Blockchains & Cryptocurrencies

Scaling I

Instructor: Matthew Green
Johns Hopkins University - Fall 2024

Housekeeping

- Project updates due!
- Slides and recordings have been updated
- Midterm: Monday 11/4

News?

News?

EXCLUSIVE CURRENCIES

Federal Investigators Probe Cryptocurrency Firm Tether

Authorities looking at possible violations of anti-money-laundering and sanctions rules

By Angus Berwick Follow *, Vivian Salama* Follow *and Ben Foldy* Follow

Updated Oct. 25, 2024 2:06 pm ET

Share Resize 59 Listen (2 min)

News?

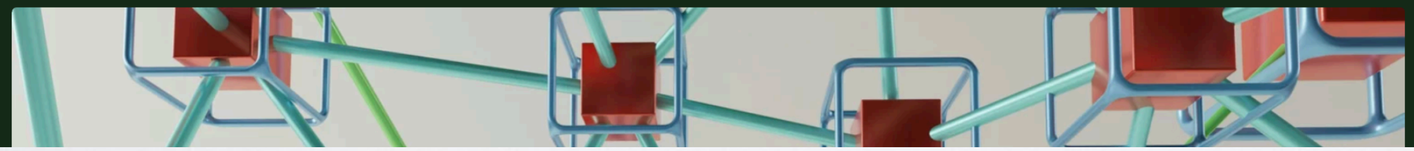
HOME / NEWS / CRYPTO / NEWS / VITALIK BUTERIN PROPOSES SOLUTION TO SHRINK ETHEREUM'S 1.1 TB NODE SIZE

NEWS 3 MIN READ

Vitalik Buterin Proposes Solution to Shrink Ethereum's 1.1 TB Node Size

PUBLISHED 7 HOURS AGO

BY PRASHANT JHA EDITED BY INSHA ZIA



SHARE ON

News?

KEY TAKEAWAYS

- Vitalik Buterin addressed Ethereum's bloated network.
- Buterin's "Purge" aims to decrease node storage requirements through History Expiry and Merkle proofs.
- The Ethereum co-founder seeks to find a balance between node size and network scalability.

Possible futures of the Ethereum protocol, part 5: The Purge

2024 Oct 26

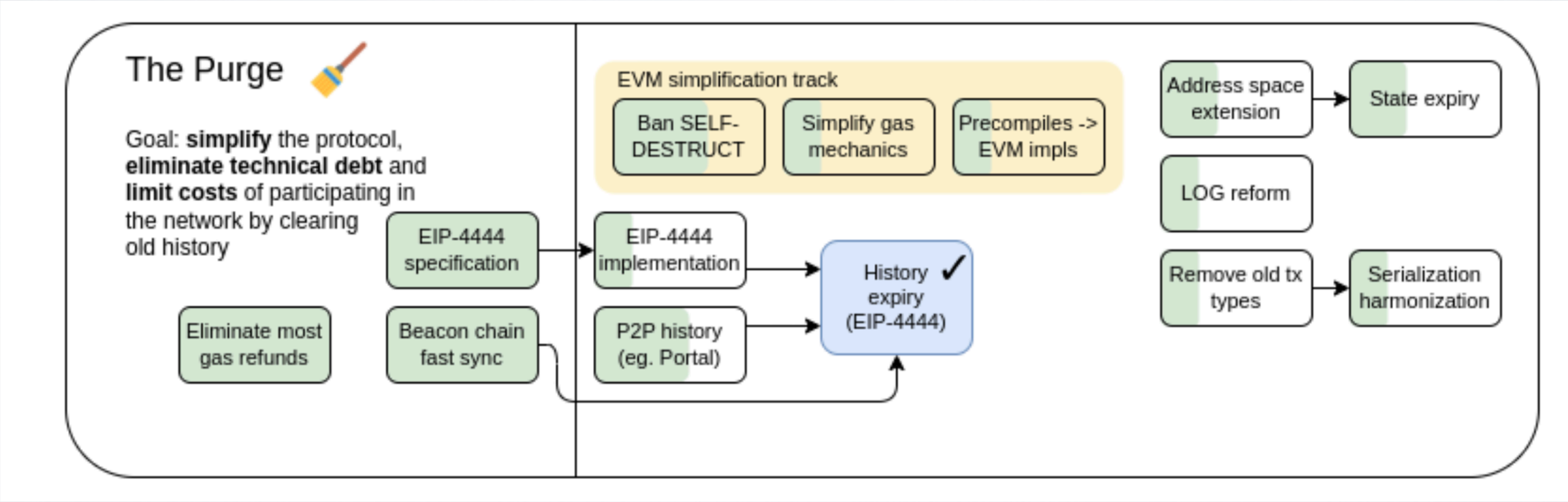
See all posts

Special thanks to Justin Drake, Tim Beiko, Matt Garnett, Piper Merriam, Marius van der Wijden and Tomasz Stanczak for feedback and review

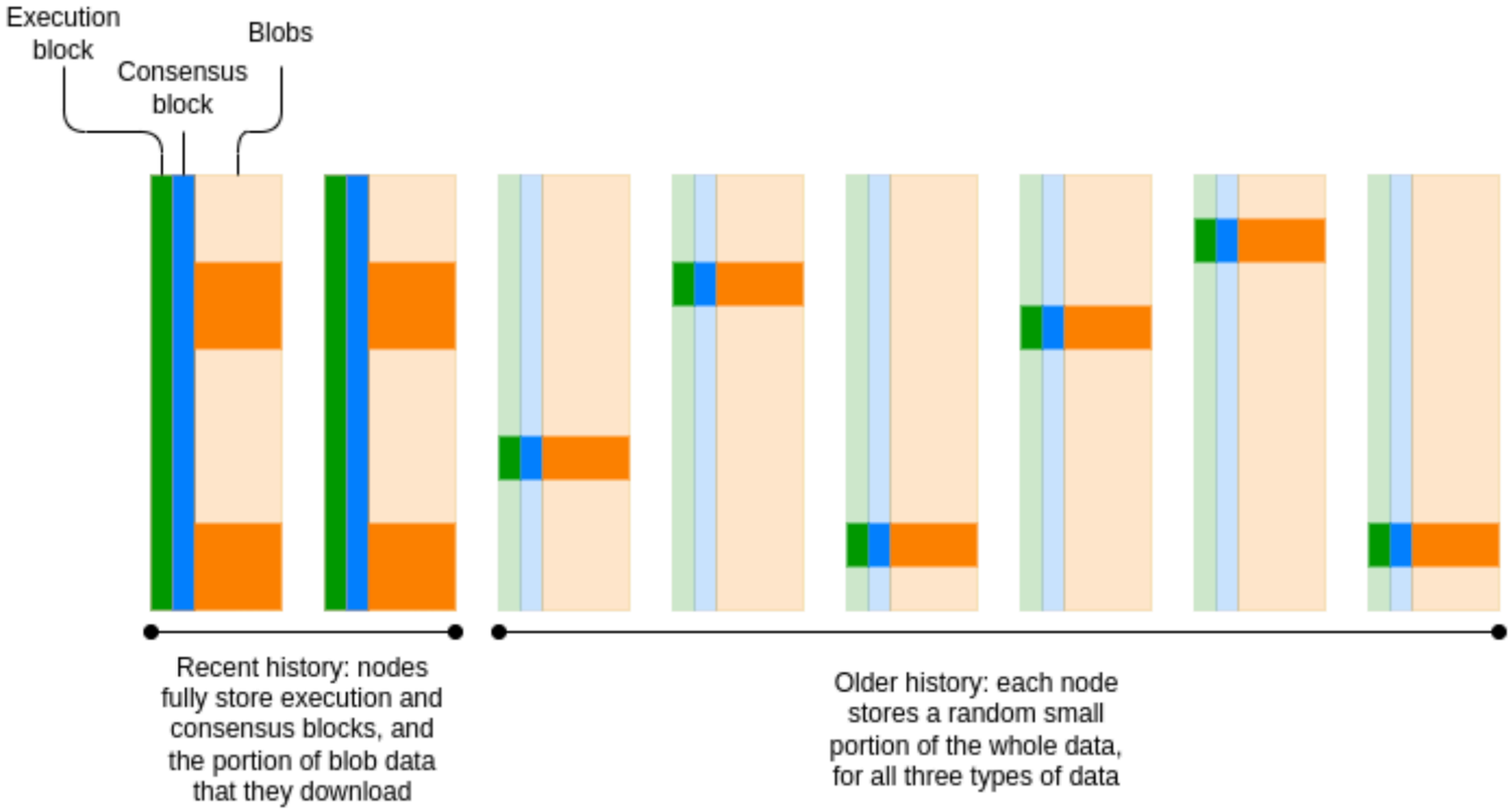
One of Ethereum's challenges is that by default, any blockchain protocol's bloat and complexity grows over time. This happens in two places:

