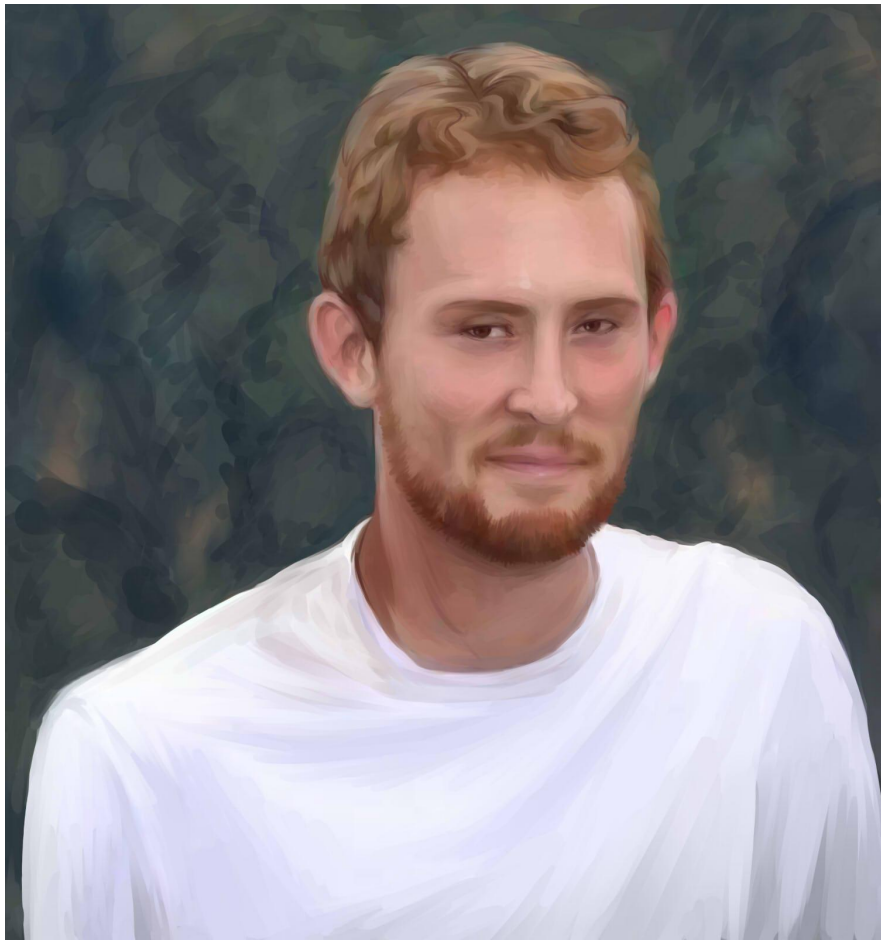




# The Lightning Network Thunderstruck!

# Update!

<https://www.youtube.com/watch?v=HPuRMXGdBjo&feature=youtu.be>



Alex Melville

Software Engineer  
@BitGo

Digital Nomad

[github.com/Melvillian](https://github.com/Melvillian)



# Main Goal: How to use the Lightning Network

- What immediate problems does Lightning solve?
- What will Lightning allow us to do that we can't do now?
- What is Lightning?
- RPC commands to operating *Ind*
- How does Lightning work?
- Future use of Lightning: Neutrino
- Readings for future study

# Transaction Scalability:

Transactions per second (TPS)

Average: 2000  
Peak: 56,000

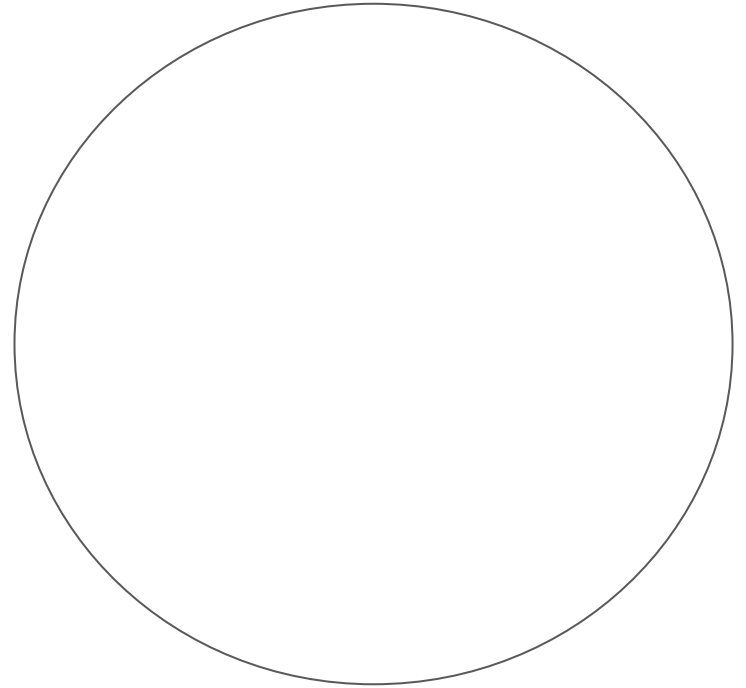
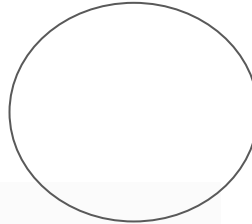
Average: 4  
Peak: 7

