Lightning Network

Layer 2 scaling channels

Thaddeus Dryja <tdryja@media.mit.edu>

Dev++ 2017
2017-11-03

Blockchain scalability

- How can we allow many millions of people to use bitcoin?
- While preserving the important properties of the system?

Scalability Problem

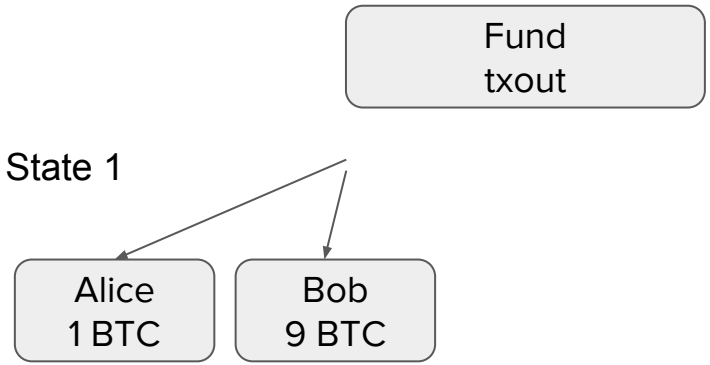
- Every node verifies every message
- The whole world on a single wifi AP
- Everyone needs to agree, but that's super costly

Brute force scaling

- Crank up the message rate
- Can get 2x, 4x speedup
- like 802.11g ➔ n ➔ ac
- Really, gotta split up the network
- But how to maintain consensus
- Can enable other applications

(Micropayments)

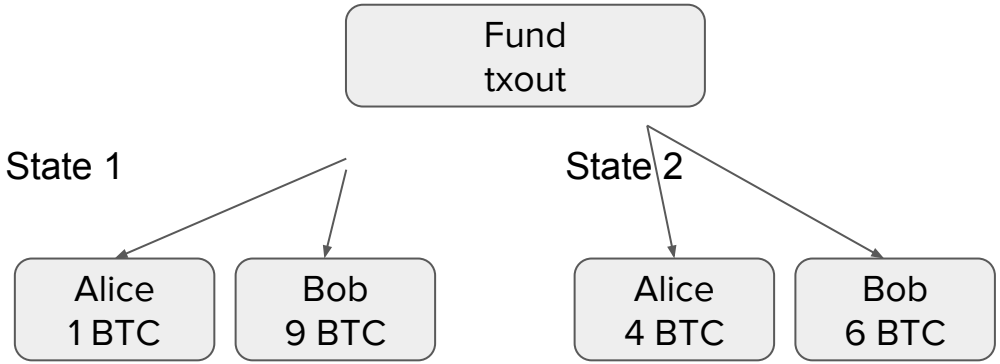
Payment channels



Alice & Bob can transact without sending any data to the blockchain.
While the channel is open, Alice can reduce her balance and increase Bob's as many times as she wants. (Bob can do the same)

At any time, either party can close the channel, without interacting with the other party. The most recent channel state (balances) are then confirmed.

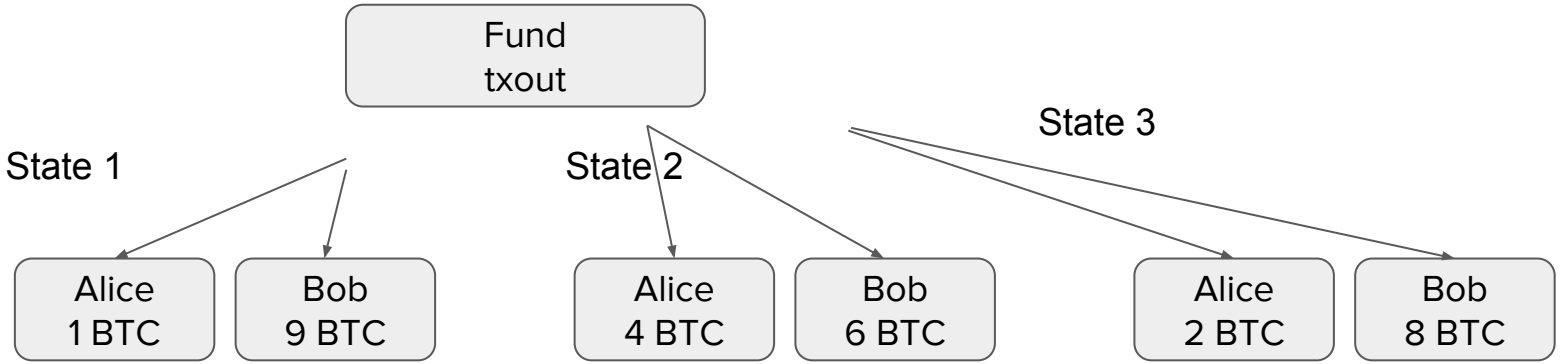
Payment channels



Alice & Bob can transact without sending any data to the blockchain.
While the channel is open, Alice can reduce her balance and increase Bob's as many times as she wants. (Bob can do the same)

At any time, either party can close the channel, without interacting with the other party. The most recent channel state (balances) are then confirmed.

Payment channels



Alice & Bob can transact without sending any data to the blockchain.
While the channel is open, Alice can reduce her balance and increase Bob's as many times as she wants. (Bob can do the same)

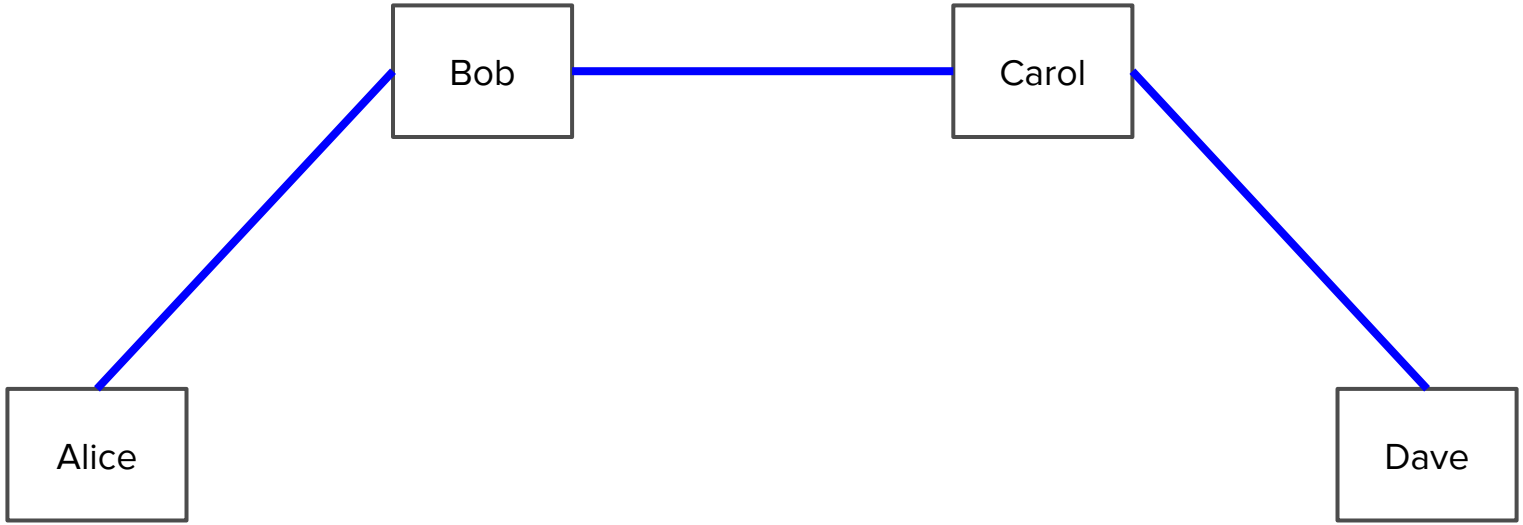
At any time, either party can close the channel, without interacting with the other party. The most recent channel state (balances) are then confirmed.

Multi-channel network

- A single channel has 2 participants. 2 txs are needed per channel (open, close)
- Like opening an account; useful
- Sometimes don't want an account, want 1-off
- Make a network of channels

Multi-hop via channel networks

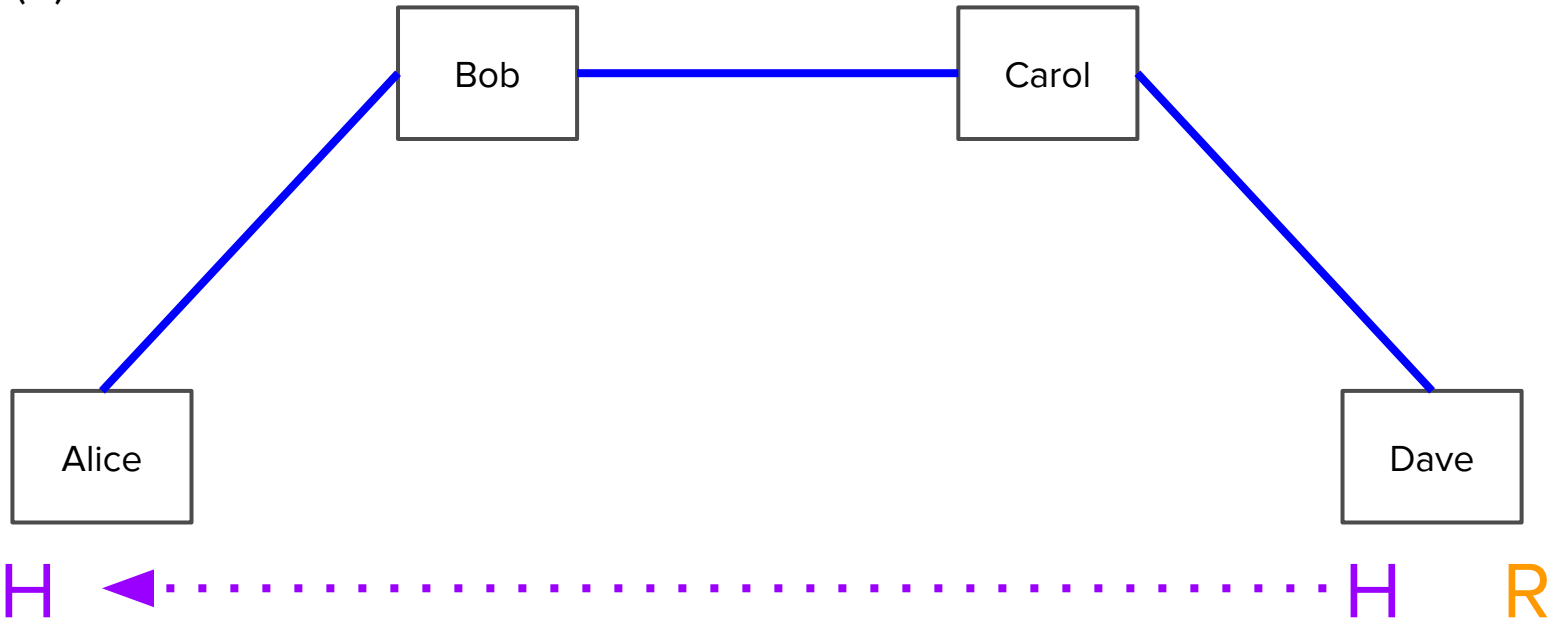
Alice wants to send coins to Dave. But she doesn't want to open a new channel with him.



