Blockchains & Cryptocurrencies

Scaling II (Eth 2)

Instructor: Matthew Green
Johns Hopkins University - Fall 2020

Housekeeping

- Assignment 3: Nov 23
- Please respond to Gijs about presentation times **if you have not already**
  - So far we have last 2 class periods dedicated to presentations (10 min + 2min questions)
  - Dec 2 + Dec 7: assigned slots will be sent out soon
- Also, no class Dec 9 because JHU calendar sucks and I misunderstood the schedule :(

News?

Cryptocurrency: Bitcoin hits three-year high as investors jump in

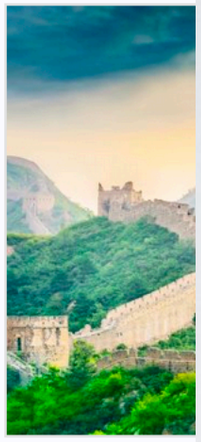
By Justin Harper
Business reporter, BBC News

11 hours ago

News?

Grin Network Victim of 51% Attack, Unknown Miner Commands 58% of the Hashrate

In Case



Bitcoin Asc
State-Back

China's Cer
Developme
Ministry of I
Technology
crypto proj
are ranked
as ... **read r**

The Lat

News?

At the time of publication, an unknown miner controls 58.1% of the Grin network's hashrate. Statistics also show that at approximately 6:17 p.m. (EST) the unknown entity reorganized a single block at height 0000ada4. The mining pool 2miners who tweeted about the incident have around 24.5% of the global Grin hashrate today.

This is followed by Sparkpool, F2pool, Grinmint, and other small miners pointing hash at the network. If a mining entity controls more than 51% of a cryptocurrency network, the attacker can potentially reorganize blocks and invalidate transactions.

Ethereum Classic (ETC) has been notorious for getting 51% attacked on various occasions, alongside this the blockchain Bitcoin Gold (BTG) has also been 51% a few times. Much of the blame on all of these attacks have been cast at the Nicehash firm, a cloud mining operation that allows users to rent hashrate.

Currently, the price of grin is trading for $0.233 per coin, and there's $3.4 million in global trade volume. The price of grin (GRIN) has lost 2.9% during the

Review stuff (from last time)

