<CRYPTO /DEVS>

MODULO 3

ETHEREUM

DAPPS

SMARTCONTRACTS

RSK

A·B ONG BITCOIN ARGENTINA

SIGNATVRA

BLOCKCHAIN ACADEMY

ATIX LABS

¿QUIENES SOMOS?

Construimos Productos Digitales

Basados en Blockchain

AtixLabs Co-Founder

ETC Scala Developer

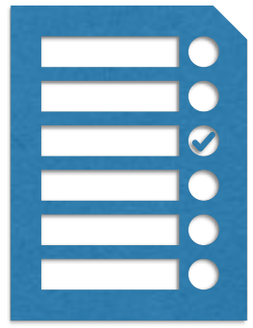
@AlanVerbner

github.com/AlanVerbner

CONTRATO TRADICIONAL

Es el convenio o pacto, ya sea oral o escrito, entre partes que aceptan ciertas obligaciones y derechos sobre una materia determinada.

Componentes fundamentales:



Los datos relativos a los sujetos que lo suscriben

Los pilares de la prestación y contraprestación que se establece

La forma en la que se da el visto bueno a aquel por parte de las dos partes implicadas.

“A smart contract is a set of promises, specified in digital form, including protocols within which the parties perform on the other promises”[1]

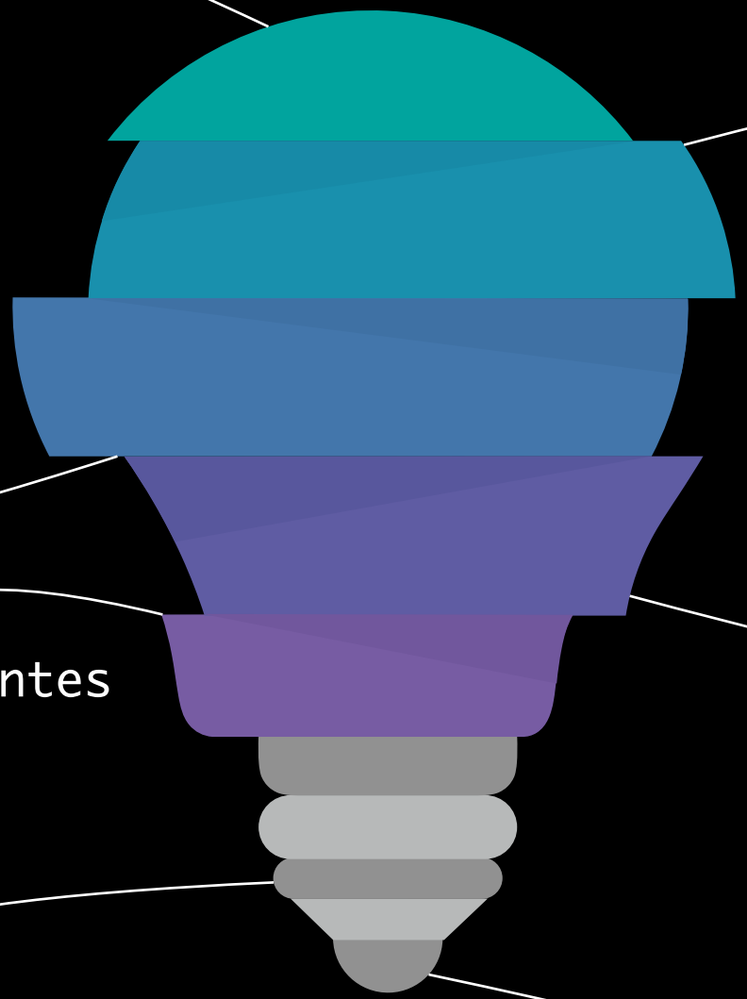
[1]: Smart Contracts: Building Blocks for Digital Markets - 1996 by Nick Szabo