Multi-hop via channel networks

First, Dave comes up with a random number R.

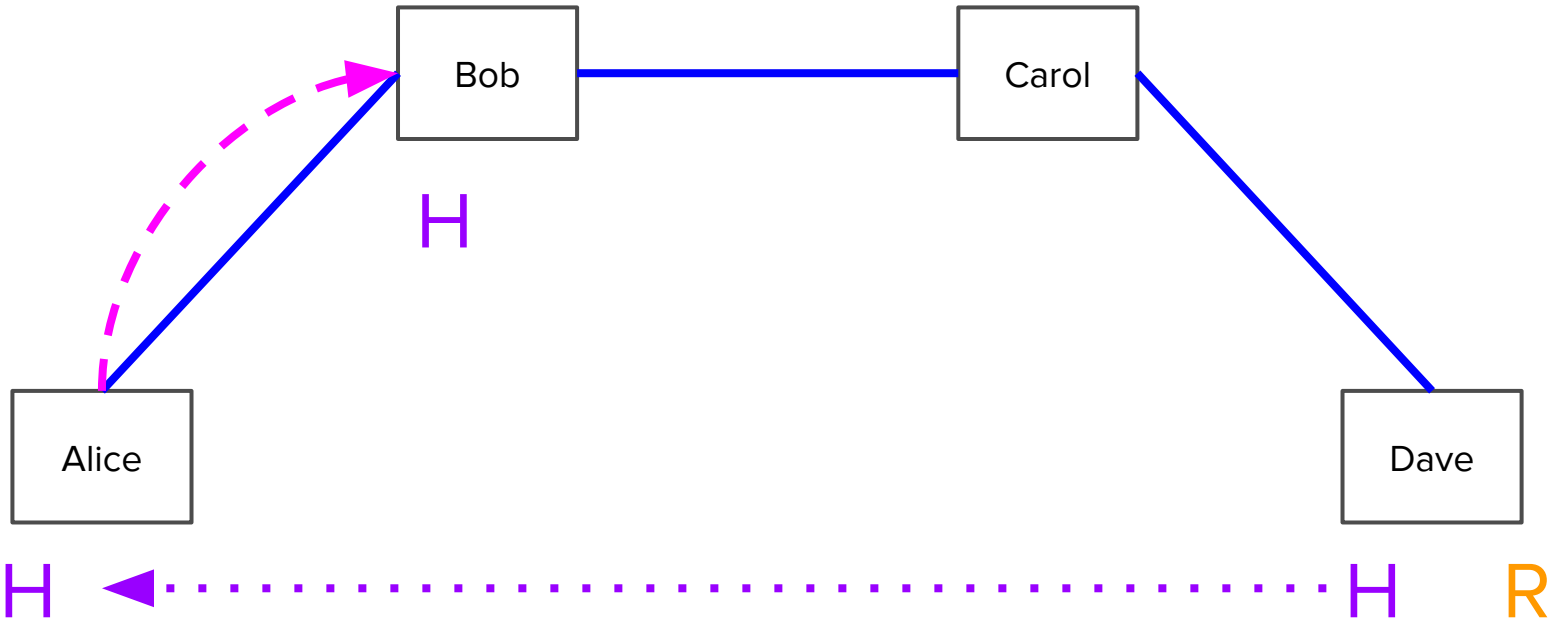
Hash(R) we'll call H. Dave sends H to Alice.



Multi-hop via channel networks

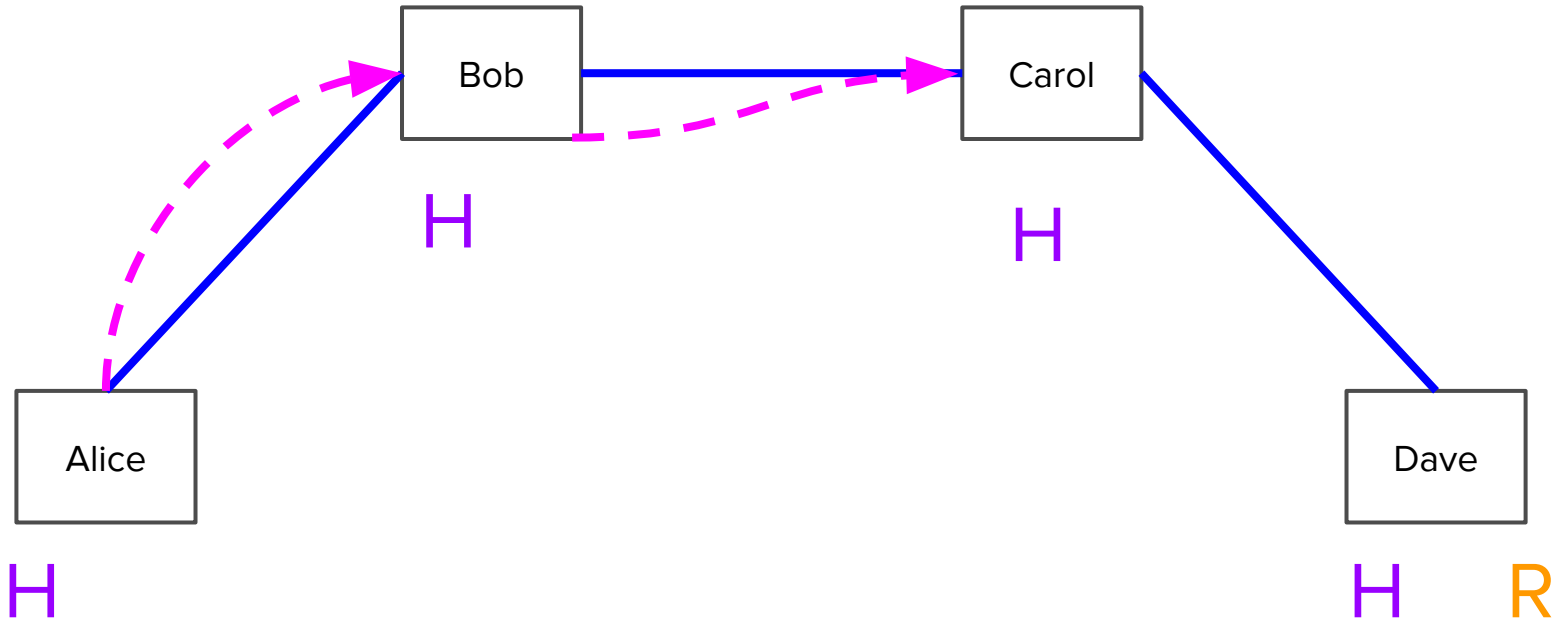
Alice pays Bob a coin. But Bob needs to know the preimage of H to spend it.

So Bob gets money, contingent on knowledge of R.



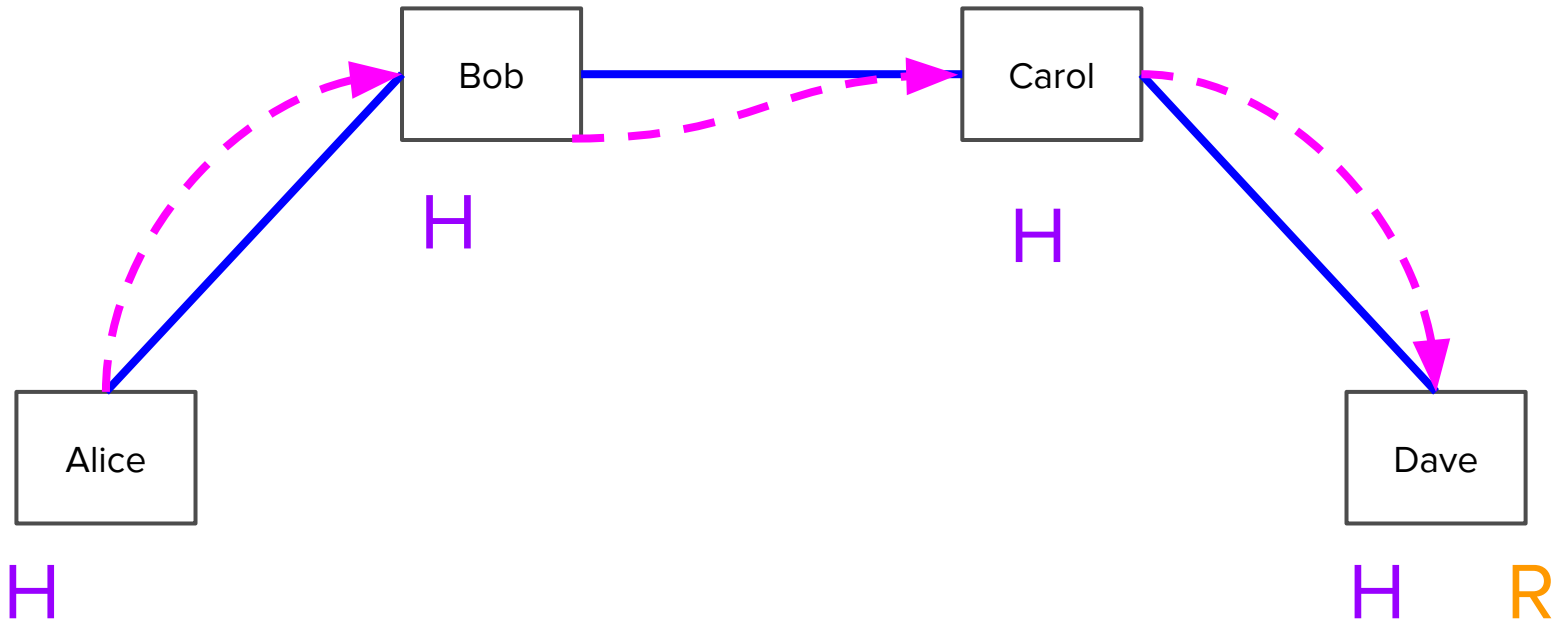
Multi-hop via channel networks

Similarly, Bob pays Carol, but Carol needs to know R to spend.