Average\*: 8  
Peak: 14

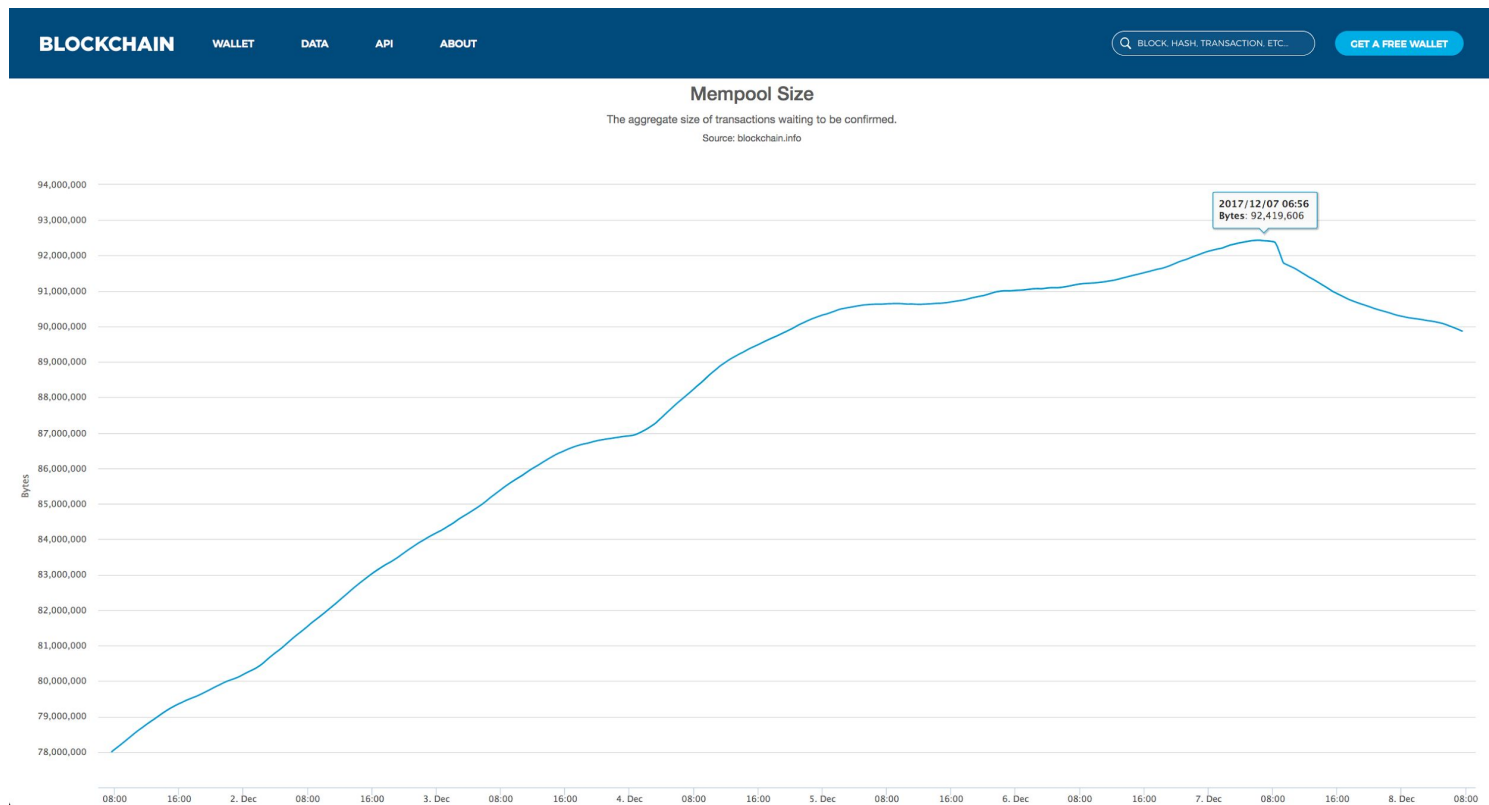
Average: 200  
Peak: 400

○

○

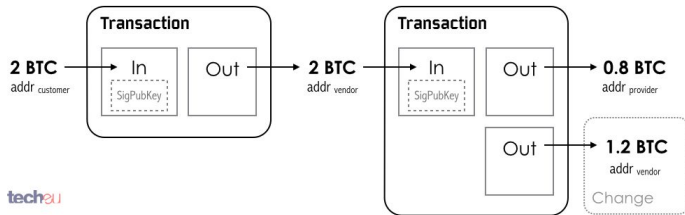


# Transaction Backlog



# How does it scale?

- Removes the need to make on chain transactions for payments.
- User wallets manage a smart contract with several payment nodes
- The contract is fully backed by bitcoin, but can be updated locally between the two nodes
- Contract updates can be chained so any connected node can pay any other.