Confiables

Autónomos

Digitalmente vinculado
y verificable

Ejecución automática

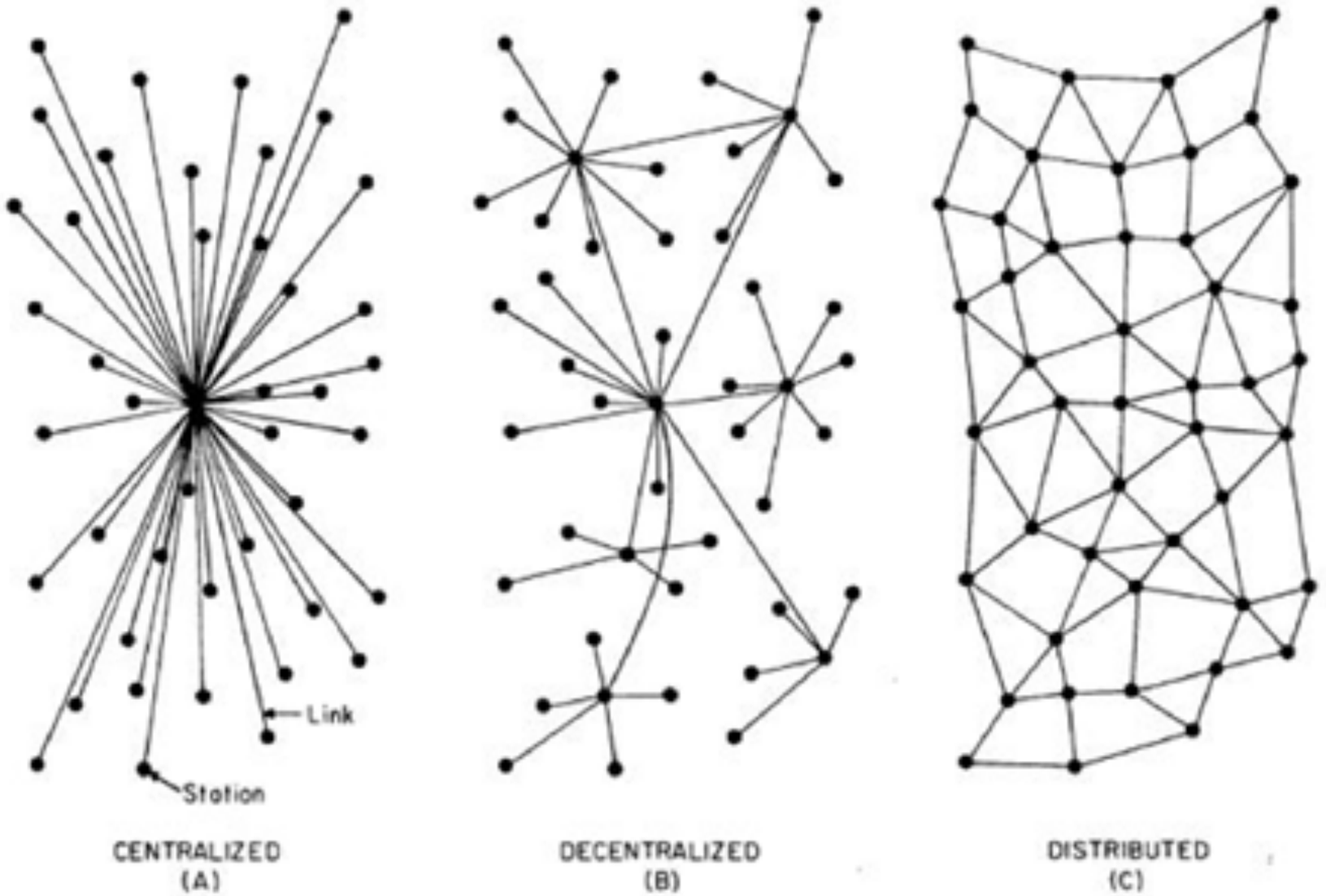


Autosuficientes

La ejecución depende de código,
no hay ambigüedades

ĐAPPS

The General Theory of Decentralized Applications, Dapps - David Johnston



CENTRALIZED
(A)

DECENTRALIZED
(B)

DISTRIBUTED
(C)

ĐApps: Definición

1. Ser completamente open source
2. Los datos y registros de operación tienen que ser criptográficamente almacenados en una base de datos pública y descentralizada (blockchain)
3. Usar un token criptográfico
4. Los tokens sean generados de acuerdo a algún estándar

[1]: https://github.com/DavidJohnstonCEO/DecentralizedApplications

ĐApps: Clasificación

- **Tipo I**: aplicaciones descentralizadas que tienen su propia blockchain. Ej: Bitcoin, Litecoin, etcs

- **Tipo II**: aplicaciones descentralizadas que usan una blockchain de Tipo I. Son protocolos y tienen tokens que son necesarios para su funcionamiento. Ej: Omni Protocol

- **Tipo III**: aplicaciones descentralizadas que usan protocolos de Tipo II. Tienen tokens que son necesarios para su funcionamiento. Ej: SAFE Network usa Omni para generar “safecoins”

ĐApps: Estado Actual

- Análisis de Fred Ehrsam

	Web 2.0	Web 3.0 (dApps)	Status
Scalable computation	Amazon EC2	Ethereum, Truebit	In progress
File storage	Amazon S3	IPFS/Filecoin, Storj	In progress
External data	3rd party APIs	Oracles (Augur)	In progress
Monetization	Ads, selling goods	Token model	Ready
Payments	Credit Cards, Paypal	Ethereum, Bitcoin, state channels, 0x	Ready