Multi-hop via channel networks

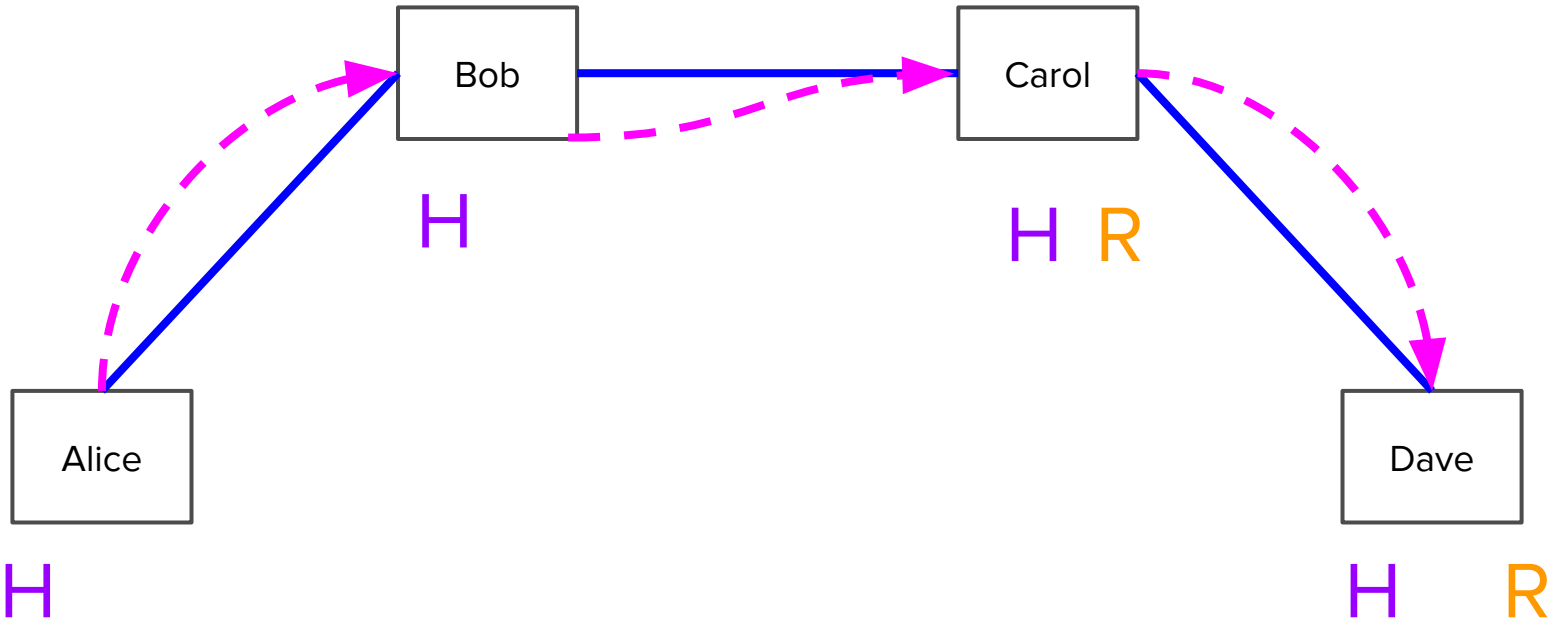
Carol pays Dave the same way. Dave knows R as he made it up.



Multi-hop via channel networks

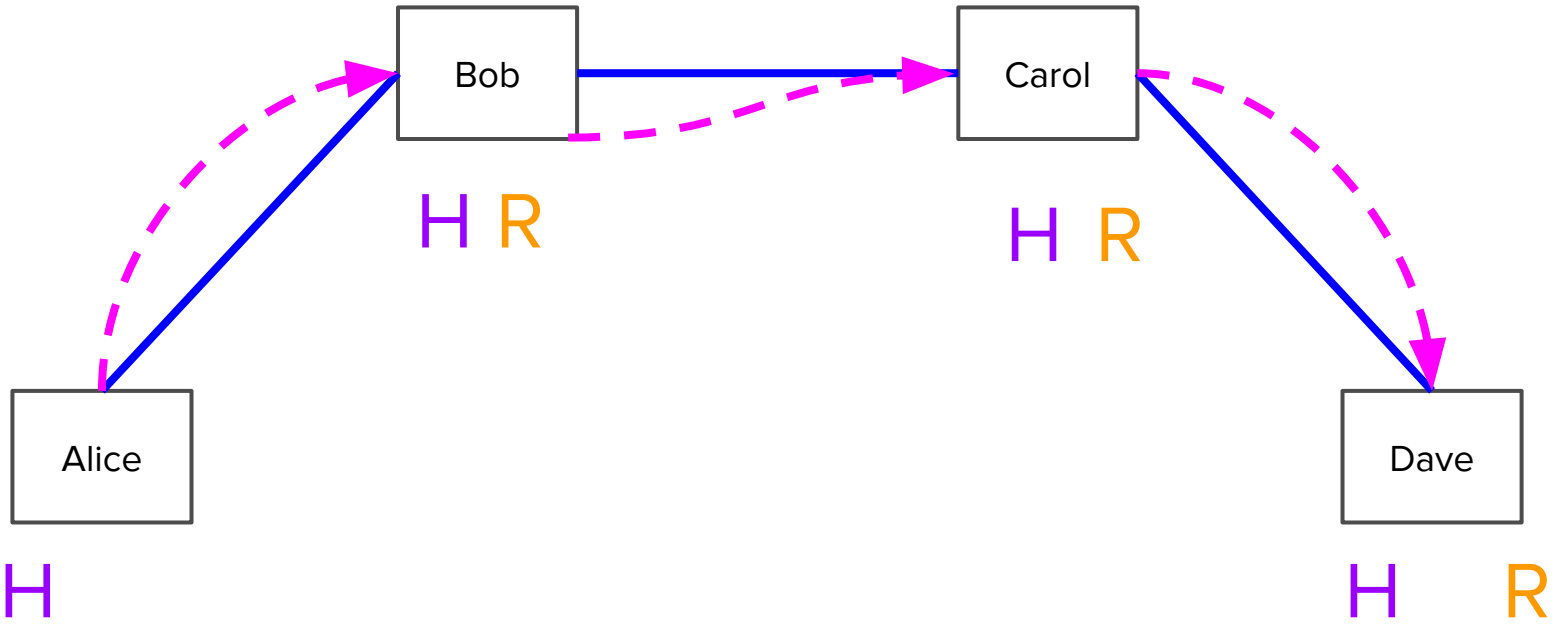
Dave could broadcast on-chain and grab the money, revealing R.

Instead he tells Carol R, making the whole "contingent" thing becomes moot.



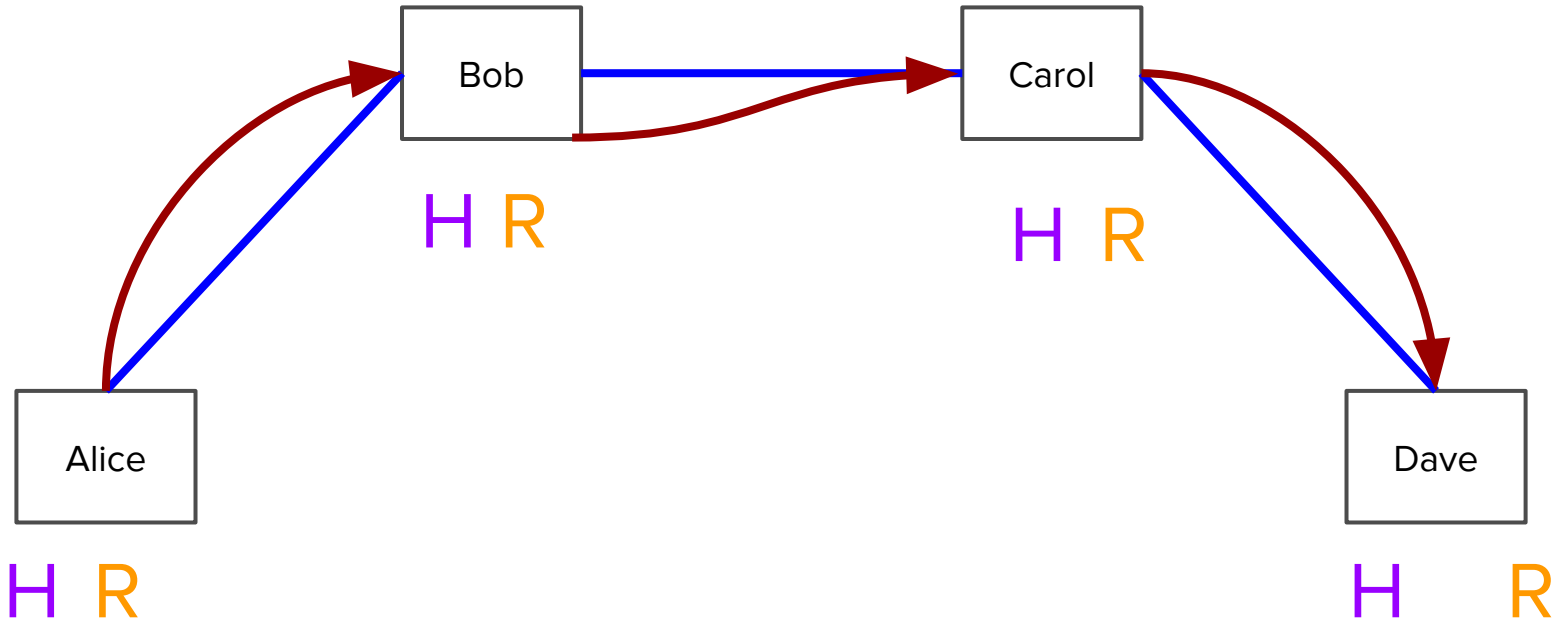
Multi-hop via channel networks

Carol reveals R to Bob, making that payment certain as well.



