What Other Chains Can Tell Us About the Future of Ethereum

Danno Ferrin

Swirlds Labs

A trailblazer…
or
a warning to those who follow?

DALL-E prompt:
A pioneer covered wagon on a coastline in front of a shipwreck of a Spanish galleon.

How To Read This Presentation

- This column lists what other chains do
- Multiple chains may do the same thing
- I didn't survey each and every chain
  - Mostly Hedera, Near, Polkadot

- This columns list what Ethereum Mainnet can learn from the other chains
- Sometimes it supports what other EIPs are proposing
  - It may inform us on tweaks that we should make
- Sometimes things went poorly
  - We should learn from their mistake instead of repeating it.
- Sometimes it's useful, but the cost of deploying the change is too much

Accounts and Cryptography

Account Cryptography

- Ethereum, bitcoin, most Alt-EVMs
  - ECDSA(secp256k1)
  - Keccak-256
- Hedera
  - EdDSA(ed25519)/ECDSA(secp256k1)
  - SHA3-384
- Near
  - EdDSA(ed25519)
  - SHA-256
- Polkadot
  - Sr25519 (Schnorr over curve 25519)
  - Blake2b
- Ethereum Consensus Layer
  - BLS12-381
  - SHA-256

- Core support for multiple crypto algorithms is in the future for Ethereum
  - It's here if you erase the Execution/Consensus division
- Changes can be small and large
  - Key/Hash size changes (256->384)
  - Algorithm Changes (Keccak->sha3)
  - Extra Data (Public Key for Edwards/BLS)
  - Quantum Resistant Algorithm
- Any alternative or change has cascading impacts across validation and tools
  - Account Addresses
  - Contract Addresses
  - Is some indication of Algorithm in the address?

Account Address

- Hedera allows multiple keys per address
  - This can produce multiple "addresses"

- Hedera's current Alias system correlates only one address to an account
  - ECDSA keys get a full Eth Address
  - Others get "long zero" Hedera IDs

- Hedera is considering moving to a multiple address per account model

- Ethereum addresses are tied to public keys

- Eth addresses are tied to one Algorithm set
  - keccak-256(ECDSA(secp256k1) PK)

- No public key to address mappings for
  - EdDSA
  - Sr25519
  - BLS12-381

- New address mapping rules mean tooling will not be able to assume Yellow Paper address mappings

Account Address and Authorization

- Hedera uses sequential account addresses
  - May have one or more keys
  - Keys to change over time
- NEAR uses user selected usernames
  - Accounts have sub-account
- Public Keys and AccountIDs (aka Externally Owned Addresses) are not strongly correlated in some chains.

- `ECRECOVER` operates on public key extraction
- Many contracts use `ECRECOVER` for account authorization
  - Ties key to address forever
  - Doesn't Work with Account Abstraction
- A new `ACCOUNT_AUTHORIZED` precompile may be needed
  - Account Address
  - Account Args (Public key, key index, threshold, algorithm selection, etc)
  - Message Hash

Account Creation and Account Abstraction

- Hedera's account system initially required some other party to create your account
  - This was a separate transaction
  - Creator needed your initial keyset

- For ease of UX for Wallets and CEXes Hedera created an account "alias" that you can send funds to a public key
  - Account creation handled transparently

- Hedera is investigating "Hollow" account creation: sending to a hashed address
  - On first TX the account key will be set from transaction signature

- Account Abstraction may change UX considerations if we need a special transaction call to create accounts
  - Only a problem if it's the only way

- Migrating EOAs to an AA account mirrors the experience of a "Hollow" account: create the bare minimum then fill in the rest at a latter date

- Account Abstraction may interact with account and contract address creation. Dev tools should pay attention,

Ethereum Virtual Machine

Precompiled and System Contracts

- Hedera has a rich Token system accessed via Precompile/System Contracts
  - 0x167 and higher
- NEAR/Aurora uses system contracts to bridge between NEAR/Aurora contracts
  - Hashed addresses
- Moonbeam provides contacts to interact with Polkadot, among other features
  - 0x800 an higher
- Arbitrum ArbSys and friends
  - 0x64 and higher
- Celo uses EIP-2537 functions
  - 0xE3->0xE9

- Community could adopt better verbiage
  - **Precompiled Contract** - A contract that can be done in pure EVM opcodes.
    Example: cryptography

  - **System Contract** - A contract that crosses system boundaries.
    Example: account access

- We should consider establishing a precompile registry like ChainList
  - Prevent Collisions
  - Encourage documentation
  - Encourage re-use

EVM and Gas Schedule

- Hedera has fairly standard mappings to Block Data Opcodes
  - `DIFFICULTY` will be re-mapped to a prior running hash opcode.

- Hedera's native fee system is very complex. Mapping solution is to make sure Eth gas always costs more.

- Gas only billing causes problems when paying for large costs that fill the gas limit
  - Hedera solution is to allow expensive system calls to accept HBAR/Eth in addition to gas

- Consider System Contracts instead of opcodes when adding new block data access
  - Preserves opcode space for VM operations
  - This may require a cheaper way to do precompile calls.

- Multidimensional-1559 is simpler than what Hedera has. It's not (too) scary.

- Consider alternative payment channels if large gas fees become required.
  - Large contract deploys shouldn't fill most of a block

AUTH and AUTHCALL

- No chains I am aware of do anything similar

- This is evidence Ethereum shouldn’t do it.