Recap

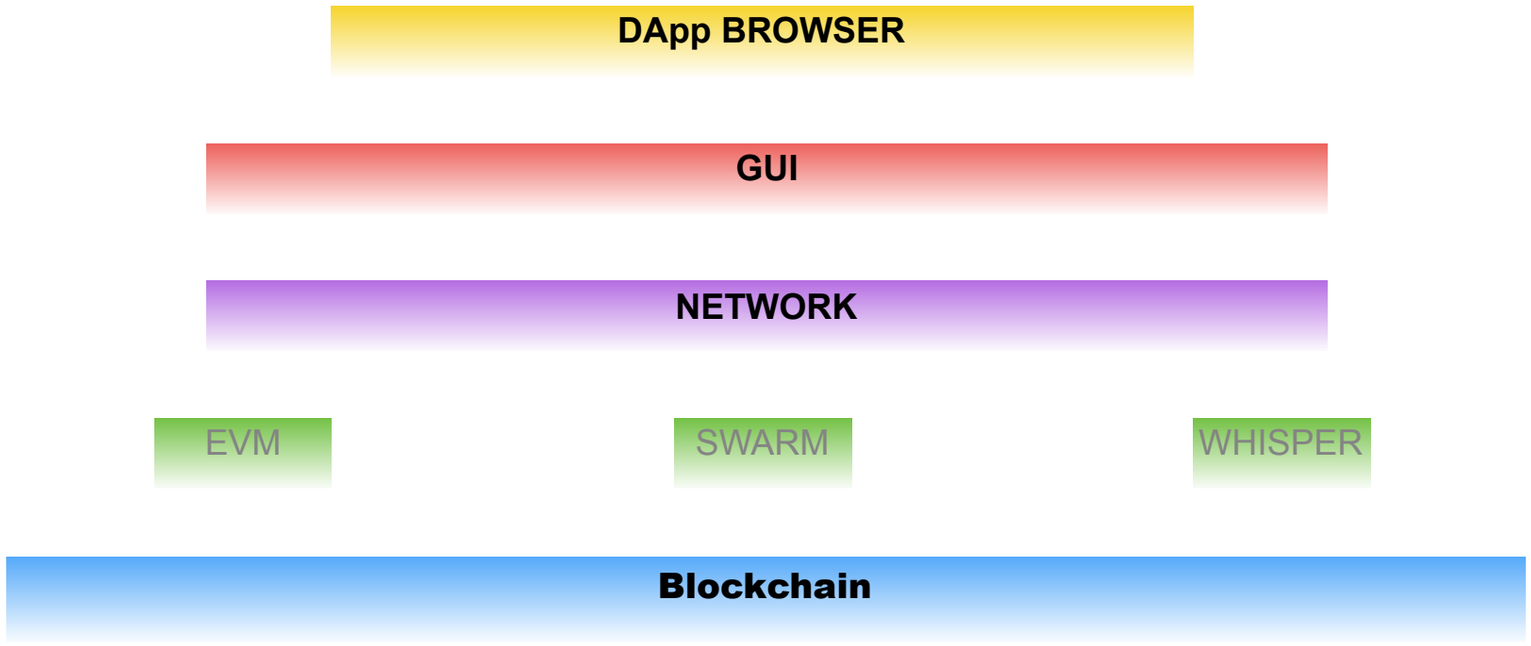
- Smart Contracts
- DApp

Preguntas

1. Pensemos 2 ejemplos de Smart Contracts
2. ¿Cuáles son los beneficios de las DApps con respecto a las Apps tradicionales?

Ethereum

- Plataforma descentralizada que permite la ejecución de Smart Contracts
- Basada en Blockchain
- Contiene una máquina virtual Touring-complete (EVM)
- Token propio, Ether