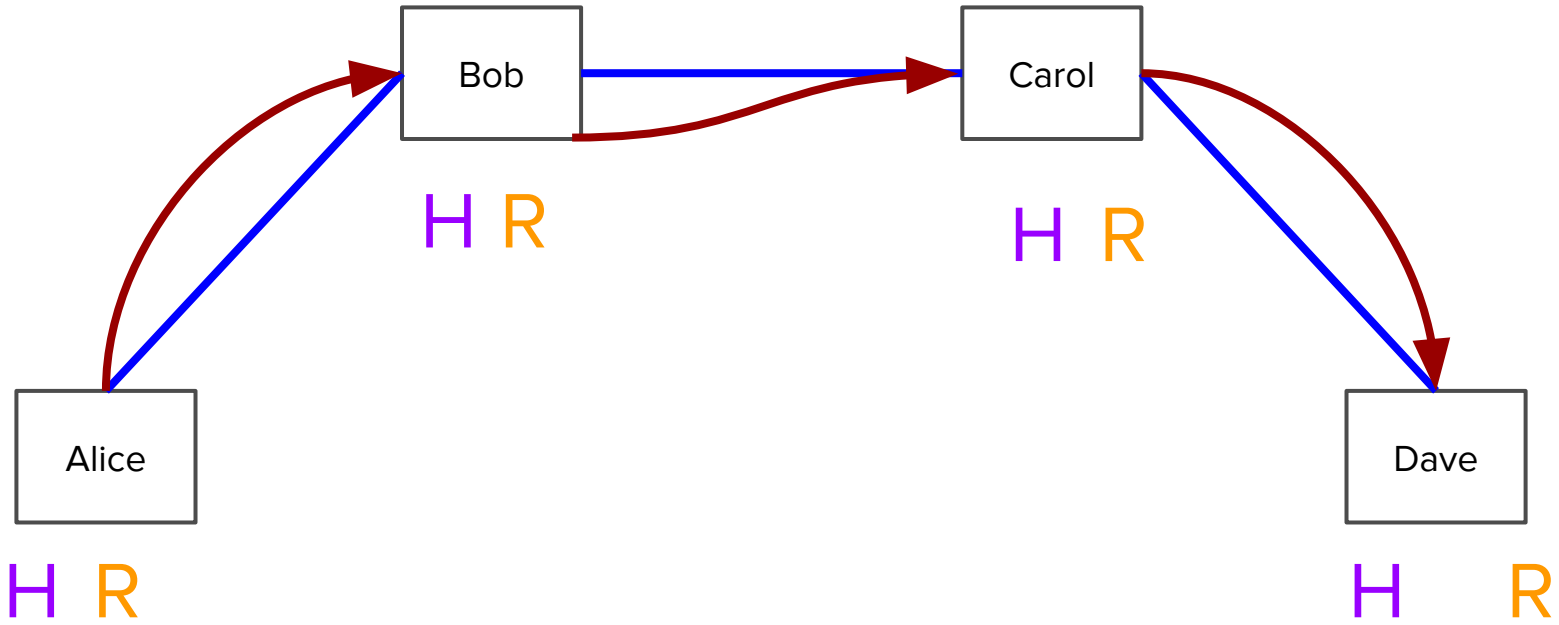
Multi-hop via channel networks

Alice learns R from Bob, which is a receipt confirming the payment went through



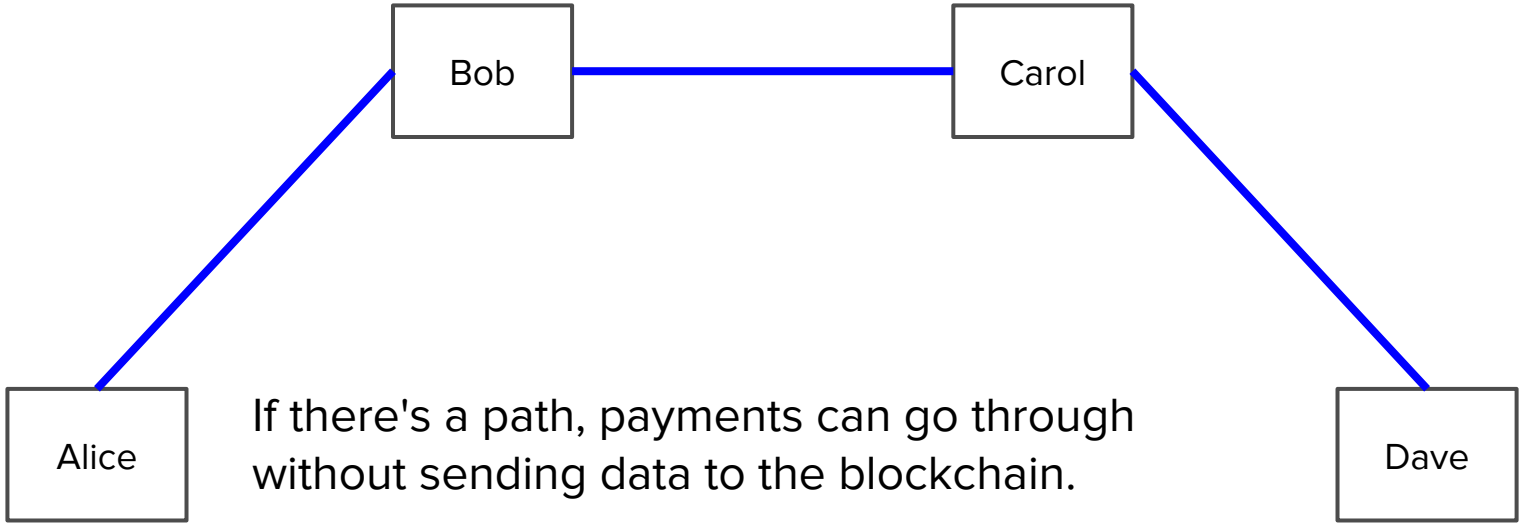
Multi-hop via channel networks

All the contingent payments can be removed and balances updated; everyone knows that everyone else knows R.



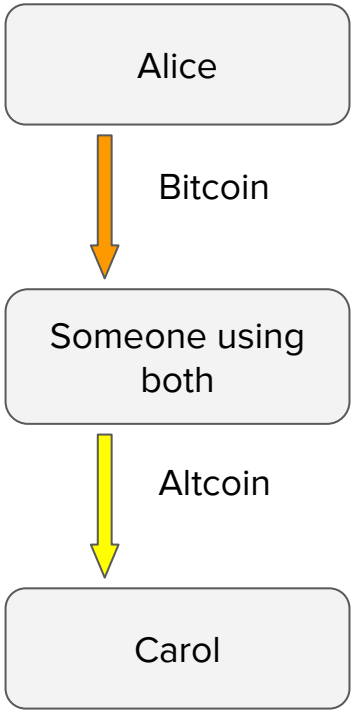
Multi-hop via channel networks

Multi hop payments can thus be made without trusting intermediate nodes



If there's a path, payments can go through without sending data to the blockchain.

Multi hop over different channels



This works even when the channels between A-B and B-C are on different blockchains.

(More on that in Ethan's talk)

Lightning Network utility

- Faster; limited by latency
- When people are chill, cheaper. When they fight, falls back to blockchain.
- Everyone makes contracts, but few go to court.
- Think of the blockchain like a global court; make agreements enforcable there

Implementation: lit

- There's others too in other languages
- Mine is lit, in golang, uses btcd libs, seems to run ok on linux/mac/win
- Modular; Includes SPV wallet
- Standalone, easy to use / save (goals)
- Uses / requires segwit
- No multihop just yet
- Multi coins supported

ご注意

- Lit　は完全出来てません、バッグまだいっぱいあります
- 今回は初めの多く人が使ってる。　新しいバッグ発見しましょう！
- 発見したら、ちょっと書いたら嬉しいです
- Githubにissuesも日本語OK

SPV

- SPV is nice! Way less data
- 150 GB down to 20 MB or so
- Big risks though!

Full node wallet operation

- Download & verify every block
- verify every tx
- Look for your addresses in output scripts
  - If you find it, add that utxo to your wallet (good!)
- Look for your utxos in inputs
  - if you find it, delete that utxo from the wallet (bad :( )

SPV wallet operation

- Check all the 80 byte headers
- Build a bloom filter, send to a full node
- Get back txs that match the filter along with every block
- Download a couple megs, get all your txs.

Problems with SPV

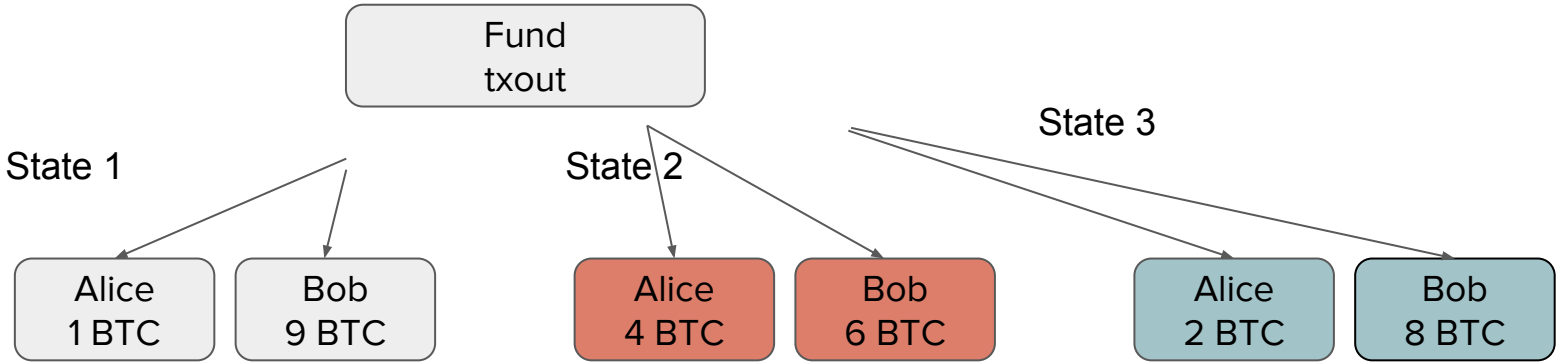
- Don't enforce network rules
  - 100 coins per block? sure!
- Can't check signatures
  - Don't have prev inputs, don't know key hashes
- Full nodes learn a lot
  - all your addresses, utxos
- Unconfirmed (mempool) txs are totally unsupported (but wallets still show em!)

