A path to Ethereum 2.0

(one guy's current understanding)

2018-07-06

Ben Edgington

PEGASYS

Background

- Scalability workshop, Taipei, 19-21 March 2018
- Client devs' and scaling workshop, Berlin, 29 June - 1 July 2018
- Lots of activity on ethresear.ch and elsewhere
- Ethereum Core Devs calls

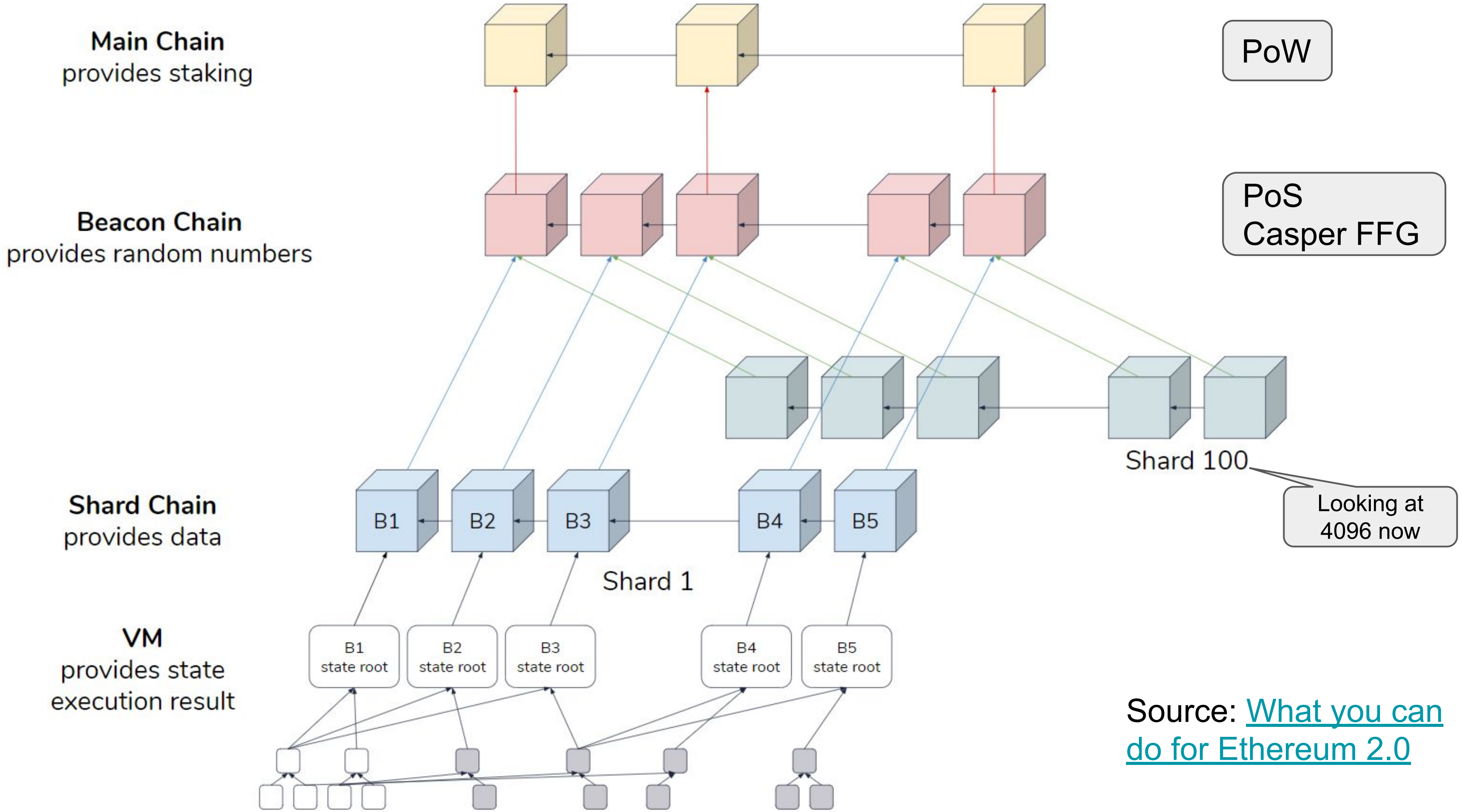
Disclaimer
None of this is set in stone; everything is subject to change. My understanding is likely to be faulty and/or incomplete. I am not party to any special knowledge: this is just an attempted synthesis of publicly available information.