- **Historical data**: any transaction made and any account created at any point in history needs to be stored by all clients forever, and downloaded by any new clients making a full sync to the network. This causes client load and sync time to keep increasing over time, even as the chain's capacity remains the same.
- **Protocol features**: it's much easier to add a new feature than to remove an old one, causing code complexity to increase over time.

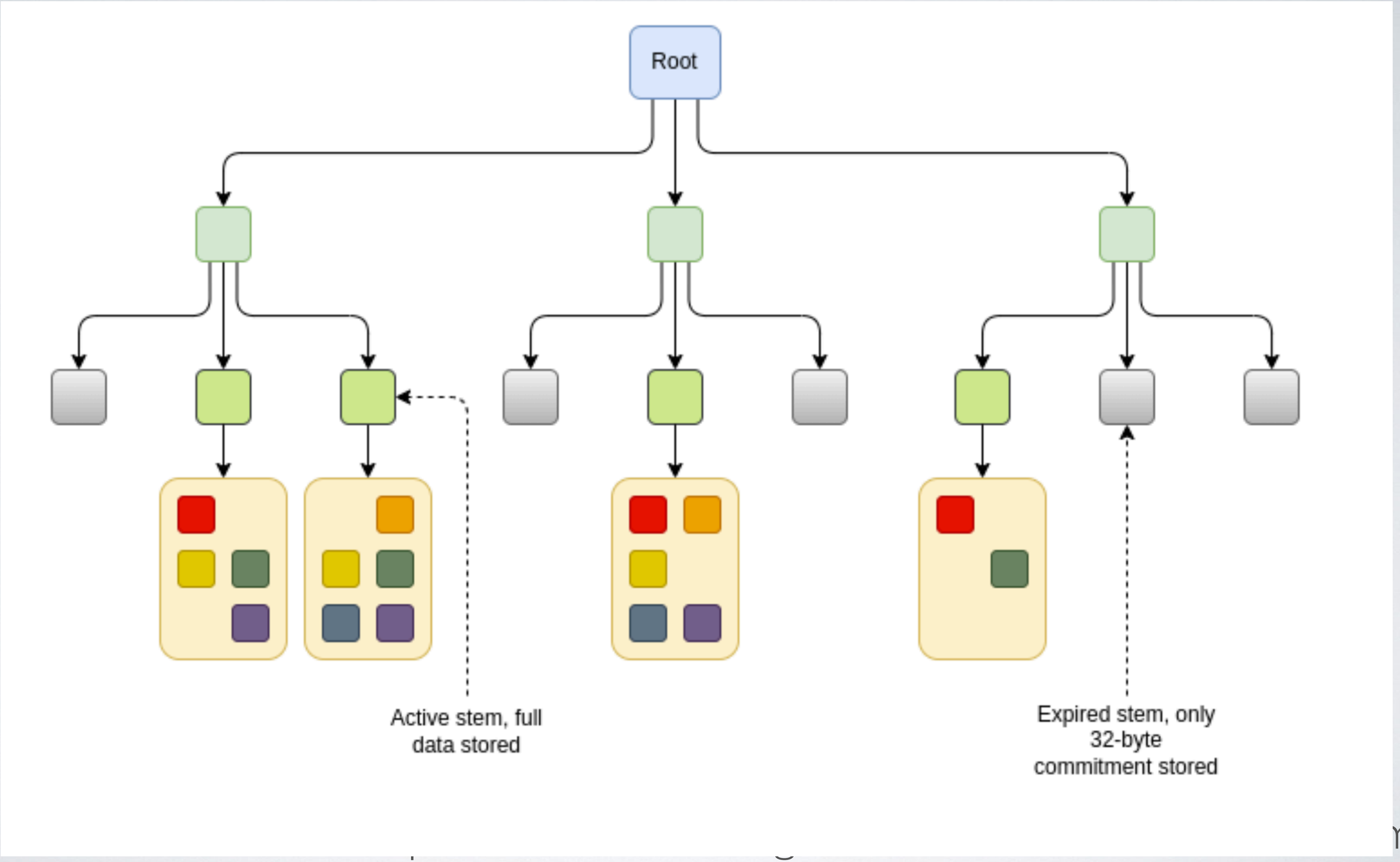
News?



https://vitalik.eth.limo/general/2024/10/26/futures5.html



https://vitalik.eth.limo/general/2024/10/26/futures5.html


Root
Active stem, full
data stored
Expired stem, only
32-byte
commitment stored

Address space contraction

Another approach goes the opposite direction: we immediately forbid some 2^{128}-sized sub-range of addresses (eg. all addresses starting with `0xffffffff`), and then use that range to introduce addresses with address periods and 14-byte hashes.

```
0xffffffff000169125d5dFcb7B8C2659029395bdF
```

The key sacrifice that this approach makes, is that it introduces security risks for **counterfactual addresses**: addresses that hold assets or permissions, but whose code has not yet been published to chain. The risk involves someone creating an address which claims to have one piece of (not-yet-published) code, but also has another valid piece of code which hashes to the same address. Computing such a collision requires 2^{80} hashes today; address space contraction would reduce this number to a very accessible 2^{56} hashes.

The key risk area, counterfactual addresses that are *not* wallets held by a single owner, is a relatively rare case today, but is likely to become more common as we enter a multi-L2 world. The only solution is to simply accept this risk, but identify all common use cases where this may be an issue, and come up with effective workarounds.

News?

Target	Date	Estimated loss
US government-linked crypto wallet	October 25, 2024	$20 Million
The Morpho PAXG/USDC Market	October 13, 2024	$230 K
Crypto Whale on a Blast Network	October 11, 2024	$35 Million tokens
Onyx Protocol`	September 26, 2024	$3.8 Million
BingX	September 20, 2024	$43 Million
Indodax	September 11, 2024	$22 Million
WazirX	July 18th, 2024	$234.9 Million
LIFI	July 16th, 2024	$10 Million
Bittensor Blockchain	July 3rd, 2024	$ 8 Million TAO Tokens

Other anonymity approaches!

Scalability

The problem

- Bitcoin transaction rate: 5-7 tx/sec
  - Bounded by block size (Segwit helps), TX size
  - All transactions must be globally verified, stored
- Ethereum: 15 transactions per second if they're small
- Visa: 24,000/sec peak (150M/day globally)
- WeChat 256,000/sec peak

Faster computers?

- Why not just build faster computers?

Faster computers?

- Why not just build faster computers?
  - Loss of decentralization
  - Eventually we saturate links, due to broadcast network
  - Replicated global state falls apart
  - Scaling is possible (see Visa, WePay etc.) but it requires dedicated, centralized servers

Can we do better?