Stronger SPV

- Download everything, just don't store
- Committed block filters

- Connect to explicitly trusted nodes

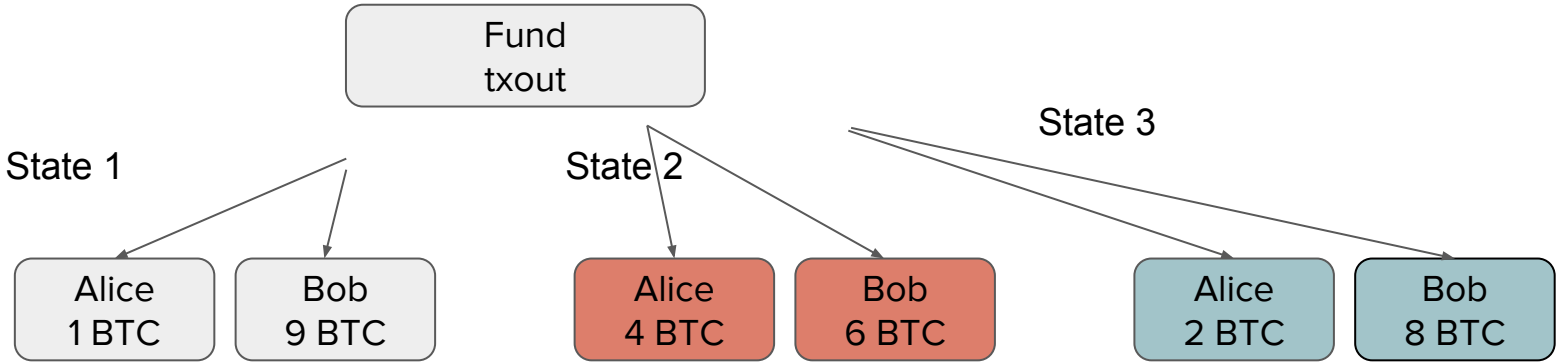
Payment channel details



How to erase history?

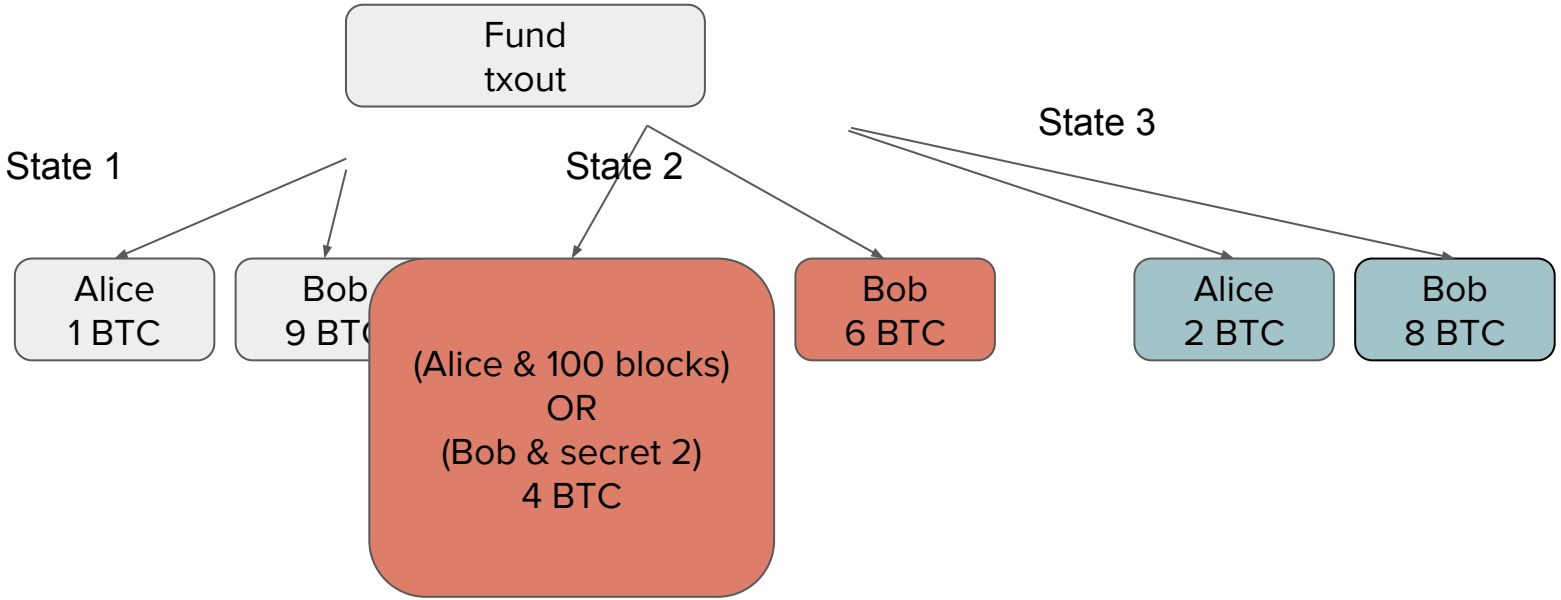
Delete state 2... but can't prove deletion!

Payment channel details



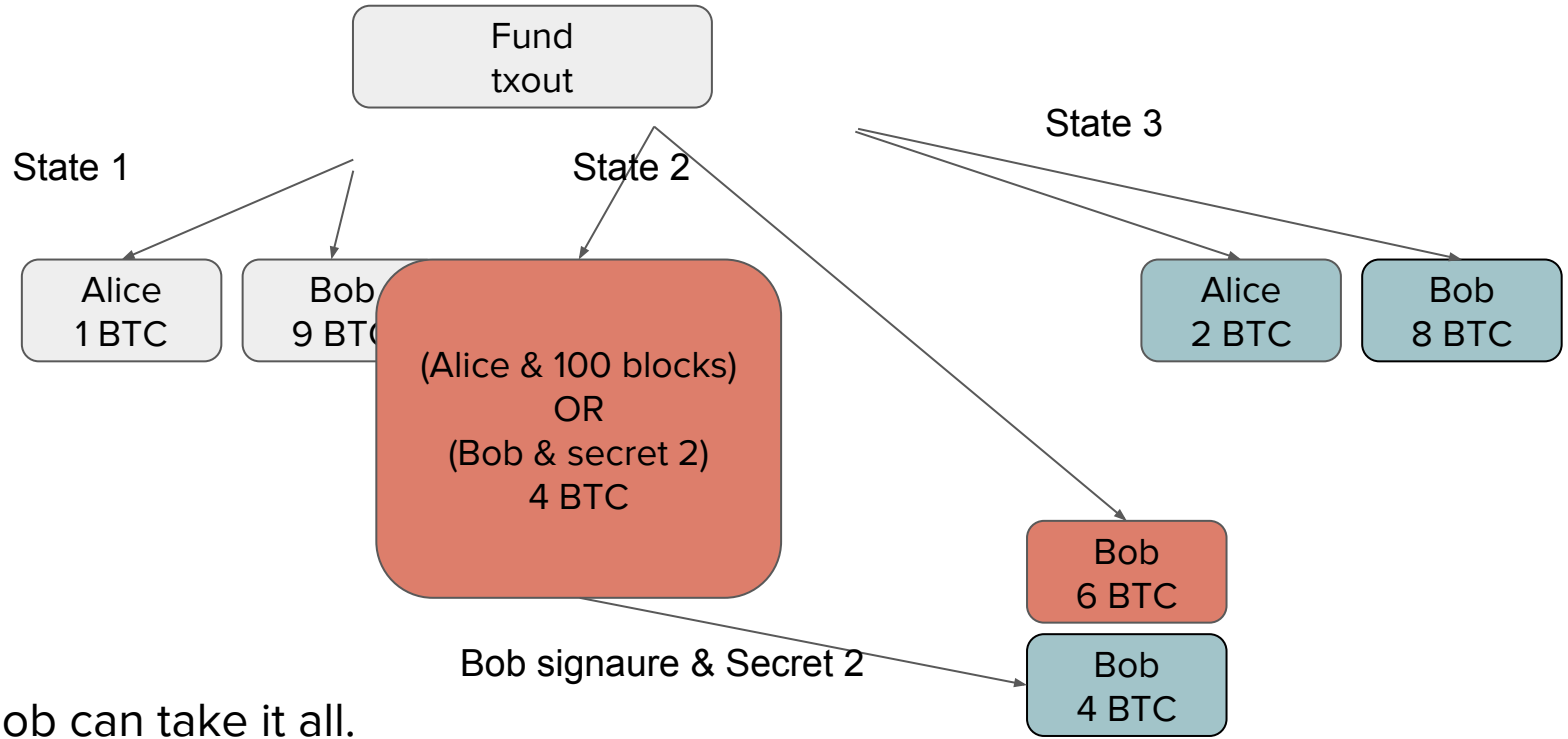
If you broadcast state 2, looks OK to the network; they don't know about state 3.

Delete history: revoke



If Alice broadcasts, she has to wait 100 blocks before spending. With her secret, Bob can spend immediately.

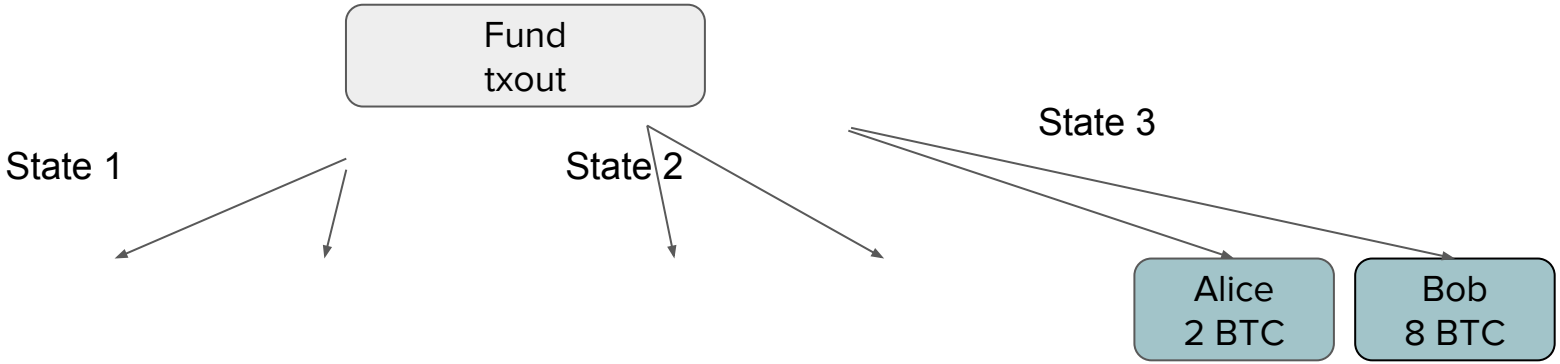
Delete history: revoke



Bob can take it all.

Thanks Alice! Your fine is appreciated!

Payment channel



Broadcast an old state = counterparty takes all the money
Old states are radioactive! Delete em!
Everyone only keeps most recent state, so the channel can close correctly.

