get/set data.
type KVStore interface {
    Get(key []byte) []byte
    Has(key []byte) bool
    Set(key, value []byte)
    Delete(key []byte)
    Prefix(prefix []byte) KVStore
    Iterator(start, end []byte) Iterator
    ReverseIterator(start, end []byte) Iterator
}
```

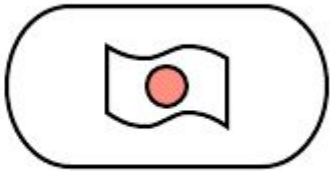
Simple Key-Value Store

```
// Load the store, fetch an account.
store := ctx.KVStore(key)
accBytes := store.Get(from)
```

Mappers and Keepers

Store

Mapper or Keeper



Handler

Mappers and Keepers

```
type accountMapper struct {
    key sdk.KVStoreKey
}

type AccountMapper interface {
    GetAccount(sdk.Context, sdk.Address) Account
    SetAccount(sdk.Context, sdk.Address, Account)
}

type coinKeepr struct {
    mapper AccountMapper
}

type CoinKeeper interface {
    GetCoins(sdk.Context, sdk.Address) sdk.Coins
    SetCoins(sdk.Context, sdk.Address, sdk.Coins)
}
```

BeginBlock & EndBlock

```
type RequestBeginBlock struct {
    Hash                []byte
    Header              Header
    Validators          []SigningValidator
    ByzantineValidators []Evidence
}

type ResponseEndBlock struct {
    ValidatorUpdates      []Validator
    ConsensusParamUpdates *ConsensusParams
    Tags                  []common.KVPair
}
```

Some Modules

Cosmos Hub:

- Delegated Proof-of-Stake
- Slashing
- Governance
- Coin transfers and issuance
- Fee distribution
- IBC
- Authentication

https://github.com/cosmos/cosmos-sdk

Ethermint:

- Same as Hub
- EVM

https://github.com/cosmos/ethermint

And much more!

Learn more

tendermint.com

cosmos.network

We’re hiring!