Can we do better?

- Current ideas:
  - “Off chain”
  - “Sharding”
  - New consensus algorithms

Off-chain transactions (channels)

- In current Bitcoin-style networks, every transaction appears on the blockchain
  - This allows the whole network to verify financial integrity
  - I.e., we can't go off an do transactions elsewhere, accidentally/deliberately inflate the money supply
- But why does the network need to see every transaction?

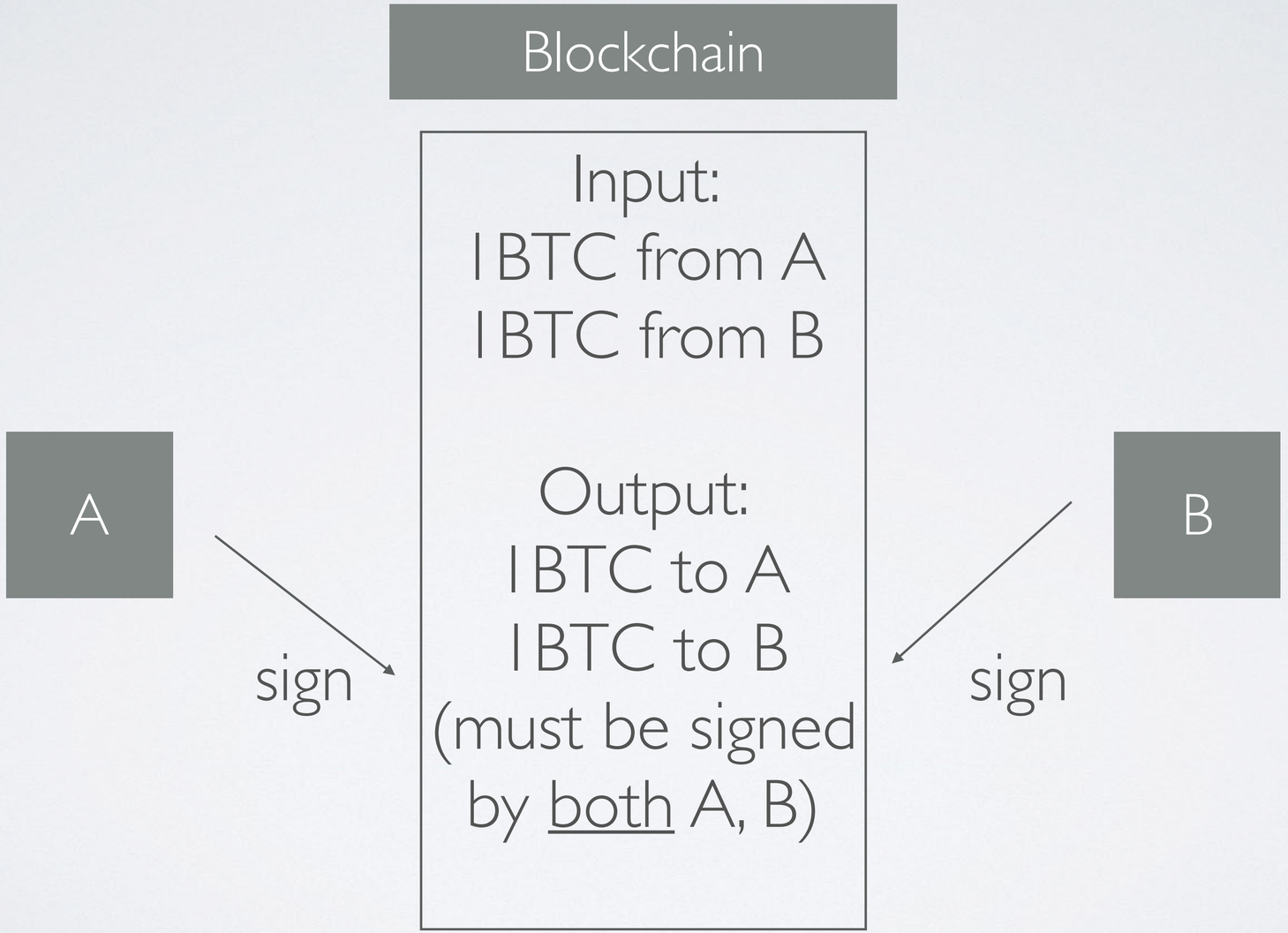
Off-chain transactions (channels)

- Overarching idea:
  - If a transaction doesn't affect anyone else (except for the parties willing to risk money), chain doesn't need to see it
  - Simplest example (but centralized):
    - Multiple parties deposit money into an exchange
    - Exchange is just a centralized bank, so everyone can quickly transmit money by adjusting balances
    - Only withdrawals need on-chain transactions

Off-chain transactions (channels)

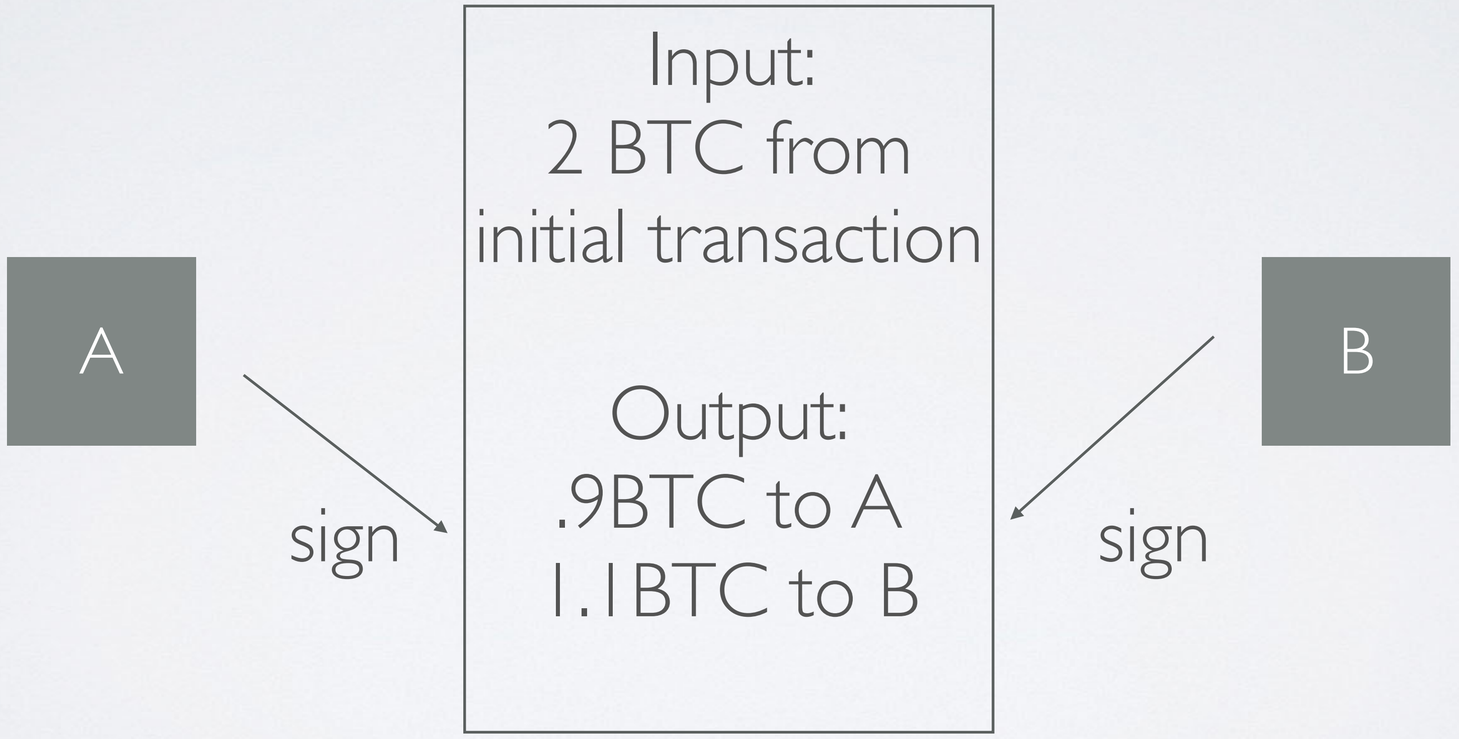
- Off-chain exchange example still risks loss of funds
  - If the exchange disappears, your money goes with it
  - See e.g., QuadrigaCX
  - The only benefit here is that the rest of the network can't lose money, e.g., due to inflation