Update 2018-07-08: I've added a (very) speculative delivery timeline expanding on some remarks from Justin Drake.

My nomenclature
Ethereum 1.0 is the current Mainnet, which continues as a PoW chain for the foreseeable future.
Ethereum 2.0 is features likely to be delivered "soon", possibly in sub-stages.
Ethereum 2.x is loosely features possibly to be implemented later. Maybe 3.0.

Recent developments (last 3-4 weeks)

- Rather than upgrade the current Mainnet (Ethereum 1.0), innovation will happen on a separate Ethereum 2.0 blockchain (loosely coupled to Mainnet).
    - Technical reasons:
        - Capacity of Mainnet would limit scalability of sharding.
        - EVM not well suited to managing sharding infrastructure efficiently (bit operations, etc.)
        - Recent innovations in crypto enable much more ambitious design.
    - Project reasons
        - Huge inertia to making changes on Mainnet.
        - Convergence of infrastructure for Proof of Stake and Sharding.
        - Separate chain allows radical changes and side-steps technical debt.



Source: [What you can do for Ethereum 2.0]

Differences and discontinuities 1.0 → 2.0

Changes are proposed to:

- Consensus algorithms
- Concurrency
- State transition
- Cryptoeconomics
- Cryptographic primitives
- P2P networking
- Contract programming model

(Implicit) Goals:

- Huge scalability
- Enhanced decentralisation
- Quantum security
- Protocol agility

Consensus

Ethereum 1.0	Ethereum 2.0	Ethereum 2.x
Proof of Work ● Huge environmental cost ● No finality ● Volatile block times ● Vulnerable to ASICs	Proof of Stake overlay ● Beacon and shard chains are PoS ● Periodic finality ● Casper FFG finality overlay on 1.0 chain ● Regular 5s block time	Full Casper CBC Proof of Stake.

- Separating the beacon chain from Mainnet allows for vastly more validators.
  - Individual validator stake can be reduced from 1500 Eth to 32 Eth
  - 10% Eth staked => 300K validators
  - Aiming for ~4096 shards
- PoS validator and sharding validator pools can be combined, providing higher security levels.

Concurrency

Ethereum 1.0	Ethereum 2.0	Ethereum 2.x
Globally single threaded ● Runs at speed O(c)	Sharded state and parallel Tx processing ● Runs at speed $O(c^2)$	Exponentially sharded?

- O(c) is the processing power of a typical single laptop.
- Goal is to have no requirements that would prevent consumers on commodity hardware from participating in staking/validating. Promotes:
  - Network security
  - Decentralisation

State transition

Ethereum 1.0	Ethereum 2.0	Ethereum 2.x
The EVM ● State root recorded in every block. ● 256-bit architecture poor fit for most hardware. ● Limited compiler support.	eWASM ● Standardised, optimised. ● Wide range of implementations. ● Strong compiler support. ● Remove pre-compiles from the protocol.	Alternative execution engines?

Not yet decided: delayed state execution.

- Blockchain only records the history of transactions.
- State can be calculated later on-demand (“lazy evaluation”).
- May allow for alternative execution engines on the public chain.

Cryptoeconomics 1: slashing

Ethereum 1.0	Ethereum 2.0

Network security is largely incentive driven:
● revolves around mining reward.
	Network security assured by a combination of
	● Staking reward
	    ○ Rewards desirable behaviour
	● Slashing
	    ○ Punishes demonstrably bad behaviour

- “Fraud proofs”/“fault proofs” become a thing. Participants are rewarded for reporting incorrect behaviour and perpetrators are slashed.

Cryptoeconomics 2: storage rent

Ethereum 1.0	Ethereum 2.x
Blockchain storage is paid for once and persists forever.	• Charge ongoing fees for data and contract storage to reduce state size growth over time. • Provide a way for users to re-upload locally stored state to the blockchain if it gets deleted.

- This is controversial and under much discussion.
- No other blockchain does this, but it seems inevitable in the long run.

