Extending Existing Blockchains with Virtualchain

Jude Nelson*, Muneeb Ali*†, Ryan Shea†, Michael J. Freedman*
Pretend cryptocurrencies do not exist
What’s in a Proof-of-Work Blockchain?

- Total ordering of writes
- “Stable” view ordering (*)
- Append-only
- 100% replicated
- Tamper-resistant
- Anyone can write
- Fixed growth rate (pay-to-play)
- Hard to upgrade once deployed
Distributed Applications and Blockchains

- Replicated state machines (RSMs) on top?
- Strategy: store input history

Bootstrapping state
Advantages

● Open app membership
● Survive total app failure
● Blockchain-agnostic
● App-agnostic
Challenges

- Blockchain failure
  - Goes offline
  - “Centralization” attacks
- Blockchain forks
  - Data loss
  - Chain reorganization
Virtualchain

- Fork*-consistent RSMs on existing blockchains
- Fork detection & recovery
- Cross-chain migration
Fork*-Consistency (Li & Maziéres, NSDI’07)

- RSMs in “fork sets”
- Fork set shares history
- Partition after fork detection

\[ \text{op}_1 \quad \text{op}_2 \quad \text{op}_3 \quad \text{op}_4 \quad \text{op}_5 \quad \text{op}_6 \quad \text{op}_7 \]

\[ FS_a = \{1,2,3,4,6\} \]
\[ FS_b = \{1,2,3,4,7\} \]
\[ FS_c = \{1,2,3,5\} \]
Nakamoto Consensus Creates Fork Sets

- Multiple leaders
- Conflicting writes

![Diagram showing block chain forks and divergent states](image-url)
Reorganitions Create Fork Sets

- Conflicting TXs discarded
- Accepted TXs re-ordered

Divergent bootstrapping state
Solution: Consensus Hashes

- In-band app-level consensus
- Used for:
  - Identifying fork sets (multiplexing)
  - Fork detection and recovery
  - Blockchain migration
  - Lightweight fork set selection
Consensus Hash Construction

- $CH(n)$: cryptographic hash
- Covers *state transition history* (“journal”)

\[
P_b = \{CH(p) \mid p = b - 2^k\}
\]

\[
CH(n) = \text{Hash(Merkle}(J_n), P_n)
\]
In-band Consensus

- Fork sets: agree on $CH(n)$ for all $n$
- Client: embed latest $CH$ in input TX
  - Obtained from preferred fork set
- Server: consider TX only if $CH$ is "recent"
  - "Send/ACK" with $K$-block timeout
**Lightweight Fork Set Selection**

- Given $CH(n)$, search for *characteristic state transitions*

<table>
<thead>
<tr>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
</tr>
</thead>
</table>

- $CH(n)$ is trusted
- $CH(n-8)$ is now trusted, because $CH(n)$ was trusted
- Block $n-11$’s journal entries can be verified now
Dealing with Blockchain Forks

- Most forks are short-lived
  - Avoid with “confirmations”
- Long-lasting forks are rare
  - But widely noticed!
  - Due to bugs or attacks
Fork/Reorganization Detection

- Continuously audit CH history
- Alert on disagreement

\[ CH(n' - 8) \neq CH(n - 8) \]
Joining Fork Sets

- CHECKPOINT from trusted party
  - Last “valid” $CH(n)$
- Host $CH(n)$’s journal off-chain
- Fetch and verify with $CH(n)$

\[ \text{CHECKPOINT from trusted party} \]
\[ \text{Last “valid” } CH(n) \]
\[ \text{Host } CH(n)’s \text{ journal off-chain} \]
\[ \text{Fetch and verify with } CH(n) \]
Cross-chain Migration

- Similar to fork set join
- Trusted MIGRATE

\[ \text{MIGRATE\_TO}(2) \]
\[ \text{MIGRATE\_FROM}(1, b) \]
\[ \text{CHECKPOINT}(CH(b)) \]

\[ J \text{ ends on blockchain 1. Final RSM state is } S_{1,b} \]

Application state at blockchain 1, block \( b \)

\[ J \text{ resumes on blockchain 2. Initial RSM state is } S_{1,b'+1} \]
On Centralization, Trust, and Cryptocurrencies

- Already trust RSM author
- **Use CHECKPOINT, MIGRATE** judiciously
  - Ignore with **no loss of security**
- Cryptocurrency: RSM input rate-limiter
  - RSMs becoming key use-case
  - Cloud market is >10x more valuable
Example: Bitcoin OP_RETURN Usage
Source: Harry Kalodner
Concluding Remarks

- In production for >1 year in Blockstack
- [https://github.com/blockstack/blockstack-virtualchain](https://github.com/blockstack/blockstack-virtualchain)
- Ali, Nelson, Shea, Freedman (ATC’16)
- Migrated from Namecoin to Bitcoin

Source: opreturn.org
Thank you!
Questions?