Channels: Step 1



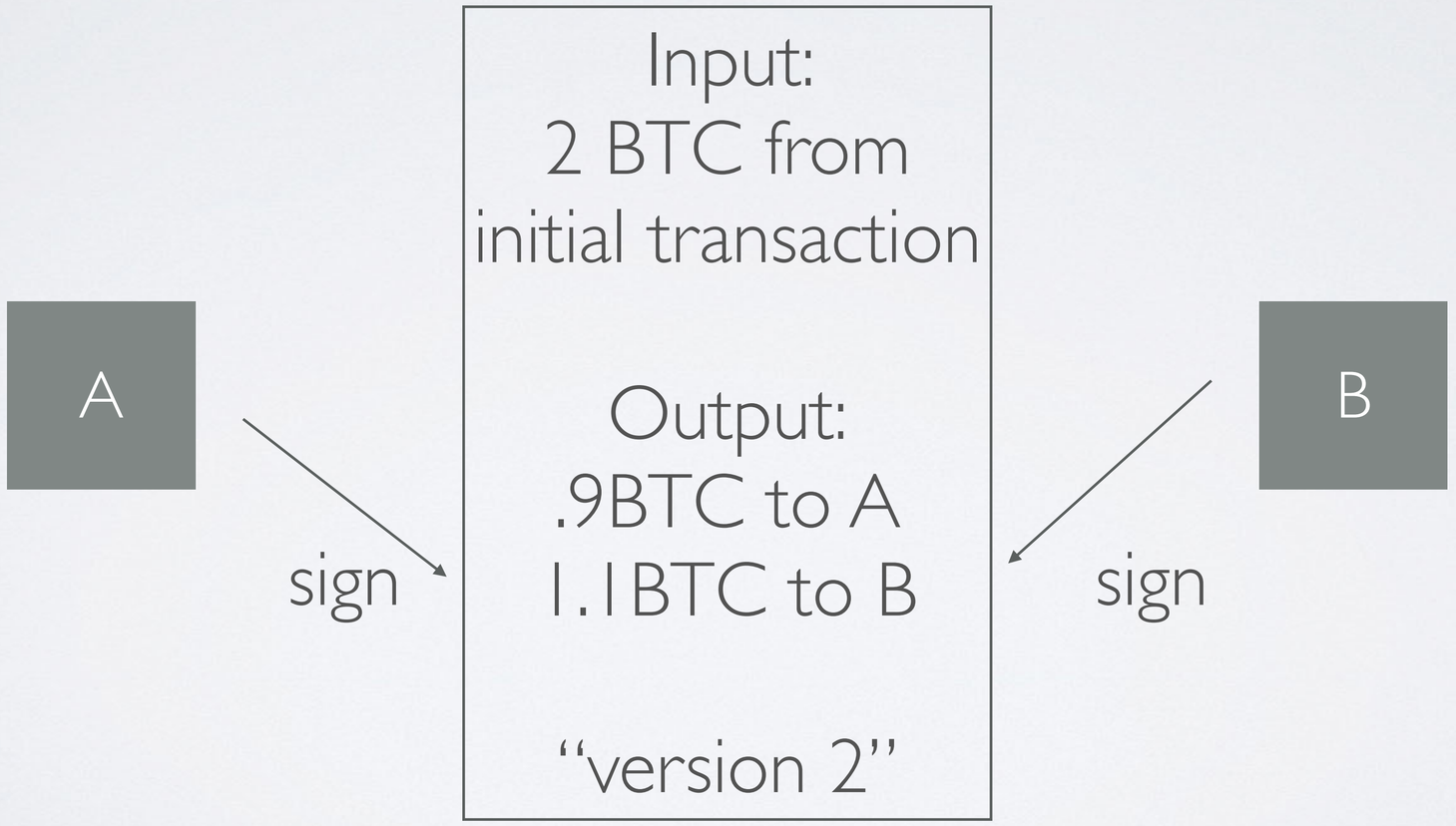
Channels: Step 2

* A pays B 0.2BTC



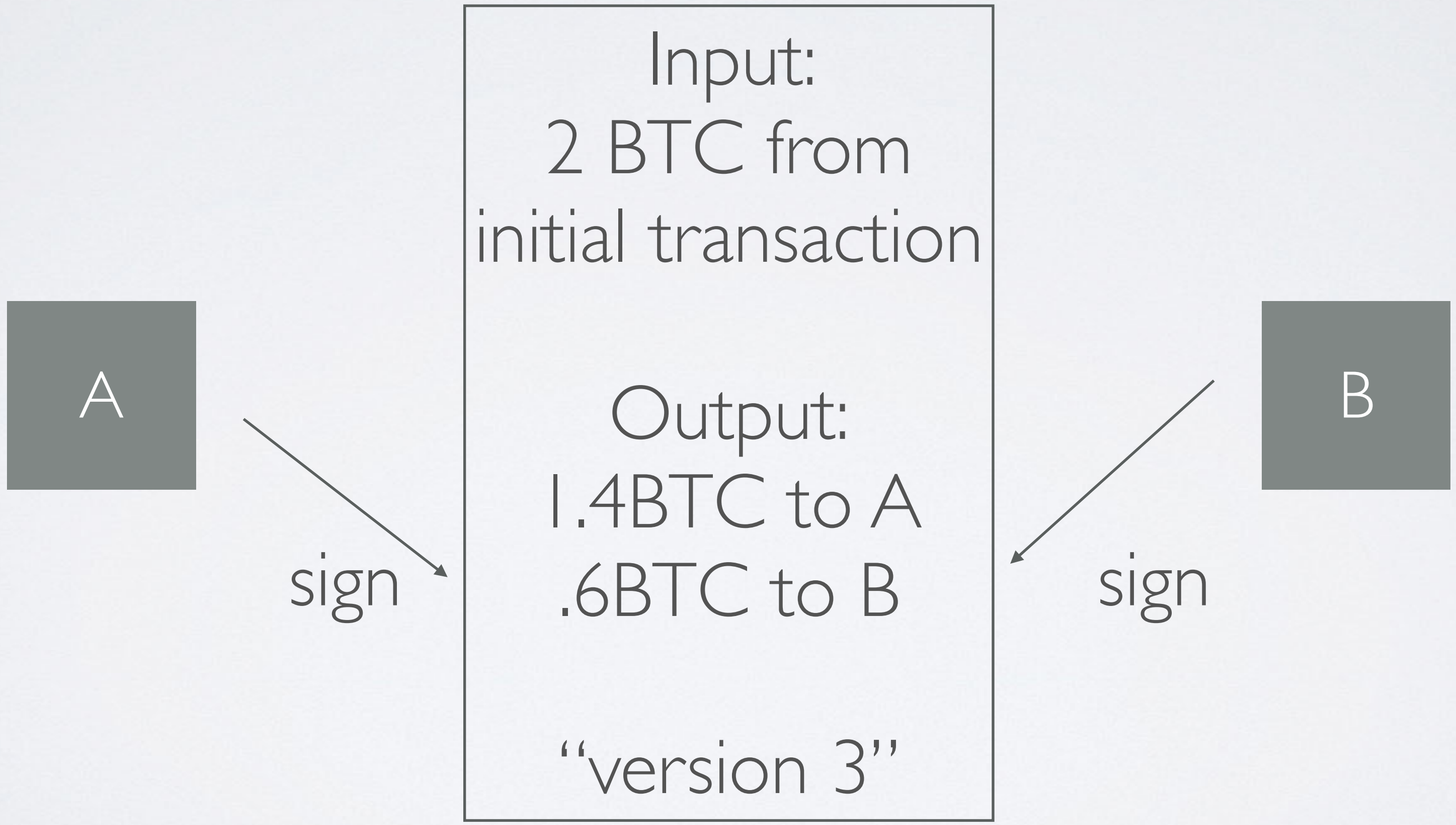
Channels: Step 2

* A pays B 0.2BTC



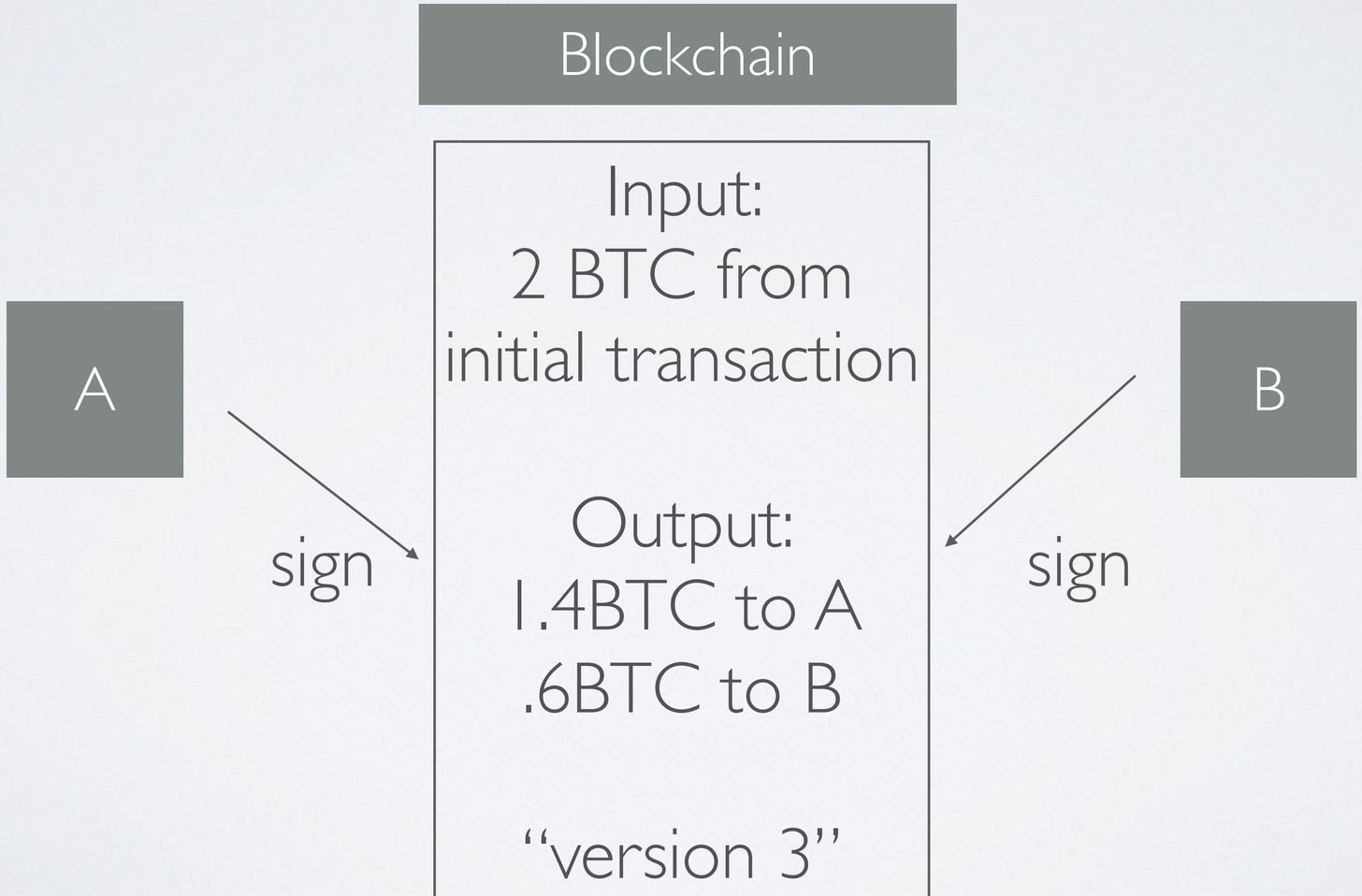
Channels: Step 3……

* B pays A .5 BTC



Channels: Closure