The problem

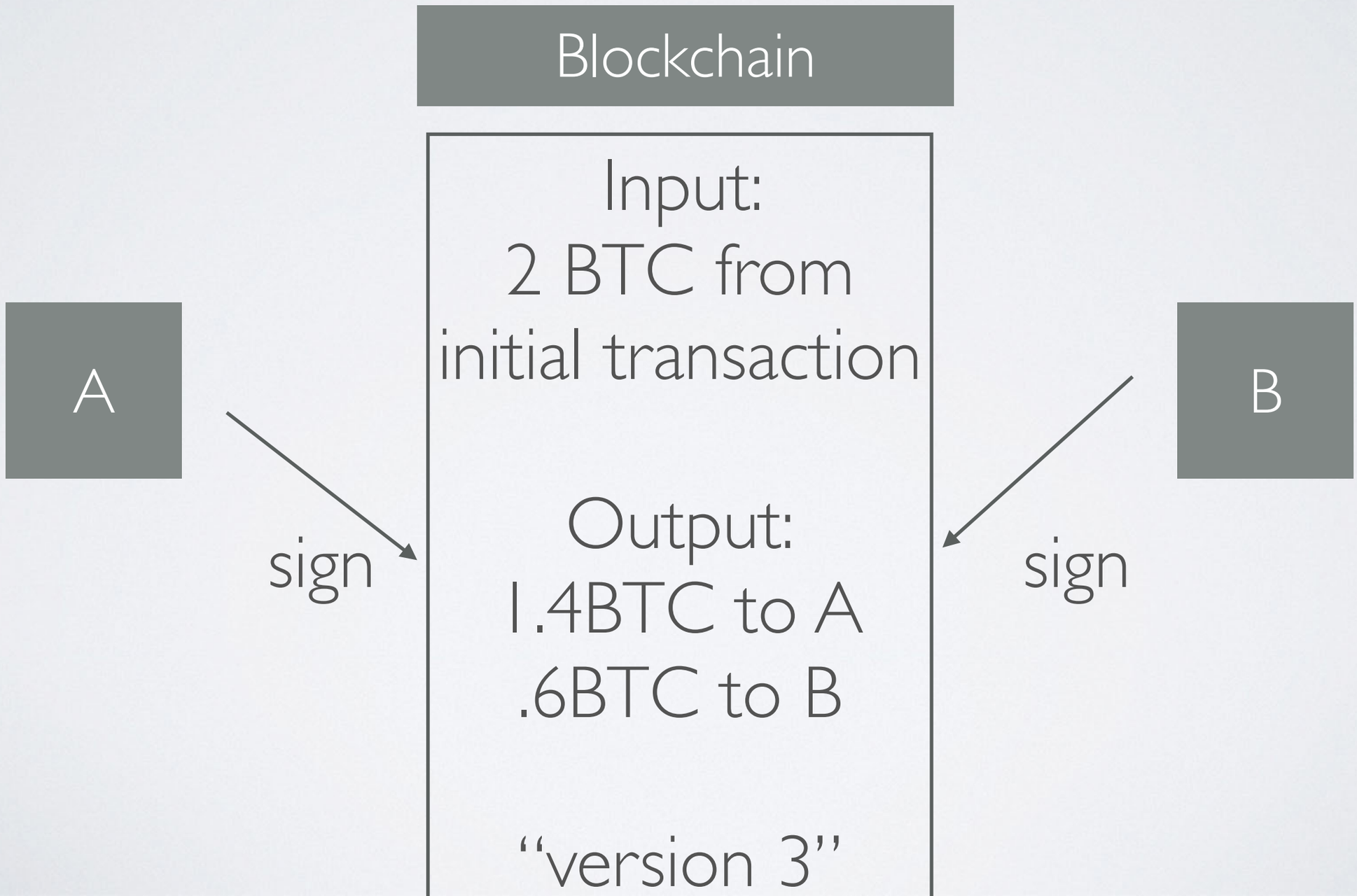
- Bitcoin transaction rate: 5-7 tx/sec
  - Bounded by block size (Segwit helps), TX size
  - All transactions must be globally verified, stored
- Ethereum: 15 transactions per second if they're small
- Visa: 24,000/sec peak (150M/day globally)
- WeChat 256,000/sec peak