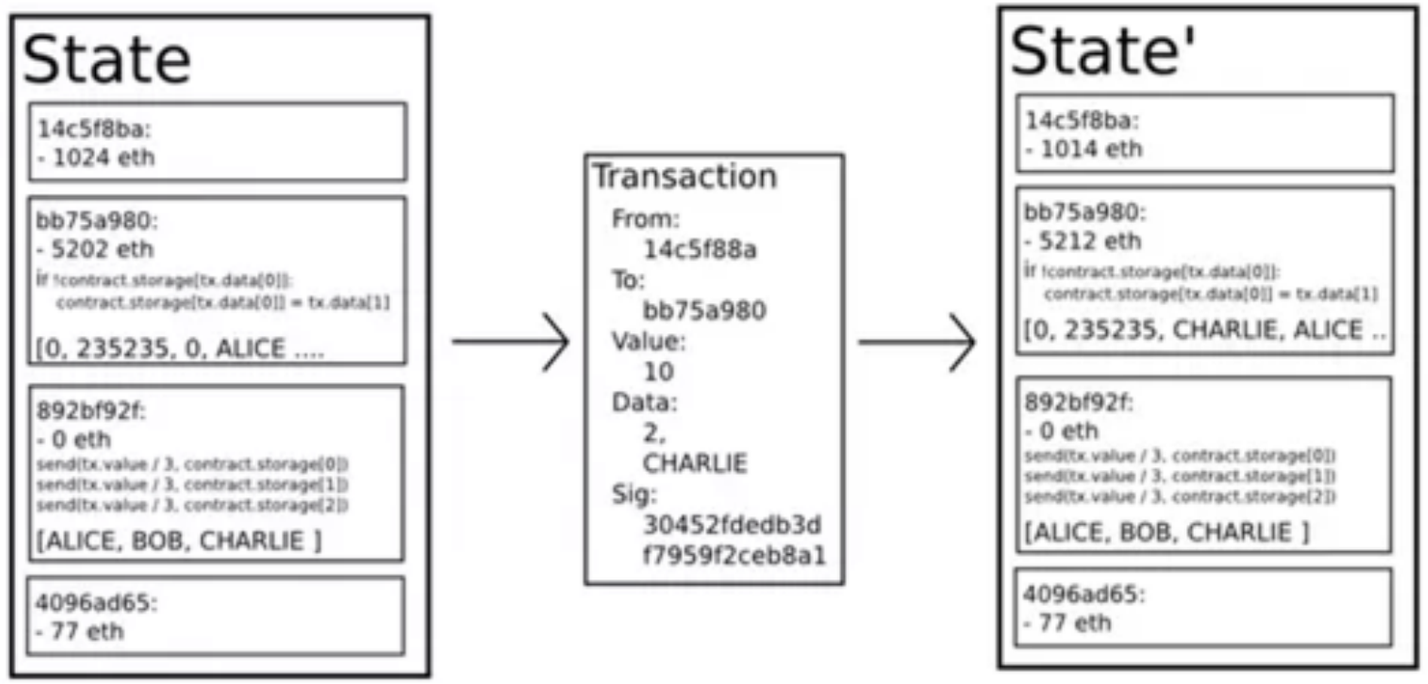


Un poco de historia

- Propuesta a fines de 2013 por Vitalik Buterin
- Financiada por una crowdsale
- “Puesta en producción” 30 de Julio del 2015
- Formalizacion: Yellow Paper por Gavin Wood
- DAO fork 20 de Julio de 2016

Máquina de estado basada en transacciones

- $\sigma_{t+1} \equiv Y(\sigma_t, T)$
- Deterministico
- El estado



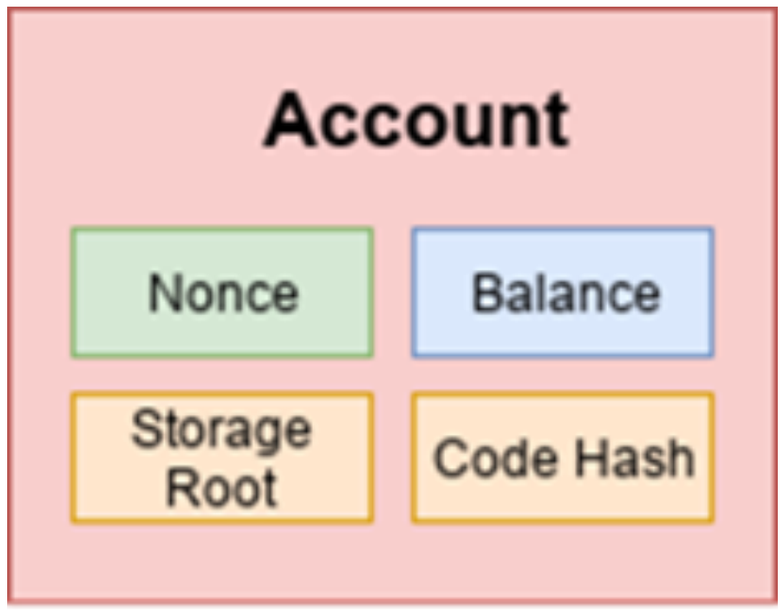
Bloques, transacciones y denominaciones

- Transacciones generan cambios de estado
- Las transacciones son agrupadas en bloques
- Los mineros generan los bloques (por ahora, con POW)
- Bloque especial: Génesis

Unit	Wei Value	Wei
wei	1 wei	1
Kwei (babbage)	1e3 wei	1,000
Mwei (lovelace)	1e6 wei	1,000,000
Gwei (shannon)	1e9 wei	1,000,000,000
microether (szabo)	1e12 wei	1,000,000,000,000
milliether (finney)	1e15 wei	1,000,000,000,000,000
ether	1e18 wei	1,000,000,000,000,000,000