* Either party posts the most recent version of the transaction to the blockchain (all older versions get ignored)



Dispute resolution

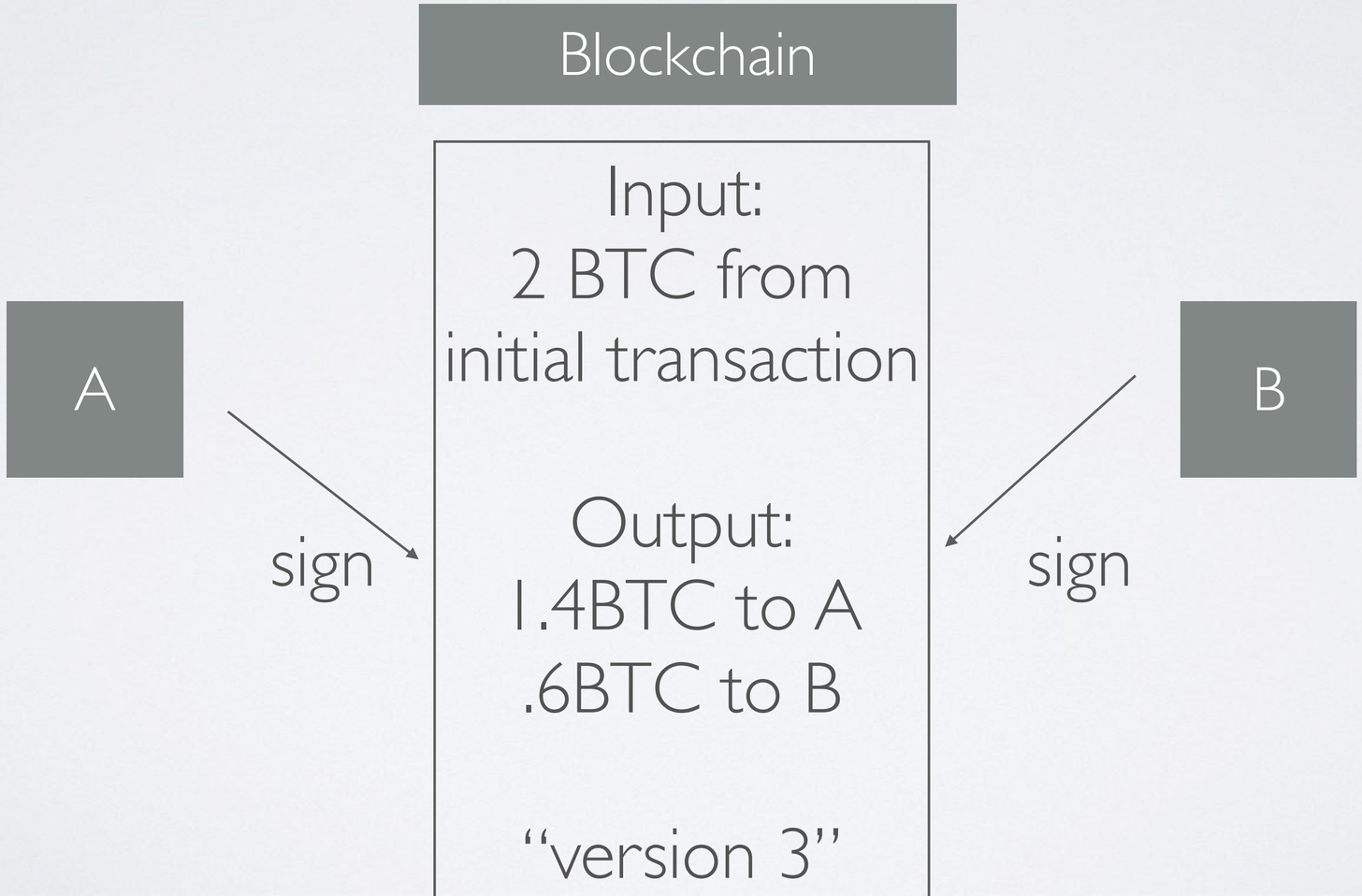
* What if someone posts an older, out-of-date version?
* What if nobody signs the first "closure" transaction? How do you escape a payment channel in the worst case?