Crypto 1: aggregate block signatures

Ethereum 1.0	Ethereum 2.0	Ethereum 2.x
Not required	BLS aggregate signatures allow for large numbers of validators to be handled efficiently. ● Not quantum safe!	ZK-STARKs ● Quantum resistant

- Looking at the BLS12-381 elliptic curve for the aggregate signatures.
- ZK-STARKS.
    - It is expected that STARKs will play a big role in the future in a number of protocol aspects.

Crypto 2: randomness

Ethereum 1.0	Ethereum 2.0
- No in-protocol use of randomness. - No easy way for applications to access randomness.	Protocol fundamentally relies on availability of a random beacon. Quality of randomness expected to be sufficient for many applications

Choice of implementations for the random beacon:

- BLS threshold signatures *a la* Dfinity
    - Complex set up (DKG)
    - Not quantum secure
- RANDAO (current proposal)
    - Needs a VDF (verifiable delay function) to avoid “last actor” issue. Blergh.

Crypto 3: hash algorithm

Ethereum 1.0	Ethereum 2.0
Keccak256 (almost SHA3)	MiMC? ● Minimises multiplications. ● Allows for more efficient STARK implementations.

Applications of hashing broaden in Ethereum 2.0:

- RANDAO hash onion for the beacon chain.
- Construction of proofs-of-custody of collation bodies.
- Provision of Merkle proofs in light and/or stateless clients.
- Quantum resistant transaction signatures?

Crypto 4: account abstraction

Ethereum 1.0	Ethereum 2.x
Transaction validity is baked into the protocol: ● Signature verification ● Nonce checking ● Fee verification	Account abstraction: all accounts become contract accounts. Allow contracts to specify their own Tx validity criteria. E.g. ● Alternative signature schemes ● Flexible fees payment options

This was initially proposed for the Mainnet in April 2016.

- As of the last-but-one Core Devs' it has now been iced until Ethereum 2.0
- An example of the difficulty of making disruptive changes to the current Mainnet.

Networking

Ethereum 1.0	Ethereum 2.0
devp2p ● creaky	Sharded p2p protocol likely based on *libp2p*. ● Allow for fast shuffling of validators