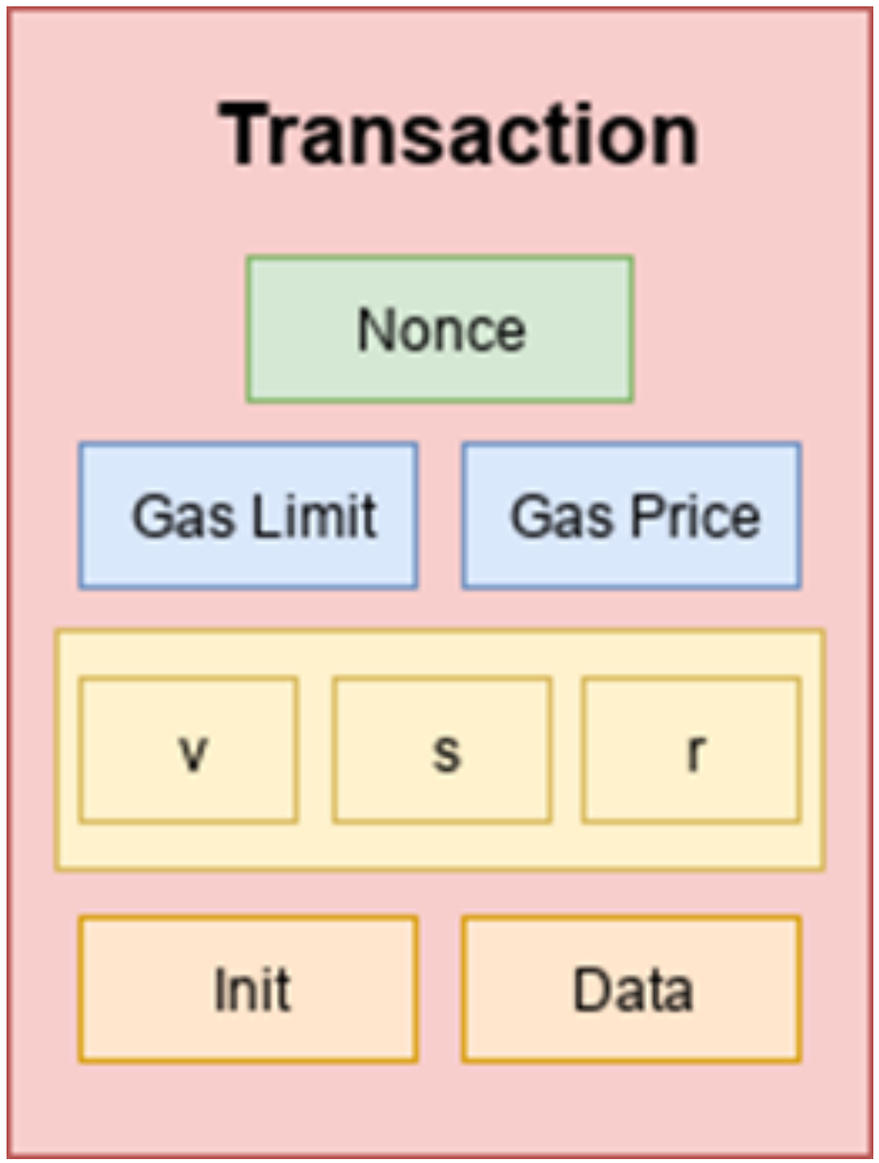
Accounts

- Conforman el estado global
- Dos tipos:
  - Non-Contract accounts (Manejadas por claves privadas)
  - Contract accounts (contienen el codigo de los smart contracts)
- Identificadas por una dirección 20 Bytes