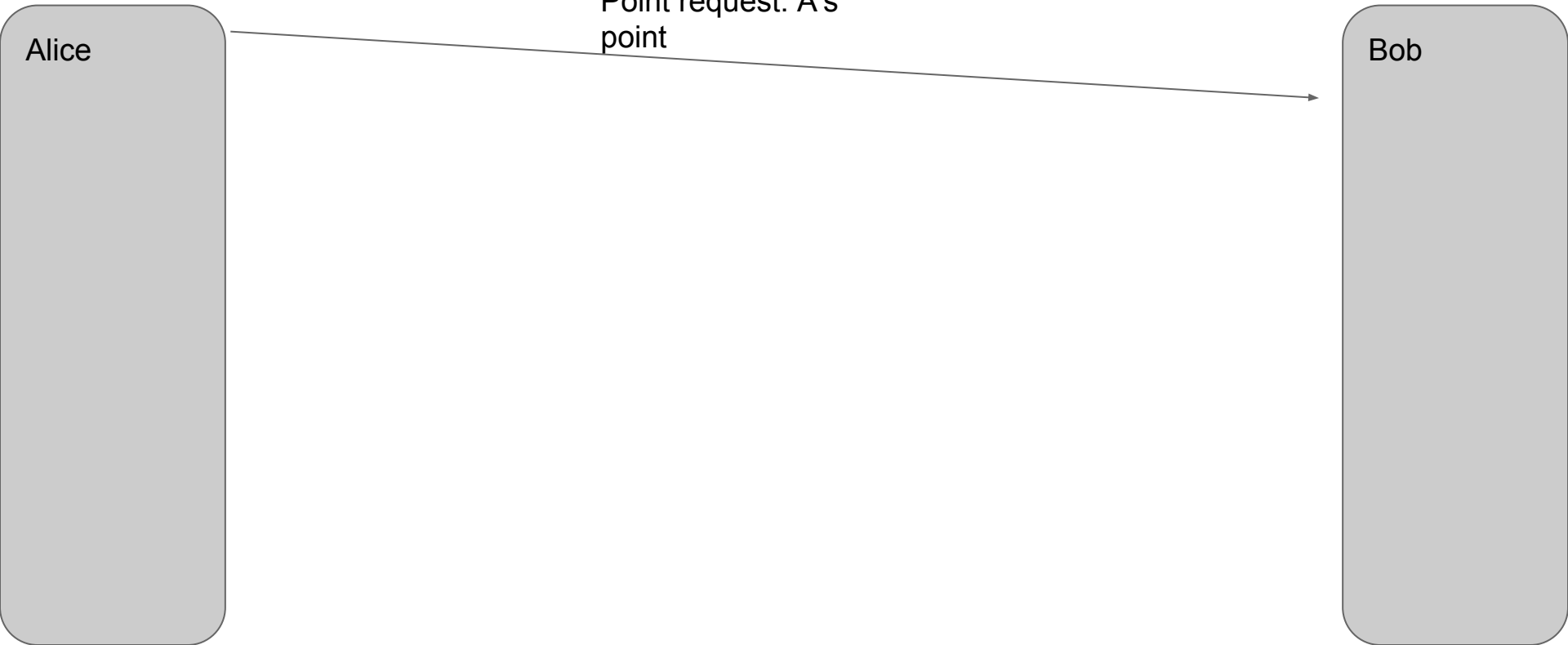
Lit message flow: fund

- Starting a new channel

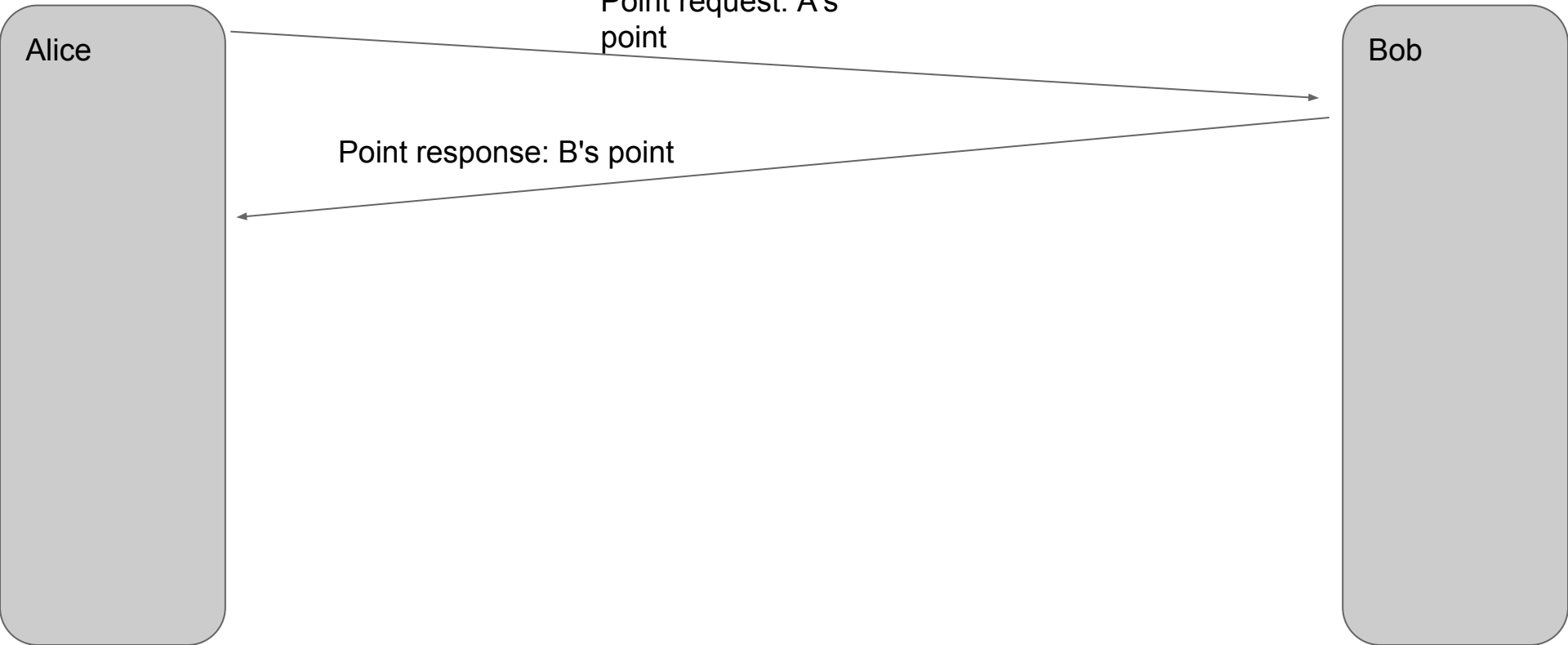
  qln/fund.go

Lit message flow: fund

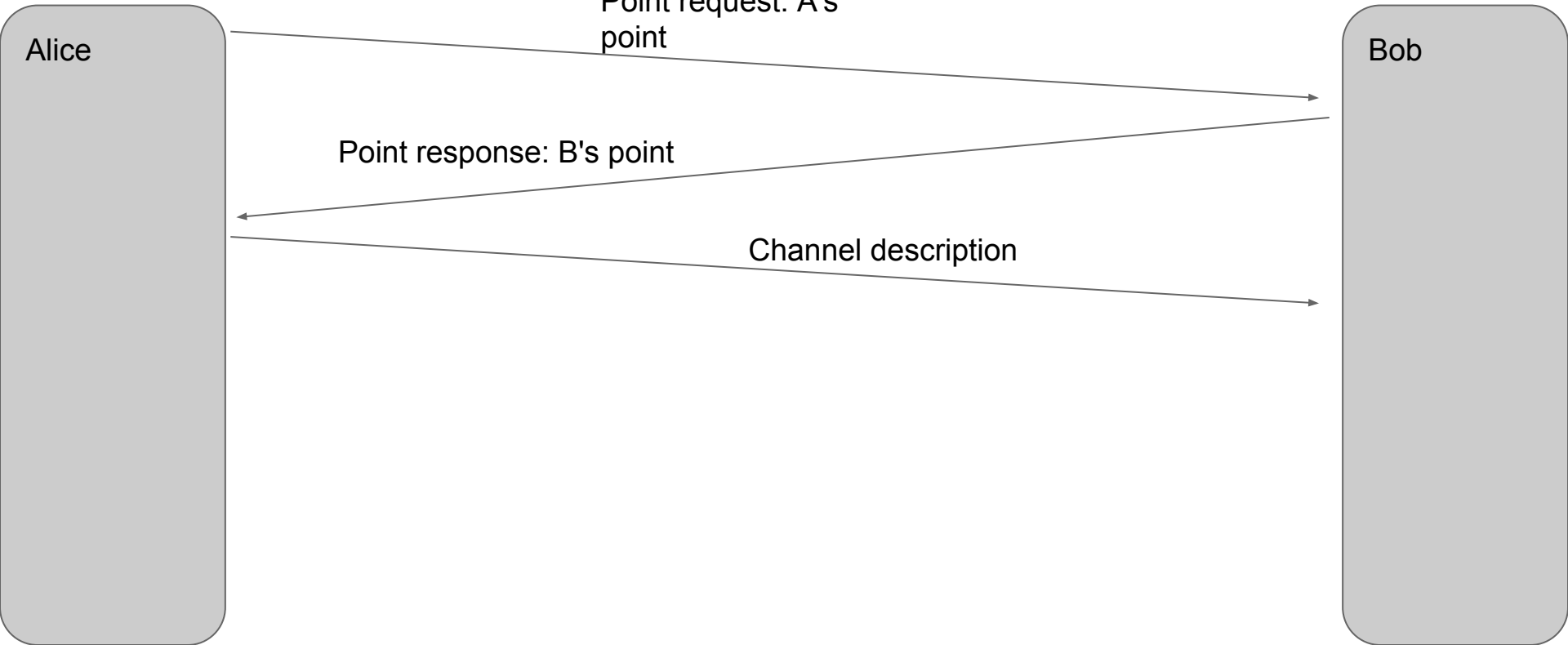
Lit message flow: fund



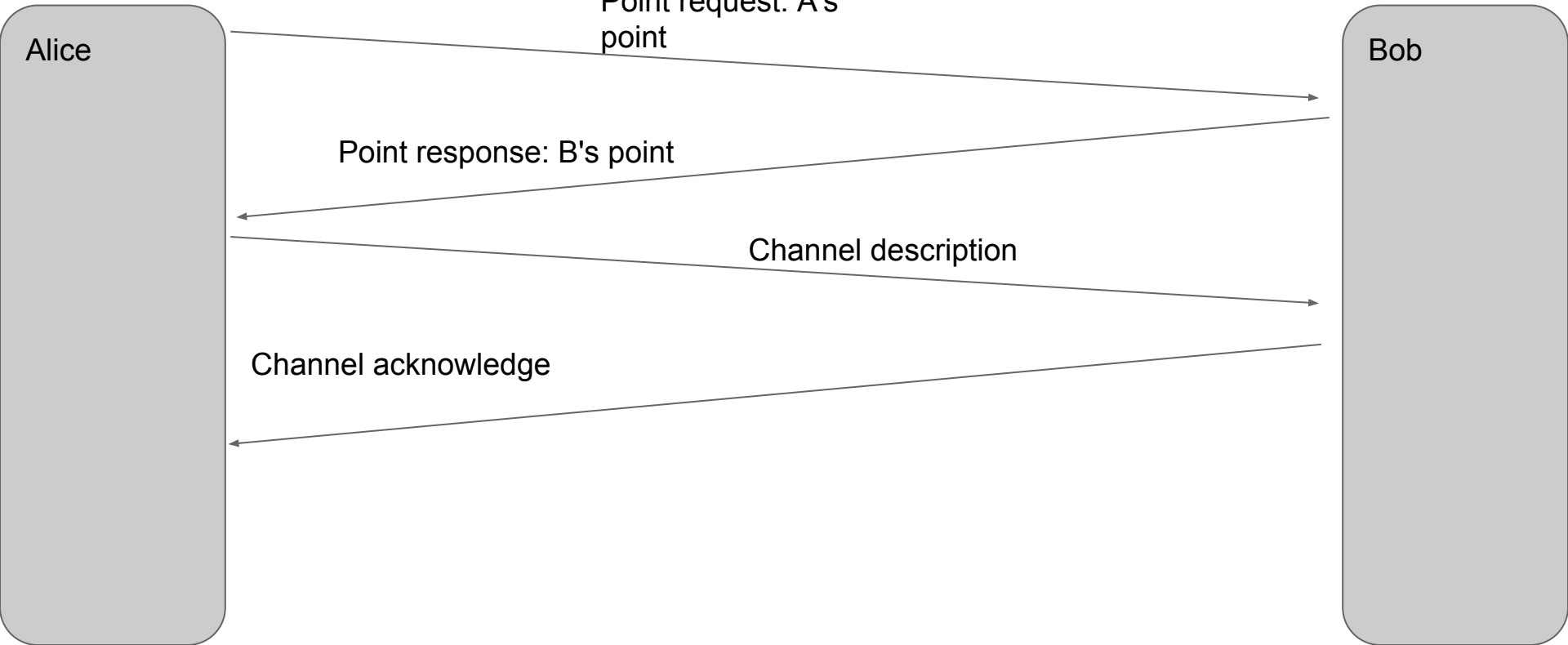
Lit message flow: fund



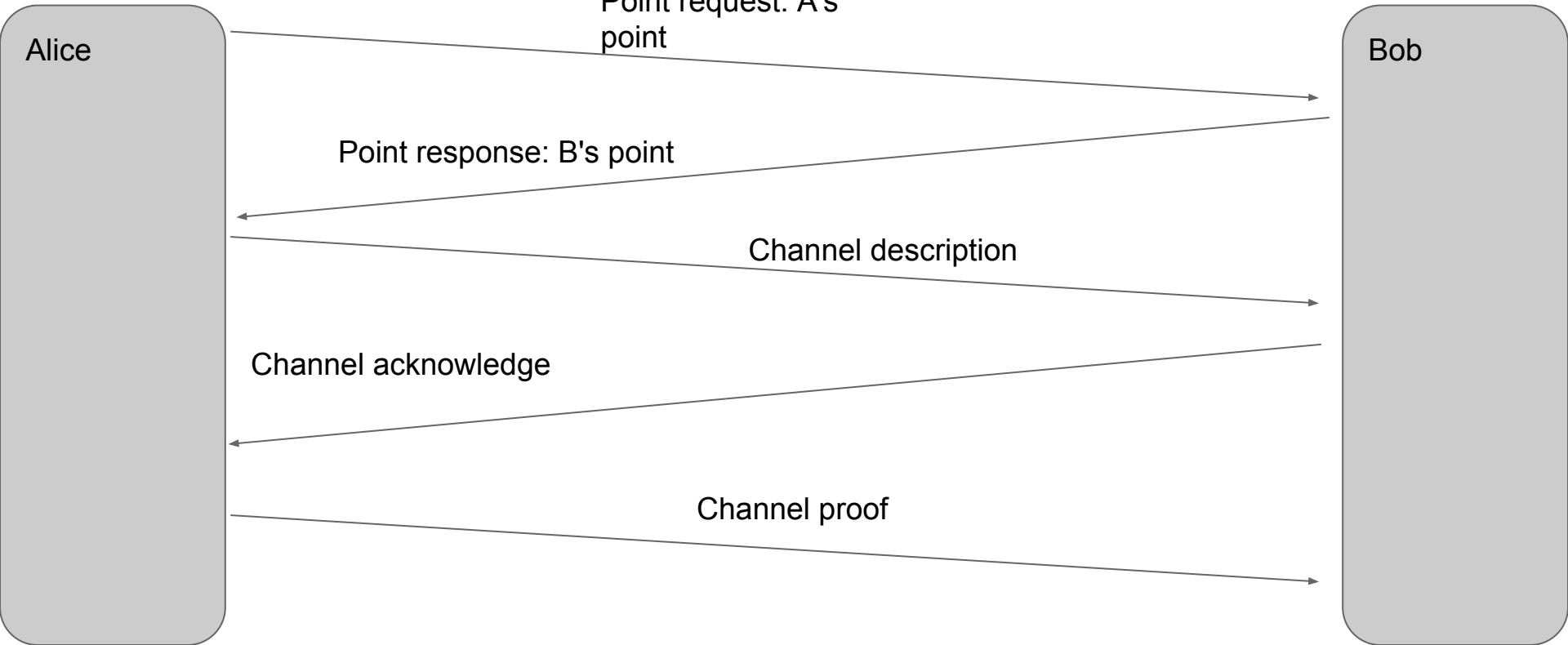
Lit message flow: fund



Lit message flow: fund



Lit message flow: fund



Lit message flow：push

- update channel balance

  qln/pushpull.go

Lit message flow: push

Alice
State: 4
Delta: 0
Amt: 66

Bob