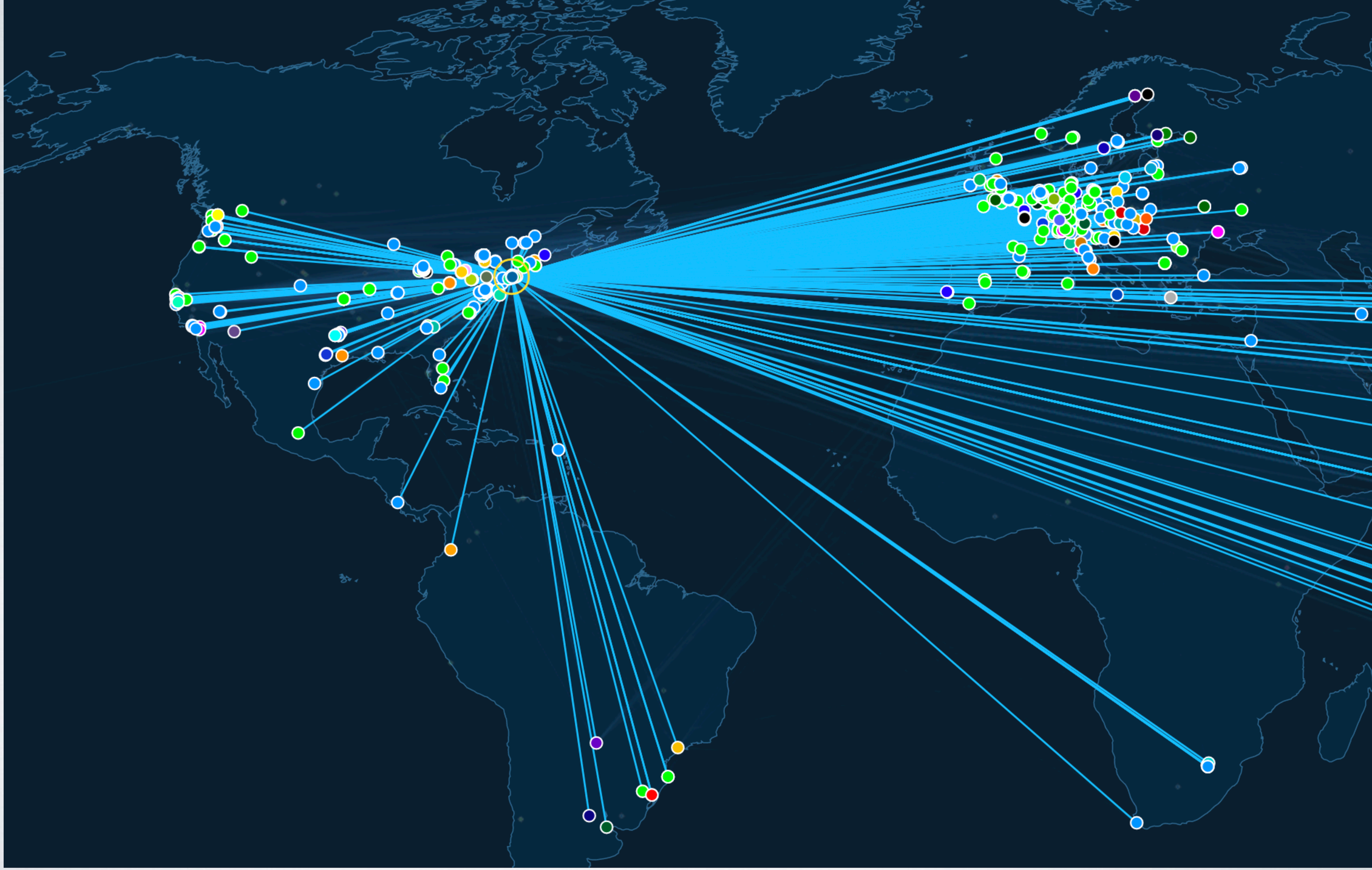
Off-chain transactions (channels)

- In current Bitcoin-style networks, every transaction appears on the blockchain
  - This allows the whole network to verify financial integrity
  - I.e., we can't go off an do transactions elsewhere, accidentally/deliberately inflate the money supply
- But why does the network need to see every transaction?

Channels: Closure

* Either party posts the most recent version of the transaction to the blockchain (all older versions get ignored)





Making channels atomic

- What happens in a channel A->B->C when one party fails to complete the transaction?
  - Answer is that we need to make each channel transaction "atomically" depend on later transactions
  - We will have C release a password to allow payment from B->C, and this will be the same password that allows A->B transaction to go through

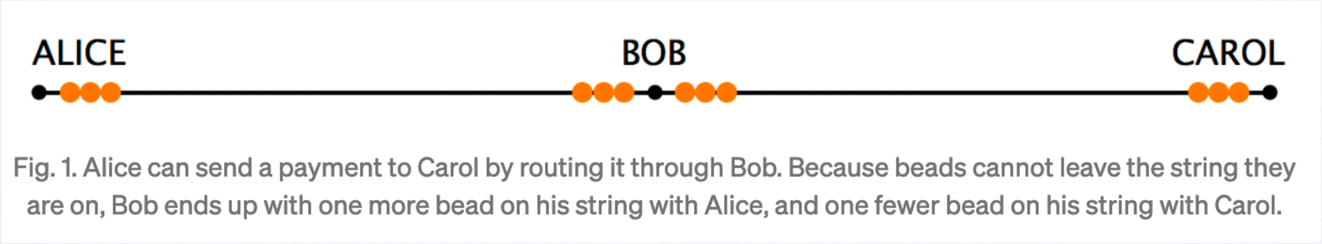


Fig. 1. Alice can send a payment to Carol by routing it through Bob. Because beads cannot leave the string they are on, Bob ends up with one more bead on his string with Alice, and one fewer bead on his string with Carol.

https://medium.com/@peter_r/visualizing-htlcs-and-the-lightning-networks-dirty-little-secret-cb9b5773a0

Making channels atomic

- What happens in a channel A->B->C when one party fails to complete the transaction?
  - This is done using hash locks: each transaction embeds **h** such that to unlock and complete payment, payer must know a **p** such that **h** = SHA2(**p**)
  - Every channel in the chain will embed the same **h**