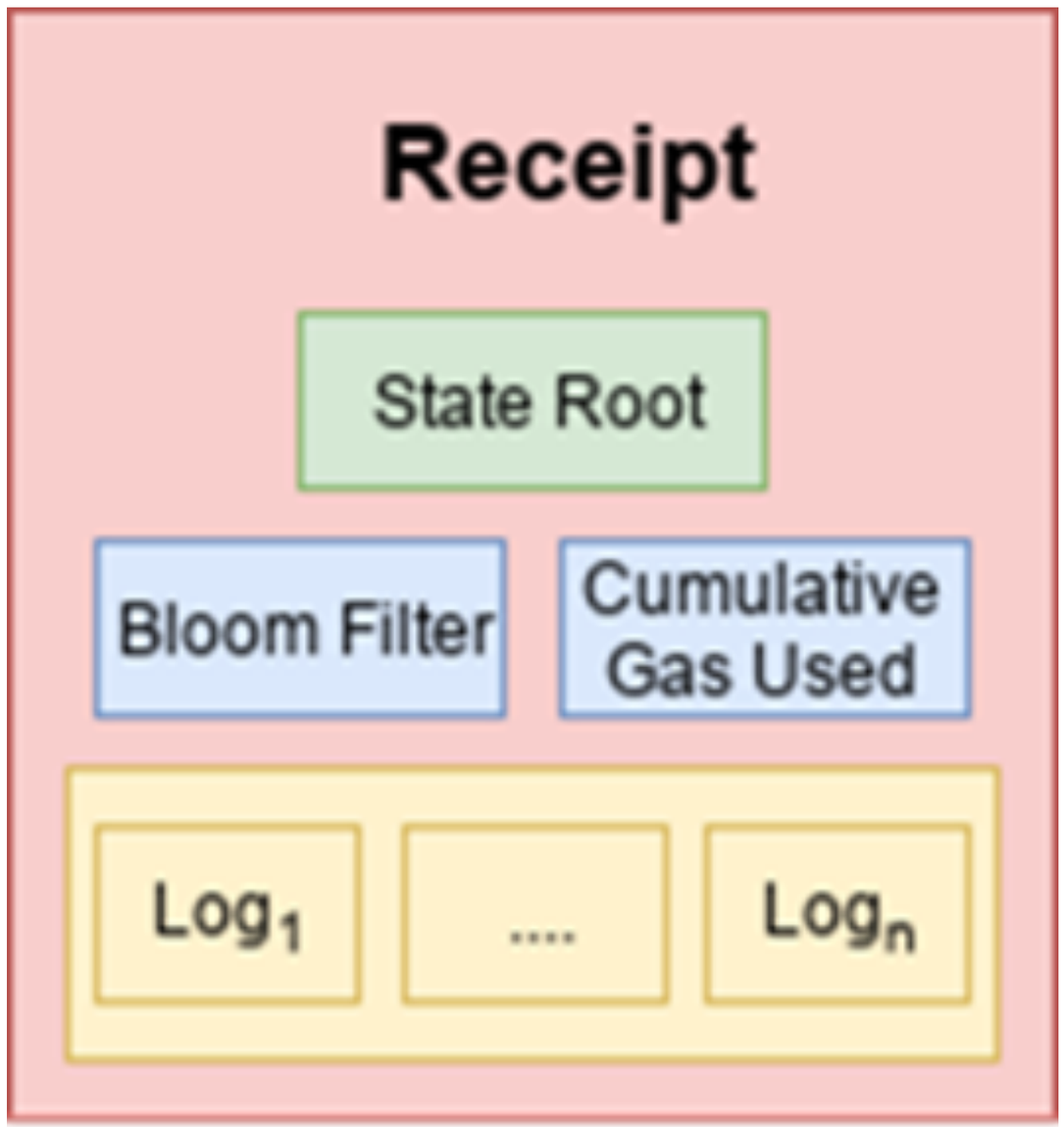
Transactions

- Instrucción firmada criptográficamente construida por un actor externo
- Tipos:
  - Contract Init
  - Message calls



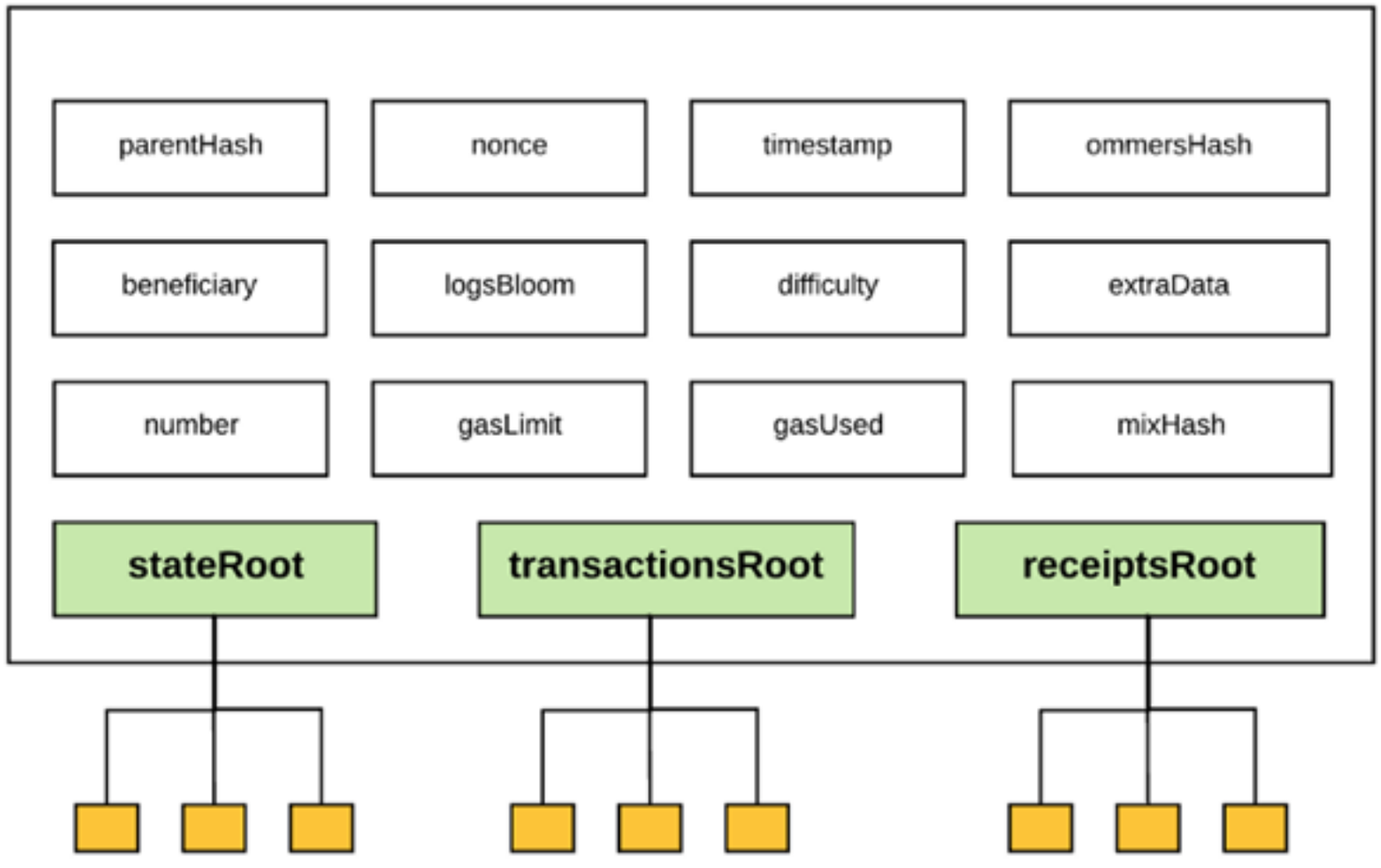
Receipts

- Contienen información sobre la ejecución de una transacción



Blocks

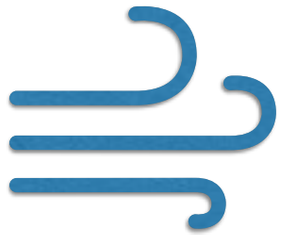
- BlockHeader
- Block Body
  - Transactions
  - Ommers