State: 4
Delta: 0
Amt: 25

Lit message flow: push

Alice
State: 4
Delta: 0
Amt: 66

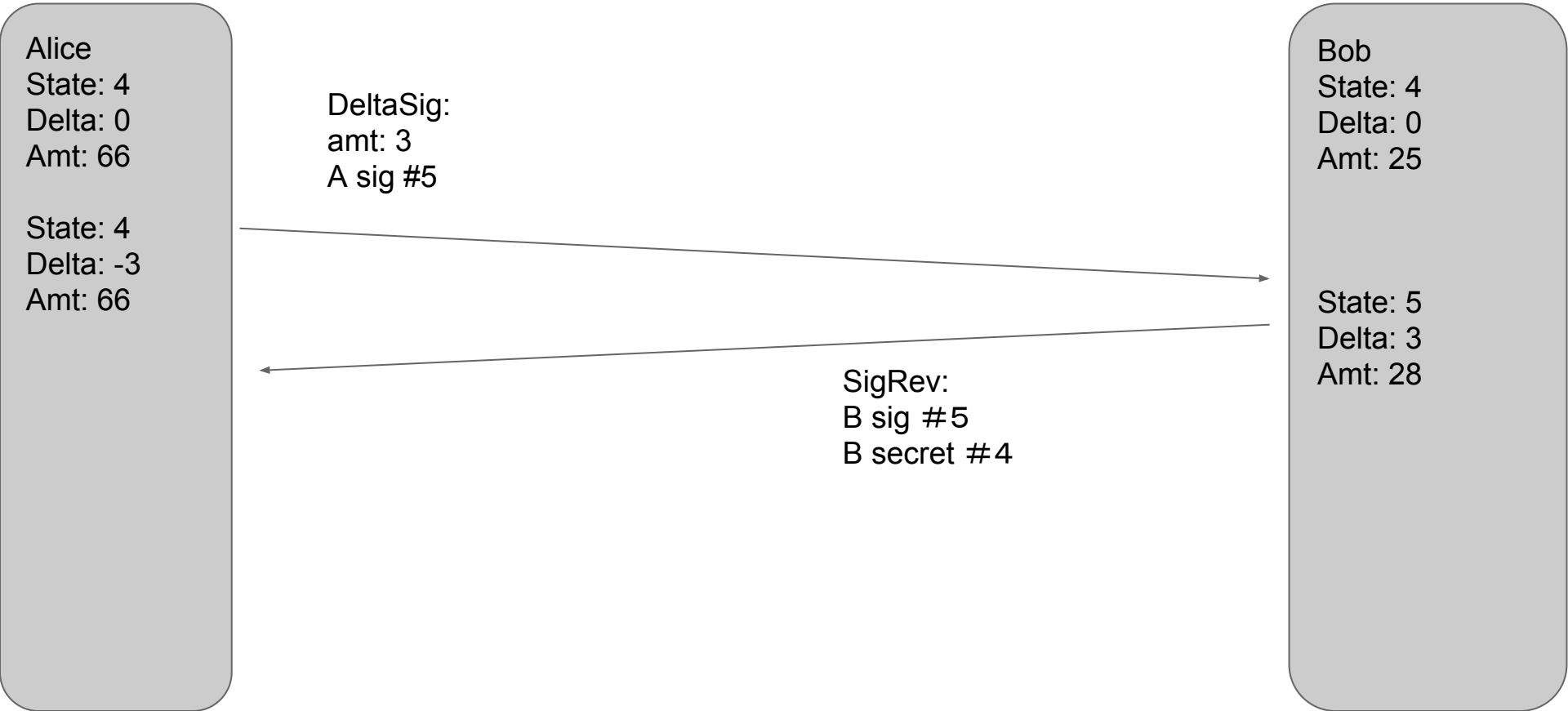
State: 4
Delta: -3
Amt: 66

Bob
State: 4
Delta: 0
Amt: 25

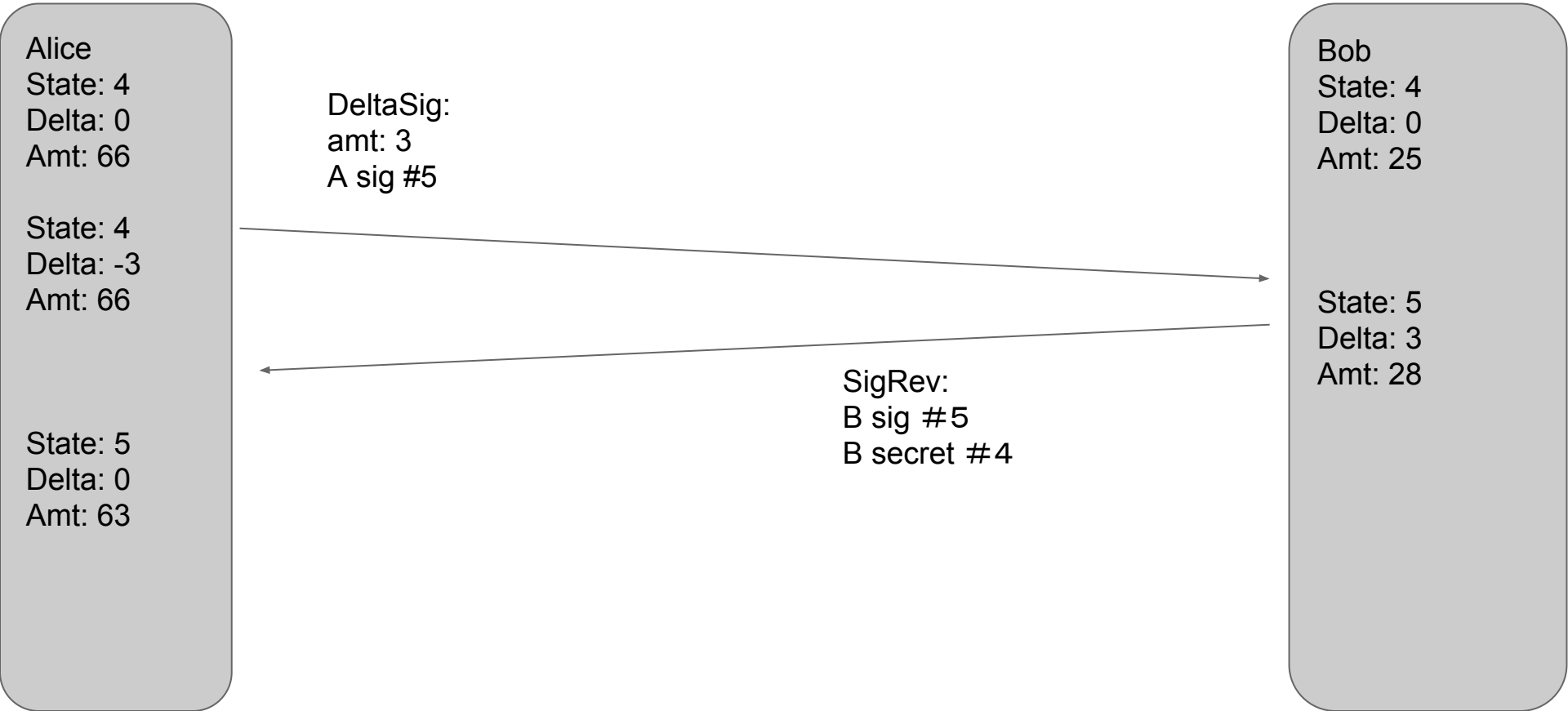
Lit message flow: push

Lit message flow: push

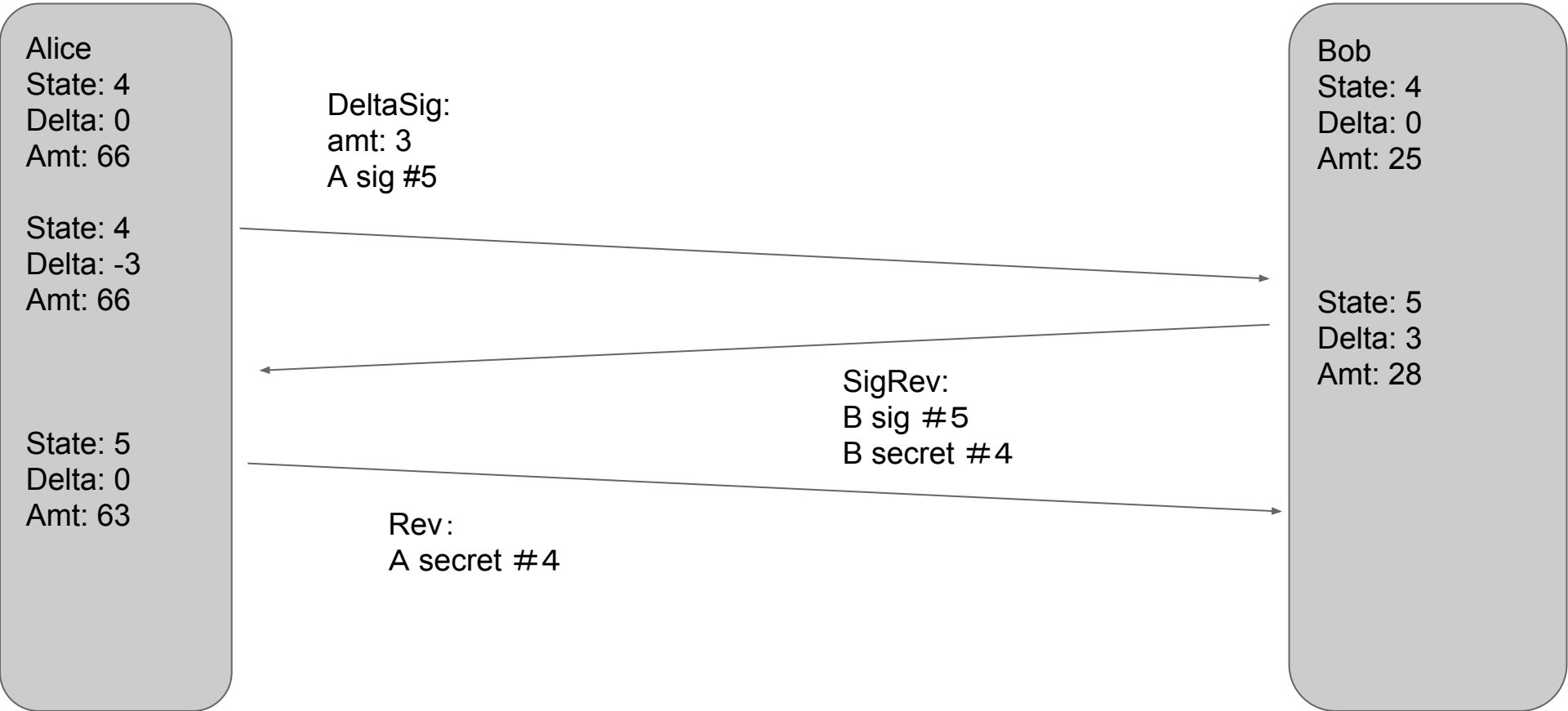
Lit message flow: push



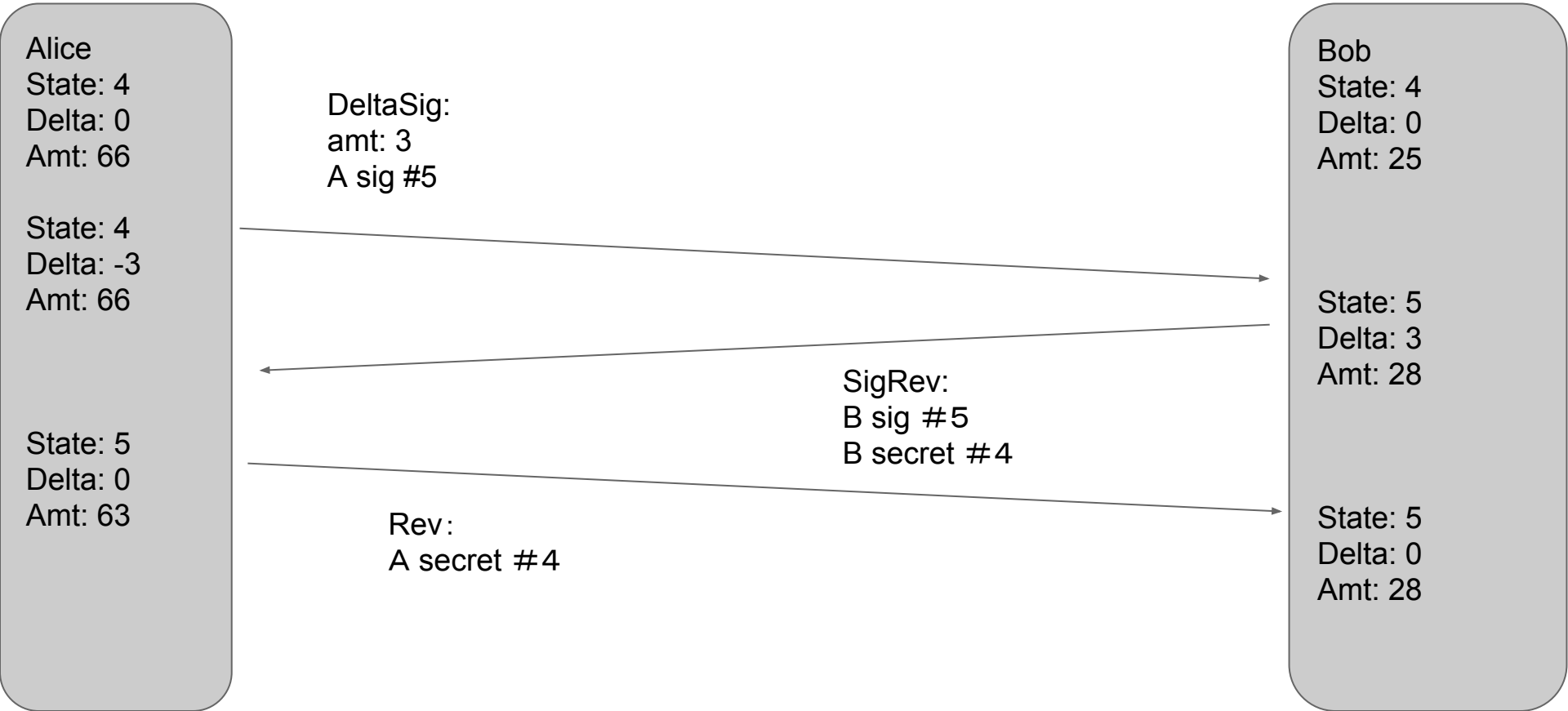
Lit message flow: push



Lit message flow: push



Lit message flow: push



Lit demo

- If we have time!
- 10.21.133.34 is my regtest server
- go get github.com/mit-dci/lit
- or binaries for the casuals:
- http://10.21.133.34:8080
  - litmac for mac, lit.exe for win
  - lit-afmac, lit-af.exe

Lit demo

- `./lit --reg 10.21.133.34 -v`
- `./lit-af`
  - `ls lis con send fund push close break`
- use slack channel for pubkeys / hostnames
- there is a MIT server that is tracking names, might work!

Lit commands

```
ls: same as everywhere :)
send: send on-chain tx
lis: listen on IP addr for incoming
con: connect to addr@host:port
fund: make a channel
push: send money in channel
close / break: end a channel
off: shutdown lit
```