(This is an unfair take, but I strongly feel a balanced analysis will still land on “don’t do this”)

The Substance only matters at the Surface

Emulating ERC-20 and ERC-721 Tokens

- Hedera’s native tokens are exposed as contracts at their address with “redirect” contracts to System Contracts
- Allows interoperability with native tokens and off the shelf DeFi contracts as though they were EVM tokens
- Most other non-evm L1s project or wrap tokens.

- Only the *effects* of the EVM execution are what matters to consensus.
- Clients could write native implementations of common and well used contracts for blazing fast evaluation
    - OpenZeppelin, Major Stablecoins, DeFi, etc.
    - Keep the original for fallback
- The many zkEVM projects are enabled by this principle

Transactions and Transacting

Block Organization and MEV

- Hedera's hashgraph consensus fully seperates Ordering from Execution
  - Always executed in order of arrival
  - No mempool
- Hashgraph is Asynchronous BFT Consensus
  - No leader and no block producer
  - Blocks are grouped by 2 second window
- Continuous execution is how Hedera gets 15 Mgps
  - On datacenter hosted nodes with lots of storage, connectivity, and different data proofs

- Many of these decisions fall out from consensus algorithm choice
- Consensus 🐼*almost*🐼 never changes
- MEV and Proposer Builder Separation relies on memory pool/dark pools
- Finding ways to use beacon slot processing "dead time" can increase gas potential
  - Hi Péter, we all know the real problem is data growth, not compute.

Transaction Nonces

- FISCO-BCOS has ethereum-like transactions with non-sequential nonces

- Hedera doesn't have transaction nonces
  - Relies on "`transactionValidStart`" and "`transactionValidDuration`" to deconflict
  - Timestamp goes down to nanos

- Hedera's Ethereum Transaction compatibility added an `ethereumNonce` to Hedera account records

- Transaction expiration times might clear up mempools
  - In practice subsequent nonces are

- The transaction `nonce` should have been named `sequenceNumber`

- But it won't happen: Near, Polkadot, et. al. all use a monotonically incrementing nonce, so there is industry alignment on monotonically incrementing nonces

Meta Transactions

- Hedera gains Ethereum Transaction compatibility by wrapping Ethereum Transactions in a Hedera Transaction
- Outer transaction only pays to “bring the Ethereum transaction to consensus,”
- Outer transaction may pay gas, may not
- Wrapped transaction pays remainder of fees

- Adopting wrapped transactions provides an in-protocol mechanism for meta-transactions
- Account Abstraction has the potential to wire in arbitrary wrappings

AUTH and AUTHCALL and MetaTransactions

- No chains I am aware of do anything similar
- Hederas JSON-RPC Ethereum Compatibility allows the relayer to either pay as much of the caller’s gas as needed
- Provides one goal of Meta Transactions: Gas-Free transactions to use on-chain assets

- This is evidence Ethereum shouldn’t do it.
- Instead, enshrine Payer/Executor separation in a new transaction type

“don’t do this, do this instead”

JSON-RPC

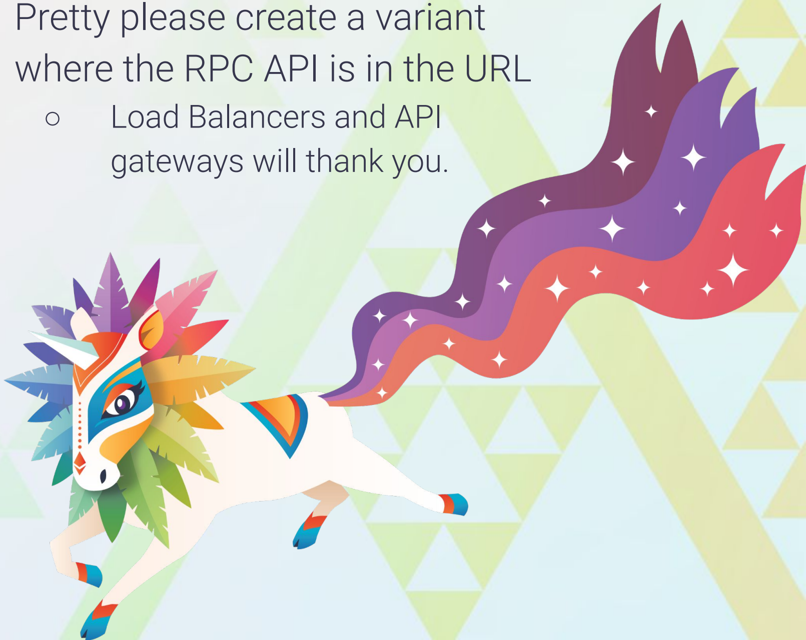
- Hedera's Full JSON-RPC compatibility is established via a relay server

- Accesses Consensus Nodes and a Mirror Node to gather data and translate into JSON-RPC
  - Most calls go to "Mirror Node"
  - Consensus node calls cost HBAR

- Raw transactions are wrapped and paid for by a relay account and sent to consensus node

- Portal Network is the Ethereum Analogue
  - Adds in decentralized data sources

- Pretty please create a variant where the RPC API is in the URL
  - Load Balancers and API gateways will thank you.



Conclusion

All Experiences are Shared Experiences

- Account Abstraction
  - Many issues can be seen in other chains, look to see what works and what doesn't
- EVM
  - Most chains can make the EVM work if they really want it bad enough
  - End user tooling is where most of the friction will be felt
- Ethereum Mainnet
  - There are some characteristics of the goals of Ethereum Mainnet that make other chain's innovations impractical for Mainnet.
  - That's OK.

Don't repeat the mistakes other chains have made

Do adapt their successes into Ethereum

Thank you!

Danno Ferrin

Principal Software Engineer, Swirlds Labs

danno.ferrin@gmail.com

@shemnon