Block Body

- Lista de transacciones
- Lista de ommers

Gas and Payment



Costo Max = GasPrice * GasLimit

- Gas Limit: cantidad máxima de gas a consumir
- Gas Price: precio que estamos dispuestos a pagar por unidad de gas

Recap

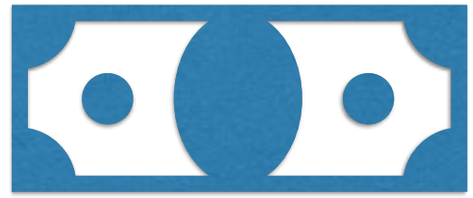
- Introducción a Ethereum
- Accounts, Blocks, Transactions, Receipts, Gas

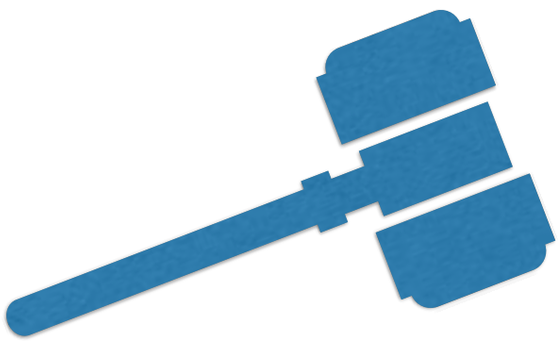
Preguntas

- Entre todos los conceptos hay uno que habla refiere al tamaño de los bloques, ¿cuál es?
- ¿Qué puede suceder si elijo poner gas price = 0?
- Si el código de los contratos es inmutable, ¿cómo se les ocurre que se puede resolver un bug?

Ciclo de ejecución

1. Validación del bloque
2. Ejecución de las transacciones
    a. Validación de cada transacción
    b. Ejecución en la EVM
    c. Generación de Receipts
3. Pago del reward
4. Validación post ejecución





Validación de bloques

1. Block Header
    a. Extra data: tamaño máximo y DAO fork
    b. Timestamp: mayor al del bloque padre
    c. Dificultad: incremento en relación al bloque padre y a la time bomb (en ethereum)
    d. Gas Usado: tiene que ser menor que el gas limit
    e. Gas Limit: incremento en relación al padre y valor máximo
    f. Numero: parentBlockNumber + 1
    g. POW
2. Block Header y Body
    a. Transactions root: MPT de las Tx sea igual al valor del header
    b. Ommers Hash: hash de los ommers sea el del header
3. Ommers correctos
    a. Cantidad: Sean la cantidad máxima sin duplicados
    b. Válidos: sean headers válidos con ancestros correctos

Validación de transacciones

- Sintáctica: esté bien armada (valores máx, etc)
- Firma válida: la firma coincida con la del sender
- Nonce: sea igual al del sender
- Gas limit: el gas limit es suficiente (intrinsic gas)
- Balance cuenta: hay suficiente para pagar el costo máximo
- Block gas limit: queda espacio para ejecutar esta tx en el bloque

Ejecución en la EVM

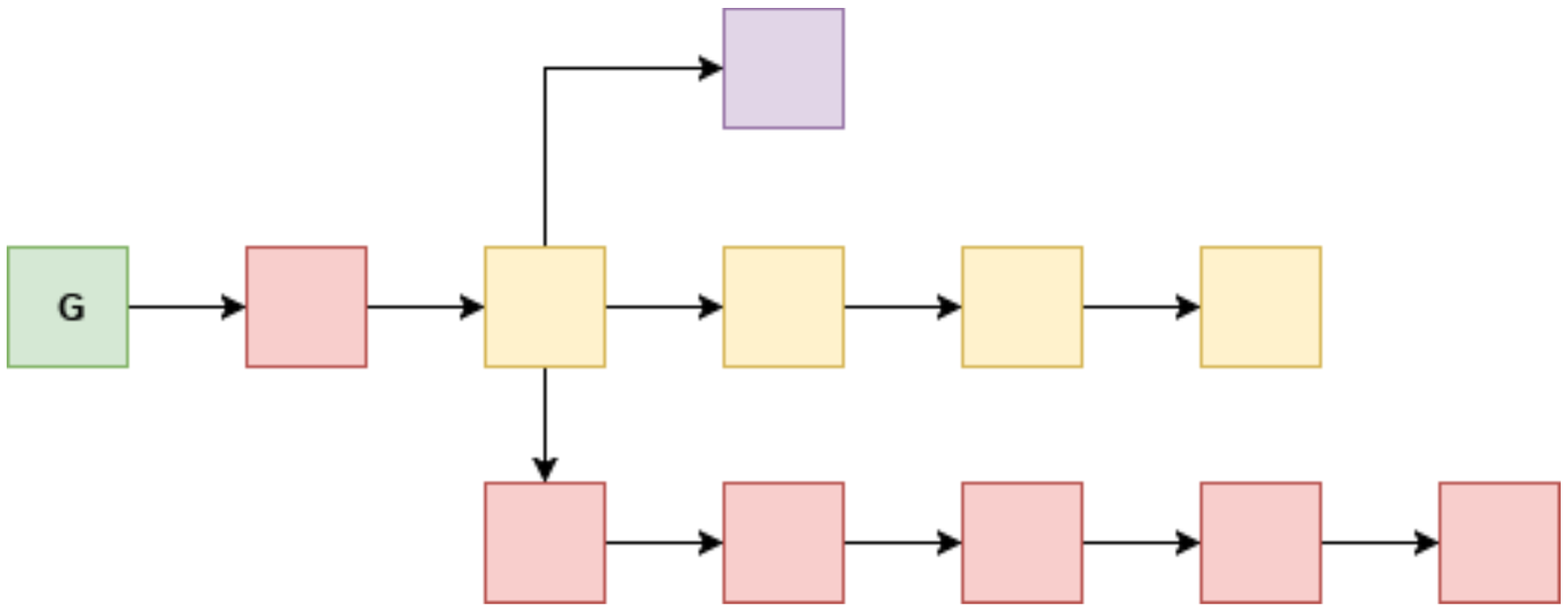
1. Se le descuenta el UpfrontCost = GasPrice * GasLimit
2. Se incrementa el nonce
3. Se transfiere el value en caso de no ser cero
4. Se ejecutan los opcodes
5. Se hace reembolso = GasPrice * RemainingGas
6. Se le paga al minero el fee de ejecución = GasPrice * GasUsed
7. Se borran cuentas (SELFDESTRUCT)