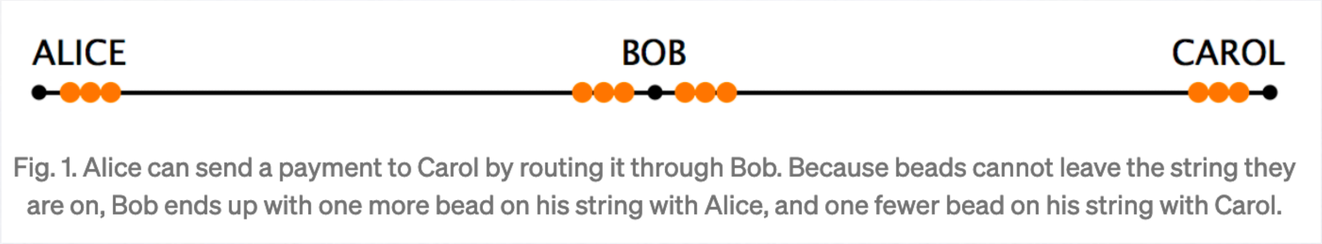


Fig. 1. Alice can send a payment to Carol by routing it through Bob. Because beads cannot leave the string they are on, Bob ends up with one more bead on his string with Alice, and one fewer bead on his string with Carol.

https://medium.com/@peter_r/visualizing-htlcs-and-the-lightning-networks-dirty-little-secret-cb9b5773a0

Making channels atomic

- What happens in a channel A->B->C when one party fails to complete the transaction?
  - We will also use timelocks to allow the channels to close if someone along the channel refuses to update their transaction
  - The combination: hash timelocks (HTLC)

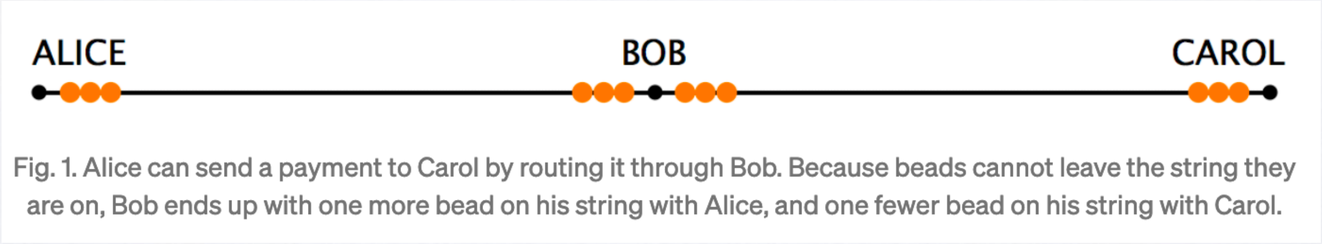


Fig. 1. Alice can send a payment to Carol by routing it through Bob. Because beads cannot leave the string they are on, Bob ends up with one more bead on his string with Alice, and one fewer bead on his string with Carol.

https://medium.com/@peter_r/visualizing-htlcs-and-the-lightning-networks-dirty-little-secret-cb9b5773a0

Problems

- There are many potential problems with channels/LN
  - Requires pairwise channels **or** lots of "locked up" liquidity in the network to support routing
  - This liquidity must be controlled by "online" signing keys, which makes it vulnerable to hacking
  - LN nodes can charge fees for participating in channels, but not clear how much people will pay

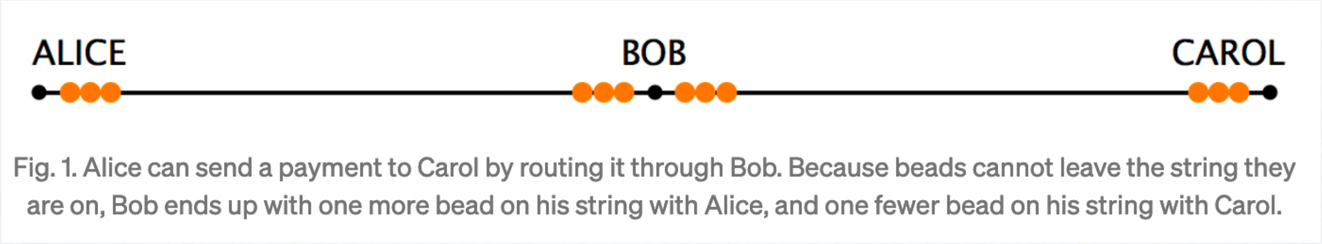


Fig. 1. Alice can send a payment to Carol by routing it through Bob. Because beads cannot leave the string they are on, Bob ends up with one more bead on his string with Alice, and one fewer bead on his string with Carol.

https://medium.com/@peter_r/visualizing-htlcs-and-the-lightning-networks-dirty-little-secret-cb9b5773a0

Problems

- There are many potential problems with channels/LN
  - Routing is hard technically. You need to find a route with enough liquidity from A->Z (via B, C, D… etc.) and that implies a lot of knowledge and comp. effort
  - If parties along the channel fail (perhaps deliberately) there may be a costly "unwinding" of each independent channel back onto the chain

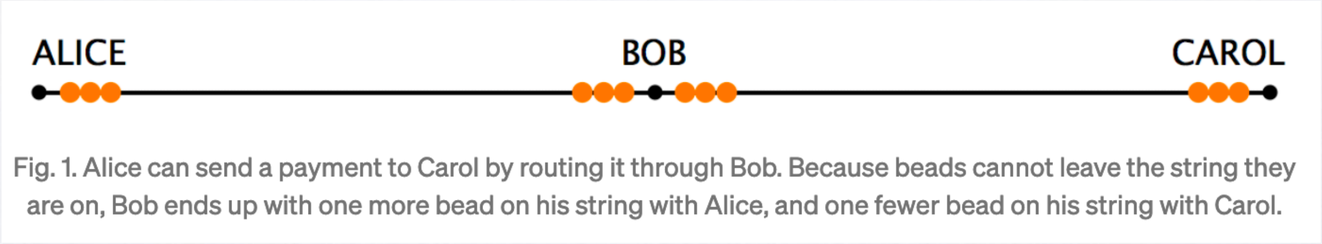
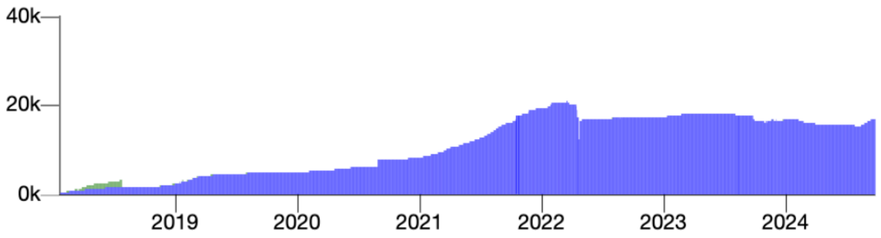


Fig. 1. Alice can send a payment to Carol by routing it through Bob. Because beads cannot leave the string they are on, Bob ends up with one more bead on his string with Alice, and one fewer bead on his string with Carol.

https://medium.com/@peter_r/visualizing-htlcs-and-the-lightning-networks-dirty-little-secret-cb9b5773a0

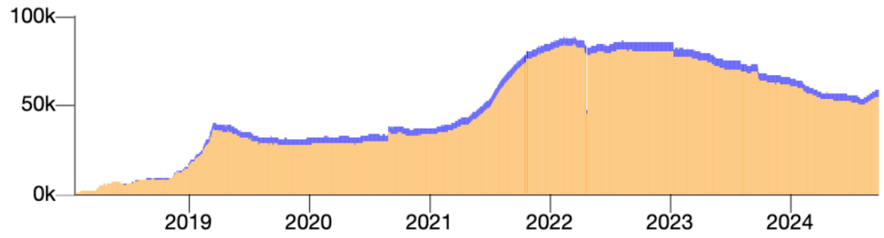
Nodes

Number of nodes with and without channels.