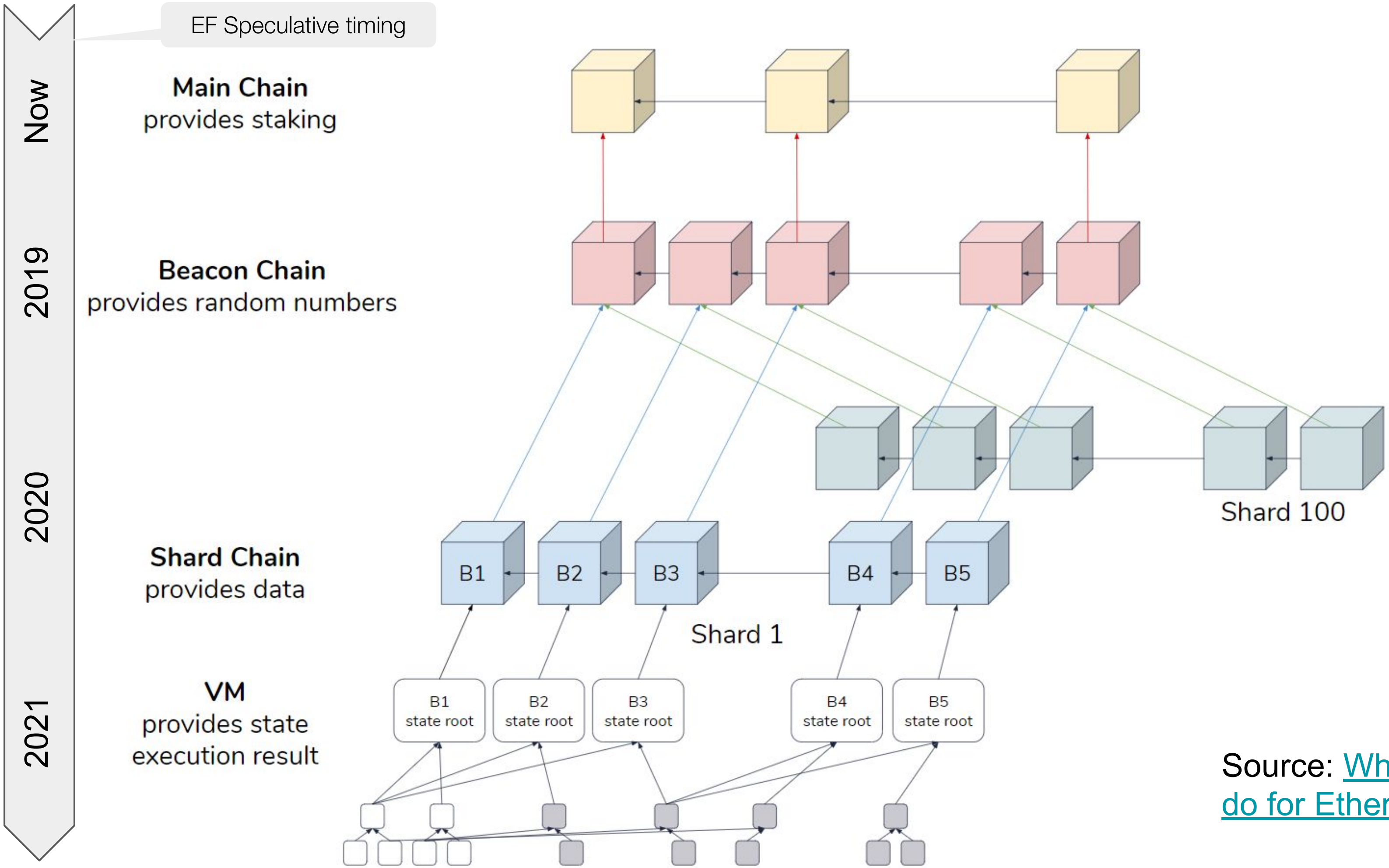
Programming model

Ethereum 1.0	Ethereum 2.x
Inter-contract calls are synchronous.	Undecided ● Sync within shards, async between? ○ ✔Natural to implement ○ ✘Developers need to be shard-aware. ● Async everywhere? ○ ✔Allows load-balance between shards. ○ ✘A major break with the current model. ● Sync everywhere? ○ ✔Maintains current programming model. ○ ✘Complex to implement?

In general, introducing asynchrony is likely to make auditing the correctness of contract systems substantially more difficult.

Delivery Timeline

(Status: speculative. Based on utterances from the Ethereum Foundation)



Source: [What you can do for Ethereum 2.0]

Phase 0: The PoS beacon chain (2019?)

- Implement PoS Beacon chain as a side chain
  - Random number generation: RANDAO using verifiable delay function.
  - Aggregate Signatures for validators
    - BLS aggregate signatures likely using BLS12-381 elliptic curve
  - Begin migration to new hash function(?)
    - Keccak256 -> MiMC hash?
  - 5s block time
  - New p2p network layer
- Brings Casper FFG periodic finality to Mainnet
- 32 Eth stake per validator
  - Slashing mechanism
- 80% Reduction in PoW mining reward (3 Eth→0.6 Eth?)

Phase 1: Sharded transaction handling (The data layer - 2020?)

- Add sharding infrastructure for transactions
  - Each shard maintains an append-only log of transaction data.
  - Implement block proposer, block attestor and block validator mechanisms.
  - Implement proofs of custody.
  - Implement cross links to beacon chain.
  - Enhance p2p networking to handle all of this.
- No state execution!
- Test the mechanism design.

Phase 2: Sharded state handling (The execution layer - 2021?)

- Add state execution to shards
    - eWASM engine to replace EVM(?)
    - Delayed state execution(?)
    - Synchronous or asynchronous intra-shard contract calls?
    - Synchronous or asynchronous inter-shard contract calls?
    - Light clients and/or stateless clients.
- A fully usable, $O(c^2)$ scaled infrastructure.
    - $O(c)$ is the compute power available to a consumer laptop or small VPS. The same speed as Mainnet today.

Phase X: Also under discussion...

- ZK-STARKs.
    - Quantum resistant zero-knowledge proofs. Lots of use cases.
- Alternative execution engines.
- Storage rent.
- Full PoS: Casper CBC.
- Account abstraction.
- Mainnet becomes a side-chain to the beacon chain.
    - Abandon PoW?