Validación post ejecución

1. Receipts y logs bloom
2. Gas Used
3. State root hash

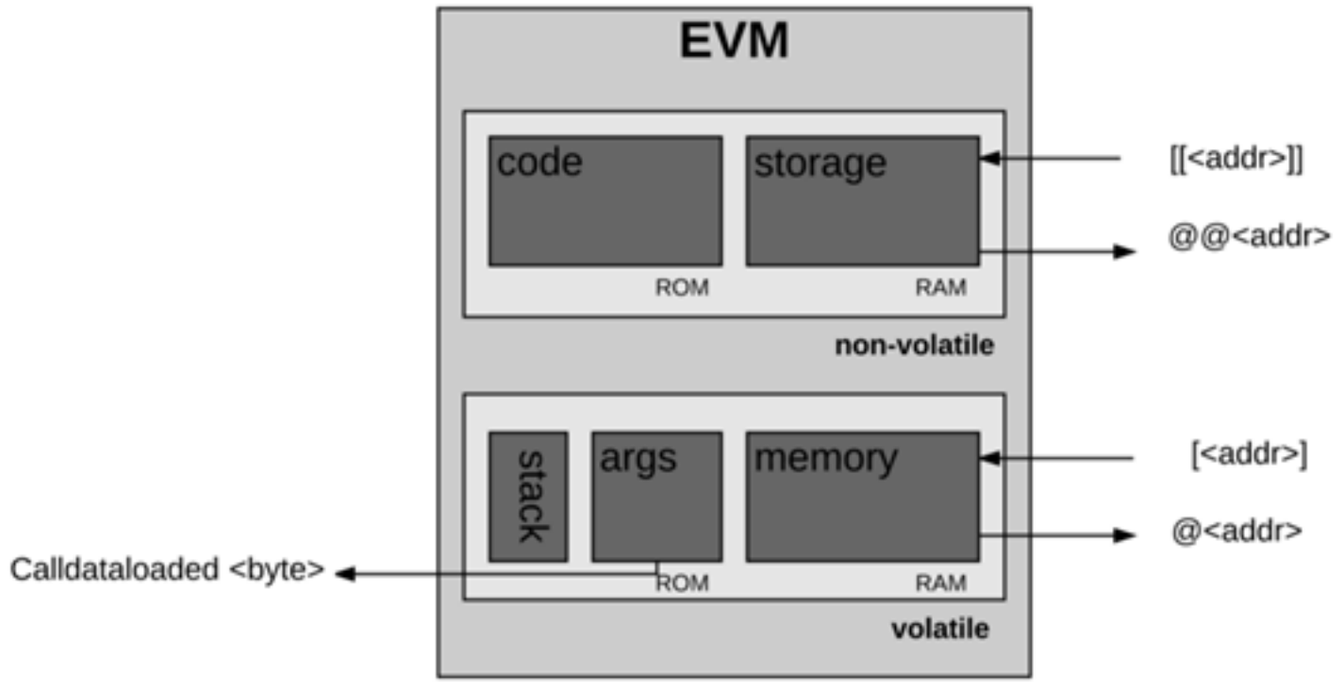
Generación de Bloques

- POW
- Pago por:
  - Haber preparado el bloque en la cadena más larga
  - Inclusión de ommers
  - Generación de ommers
- Difficulty bomb y 5M20
- 1 bloque cada 17 segundos



EVM

- Touring-complete (pero con gas limit)
- Stack Based
- UInt256 (8 Bytes)
- Memoria no permanente
- Storage persistente



Extras

- Keccak-256 (no sha3)
- RLP
- HexPrefix
- Merkle Patricia Trie
- ĐΞVp2p Wire Protocol

Para la clase que viene

- https://github.com/atixlabs/truffle_example
  - Node
  - Truffle
  - TestRPC
- https://github.com/ethereum/mist/releases
- https://www.npmjs.com/package/web3-repl