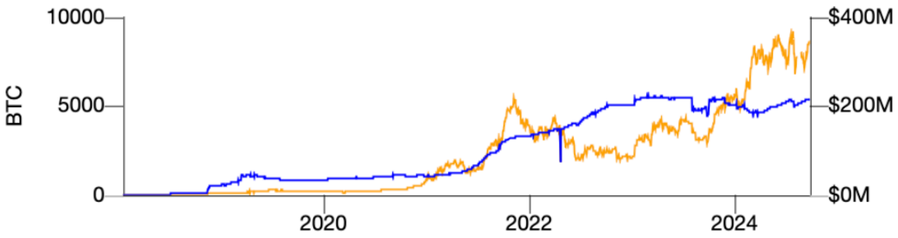
Channels

Unique = channels connecting nodes directly for the first time. Duplicate = channels between nodes that are already connected.



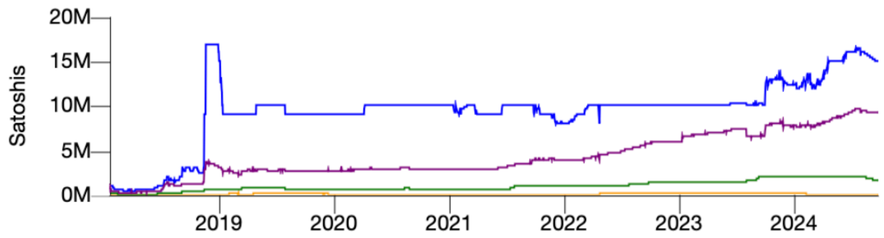
Network Capacity

Cumulative bitcoin capacity across all channels.



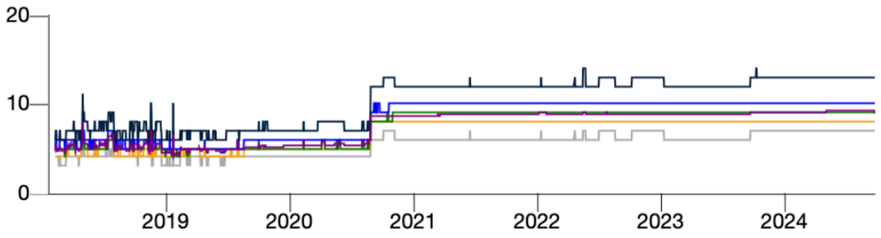
Capacity Per Channel

Channel capacity statistics.



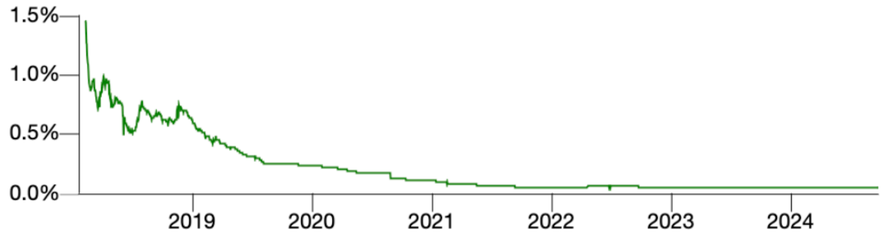
Distance Measures

The maximum number of hops required to reach another node (among shortest paths).



Completeness Measures

Density is a ratio of actual / potential channels.



Clustering Measures

Transitivity is the ratio of potential triangles present. A value of 1 means every path of length 2 loops back into a triangle. Clustering coefficient is the ratio of interconnections between a node's peers. A value of 0 means the node is a hub, and none of its peers are connected. A value of 1 means the node forms a clique with its peers.

Connectivity Measures

A cut channel (aka cut edge, or bridge) is a channel between two nodes that connects different components of the network. This channel's removal would prevent other nodes from having a path. A cut node (aka cut vertex) is the same idea, except it's a crucial node instead of a channel.

Q: Can we move this idea to Ethereum?

- Why or why not?

Two techniques

- Both go by the name "rollup": signifies that the idea is to take a chain of many sequential transactions and "compress" them into a small value that can be verified on chain

  - **Optimistic rollup:** Let's do all the transactions off-chain without verifying them, and publish them to the world in the hope that they're valid. If any transaction turns out to be *invalid*, we provide an incentivized mechanism to post a "proof" of invalidity.

  - **ZK rollup**: Let's use the magic of zero-knowledge (VC) to prove that we verified all the transactions, and produce a small proof.

Two techniques

- Both go by the name "rollup": signifies that the idea is to take a chain of many sequential transactions and "compress" them into a small value that can be verified on chain

  - **Optimistic rollup:** Let's do all the transactions off-chain without verifying them, and publish them to the world in the hope that they're valid. If any transaction turns out to be *invalid*, we provide an incentivized mechanism to post a "proof" of invalidity.

  - **ZK rollup**: Let's use the magic of zero-knowledge (VC) to prove that we verified all the transactions, and produce a small proof.

Optimistic rollup (one concept)

- Imagine we have a series of transactions and we want to prove they are all valid (in a short on-chain transaction)
  - We designate a third party ("bonded aggregator") who locks up some currency (ETH) to pay for misbehavior
  - They collect all of the transactions people sent them, and execute the transaction **off chain**
  - For each TX they compute a Merkle tree over the TXes, and publish them too (Merkle root + transactions)
  - Finally they publish a single TX to the Ethereum chain, containing the Merkle root and some extra logic (—>)

Optimistic rollup (one concept)

- Imagine we have a series of transactions and we want to prove they are all valid (in a short on-chain transaction)
  - This extra logic supports "fraud proofs" of two types:
    - If anyone can provide a proof that a single transaction in the chain is **invalid**, they can "punish" the aggregator
    - If anyone can provide a receipt that says their own TX was included in the chain, but it isn't in the rollup, then they can "punish" the aggregator
  - Punishment means "take some or all of the bond"

What guarantees do we get?

- Imagine that an aggregator is malicious
  - Example: they want to inject invalid transactions into an ERC20 contract that gives them money they shouldn't have
  - Benefit of the attack (to malicious aggregator) is potentially quite high! A single invalid TX can be worth millions USD
  - Downside is potential for getting caught, and being "slashed" (punished)

What guarantees do we get?

- Imagine that an aggregator is malicious
  - Key requirement is that the transactions in the rollup chain are published widely enough that some honest node will discover malicious behavior
  - Might need incentive mechanisms to make sure people validate the whole chain. But who keeps the validators honest?
  - How does this work in Ethereum L1 (on chain?)

ZK Rollup

- A different property, uses the magic of "verifiable computation", and cryptographic "proving technology"
  - Basic assumption is that we have a "proving system" that can take the inputs and outputs of some program, and produce a **short** proof that the program has been executed correctly
  - There are many older and emerging technologies for this: SNARKs, STARKs, IOPs, PCPs, etc. etc.
  - Key property is that if a proof exists, then the program is (almost certainly) correctly-executed

ZK Rollup

- Basic idea:
  - Aggregator (may be malicious) collects transactions from participants, writes a "receipt" for each TX it receives
  - Aggregator verifies each (sequential) TX using EVM, updates state
  - Aggregator submits a Merkle root over all the transactions, plus a **short verifiable proof** of the following:
    1. Each TX verifies w.r.t. input state
    2. Merkle root is computed over all TXes and state
  - Blockchain (L1) simply verifies this proof

Does any of this stuff work yet?

- Great question, there are a billion proposals.
  - Can't wait for your **research projects!**