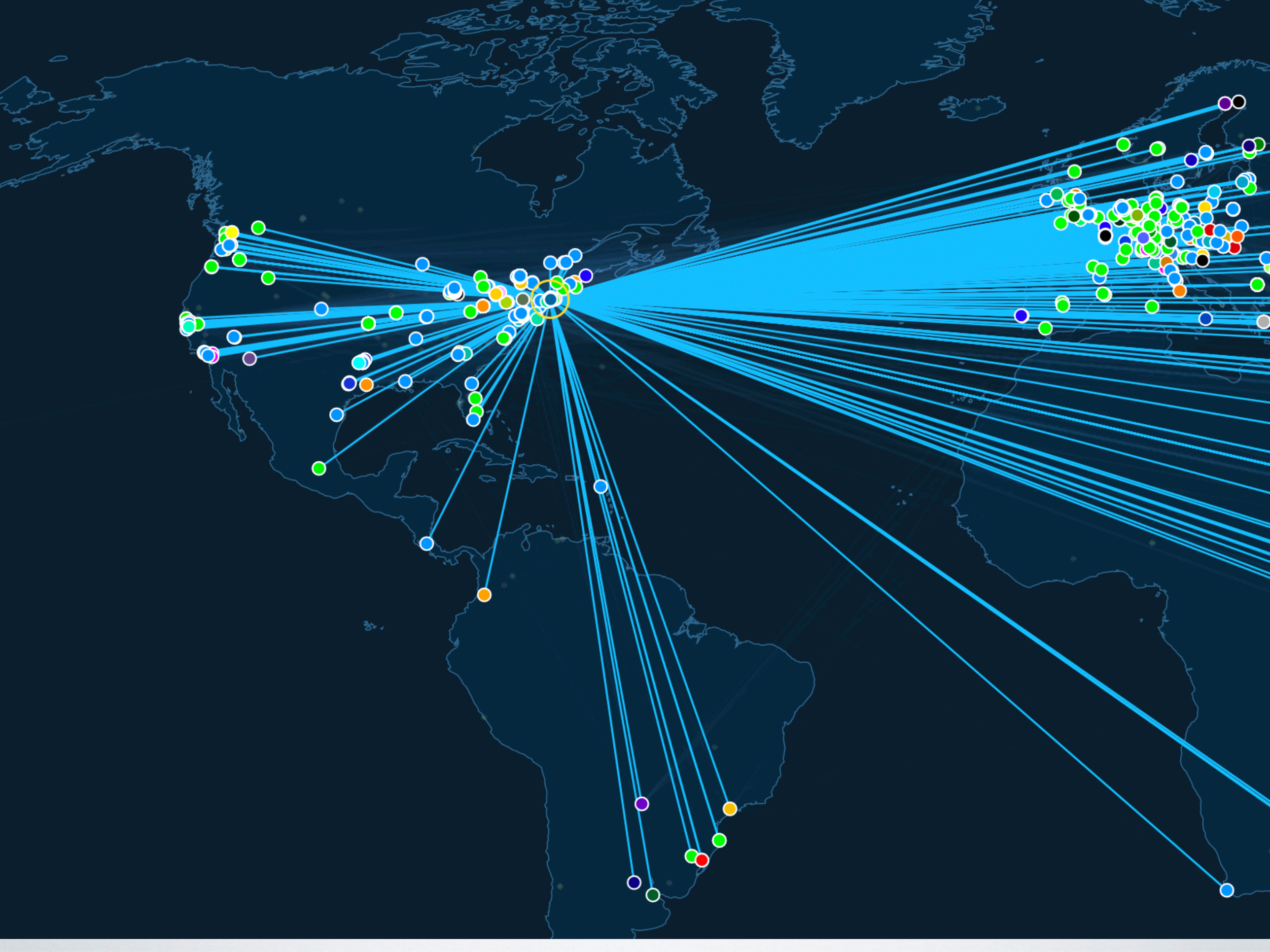
Off-chain transactions (channels)

- 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 <u>every</u> 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

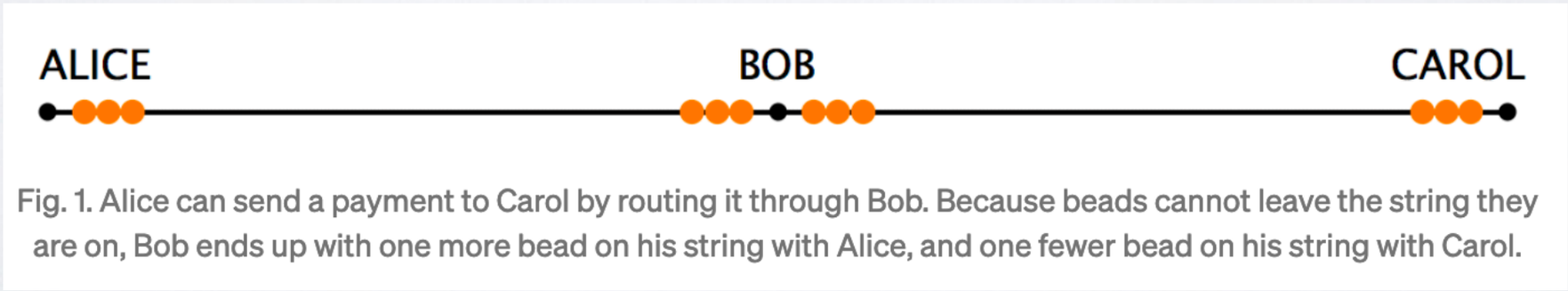


Fig. 1. Alice can send a payment to Carol by routing it through Bob. Because beads cannot leave the string they are on, Bob ends up with one more bead on his string with Alice, and one fewer bead on his string with Carol.

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**

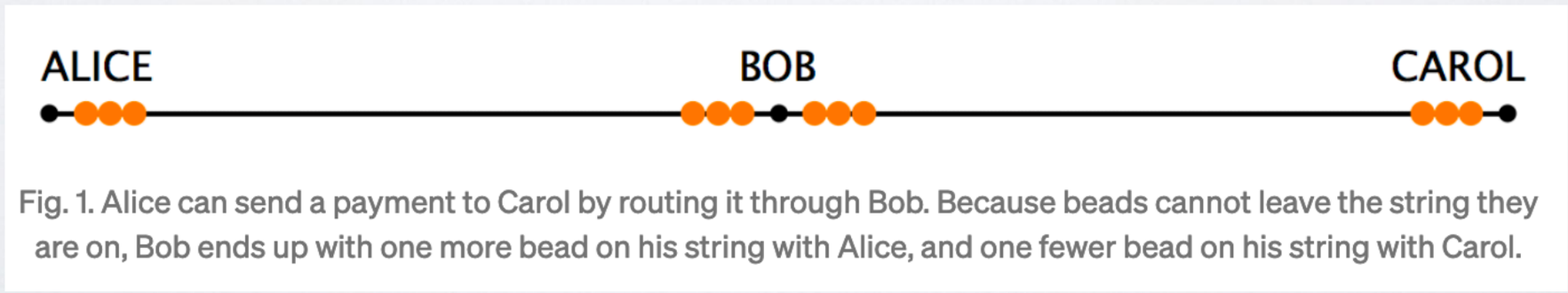


Fig. 1. Alice can send a payment to Carol by routing it through Bob. Because beads cannot leave the string they are on, Bob ends up with one more bead on his string with Alice, and one fewer bead on his string with Carol.

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)

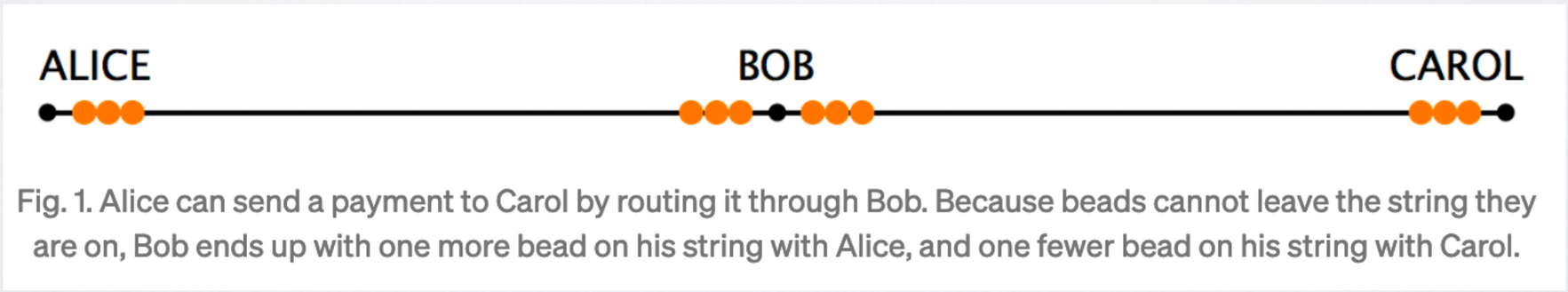


Fig. 1. Alice can send a payment to Carol by routing it through Bob. Because beads cannot leave the string they are on, Bob ends up with one more bead on his string with Alice, and one fewer bead on his string with Carol.

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

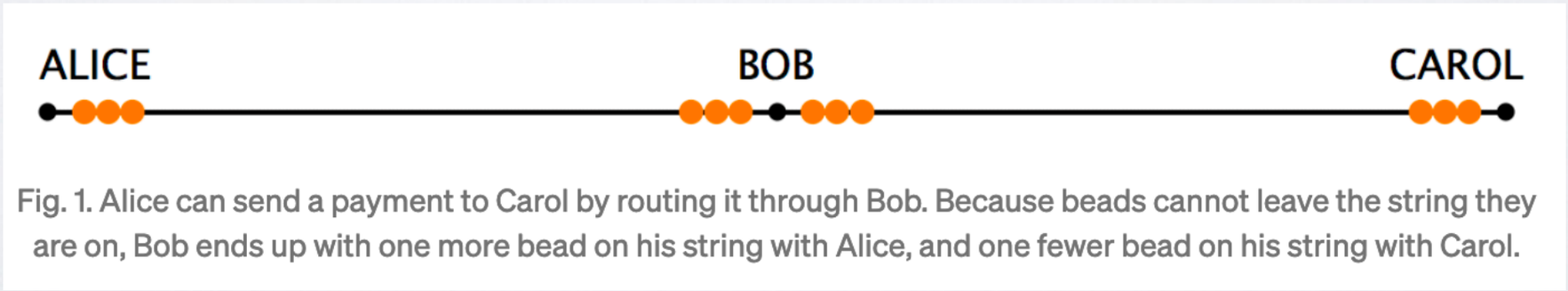


Fig. 1. Alice can send a payment to Carol by routing it through Bob. Because beads cannot leave the string they are on, Bob ends up with one more bead on his string with Alice, and one fewer bead on his string with Carol.

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

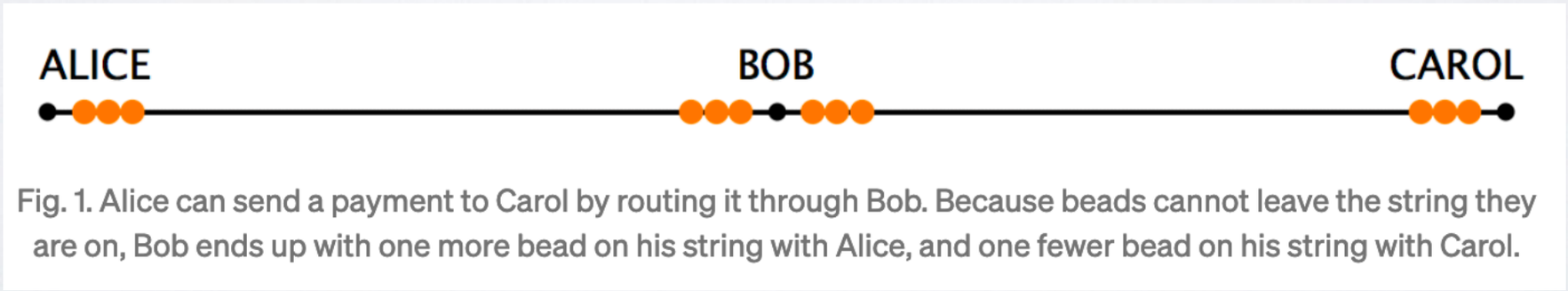


Fig. 1. Alice can send a payment to Carol by routing it through Bob. Because beads cannot leave the string they are on, Bob ends up with one more bead on his string with Alice, and one fewer bead on his string with Carol.

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

Q: Can we move this idea to Ethereum?

- Why or why not?

Ethereum state channels

- In principle we can replicate Bitcoin's LN payment channels on Ethereum. Basic ideas are similar.
  - Initial "funding transaction" that locks up funds between two parties
  - Subsequent "update" transactions that change the balance of funds between two parties (not posted to chain)
  - Final "closure" transaction that goes on-chain + a dispute resolution procedure

Ethereum state channels cont'd

- The problem is that this idea works only for specific cases where two users are affected by misbehavior:
  - Imagine A & B have a state channel, and they're both colluding to do bad things
  - Can this hurt C?

Ethereum state channels cont'd

- The problem is that this idea works only for specific cases where two users are affected by misbehavior:
  - Imagine A & B have a state channel, and they're both colluding to do bad things while off-chain
  - Can this hurt C?
  - In Bitcoin payment channels, the answer is: **NO**. Payment channels only adjust the balance between A, B. Since C isn't involved, they have no "skin in the game".
  - But ETH contract state might matter to other people!!

Ethereum state channels cont'd

- Let's assume that contract state updates (transactions) matter to many parties, i.e., not just Alice and Bob
  - Can we think of an example of such a contract?

Ethereum state channels cont'd

- Let's assume that contract state updates (transactions) matter to many parties, i.e., not just Alice and Bob
  - Can we think of an example of such a contract?
- Let's say we want to do a sequence "off chain" executions (transactions) of the smart contract, and not put them all on the chain to be verified by consensus nodes
  - How could we do this?

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!**