- **Rarely need to talk to him**
- **Especially if we know 100% what he will decide**

# Light Clients & Contract Enforcement

- Light clients are users of the bitcoin ecosystem that do not download and verify the whole blockchain
- We aspire that light clients do not trust, current wallets work by
  - Comparing headers with difficulty
  - Bloom filters
  - Merkle Trees
- Compact block filter headers
  - Reverses the filter dynamic
- Trust model changes when you are managing payment channel contracts
  - We can outsource channel monitoring



# Poon-Dryja Channel:

Each side of the channel maintains its own version of transactions

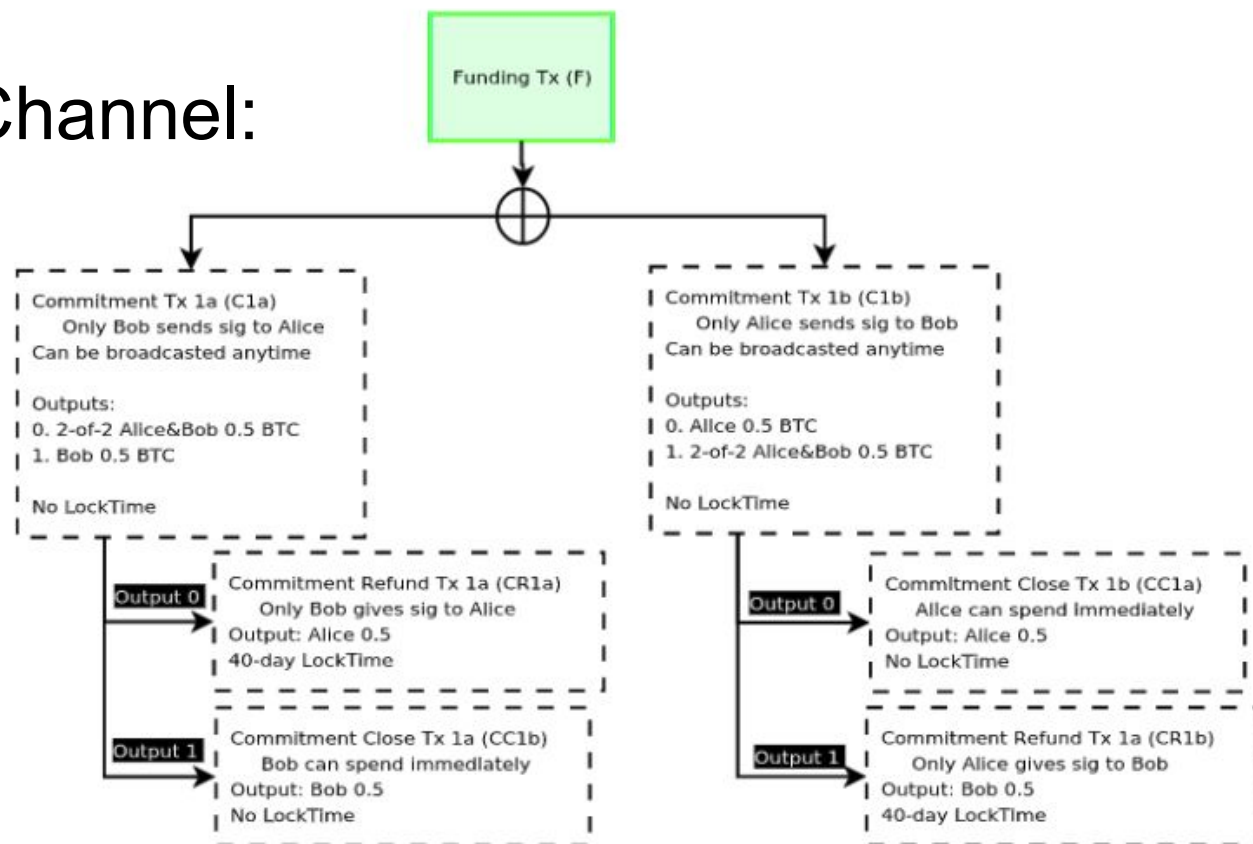


Figure 1: Figure 1 from the Lightning Network Draft 0.5

# Future Improvements from Lightning

Instant Payments to anyone

# Future Improvements from Lightning

Micropayments

# Future Improvements from Lightning

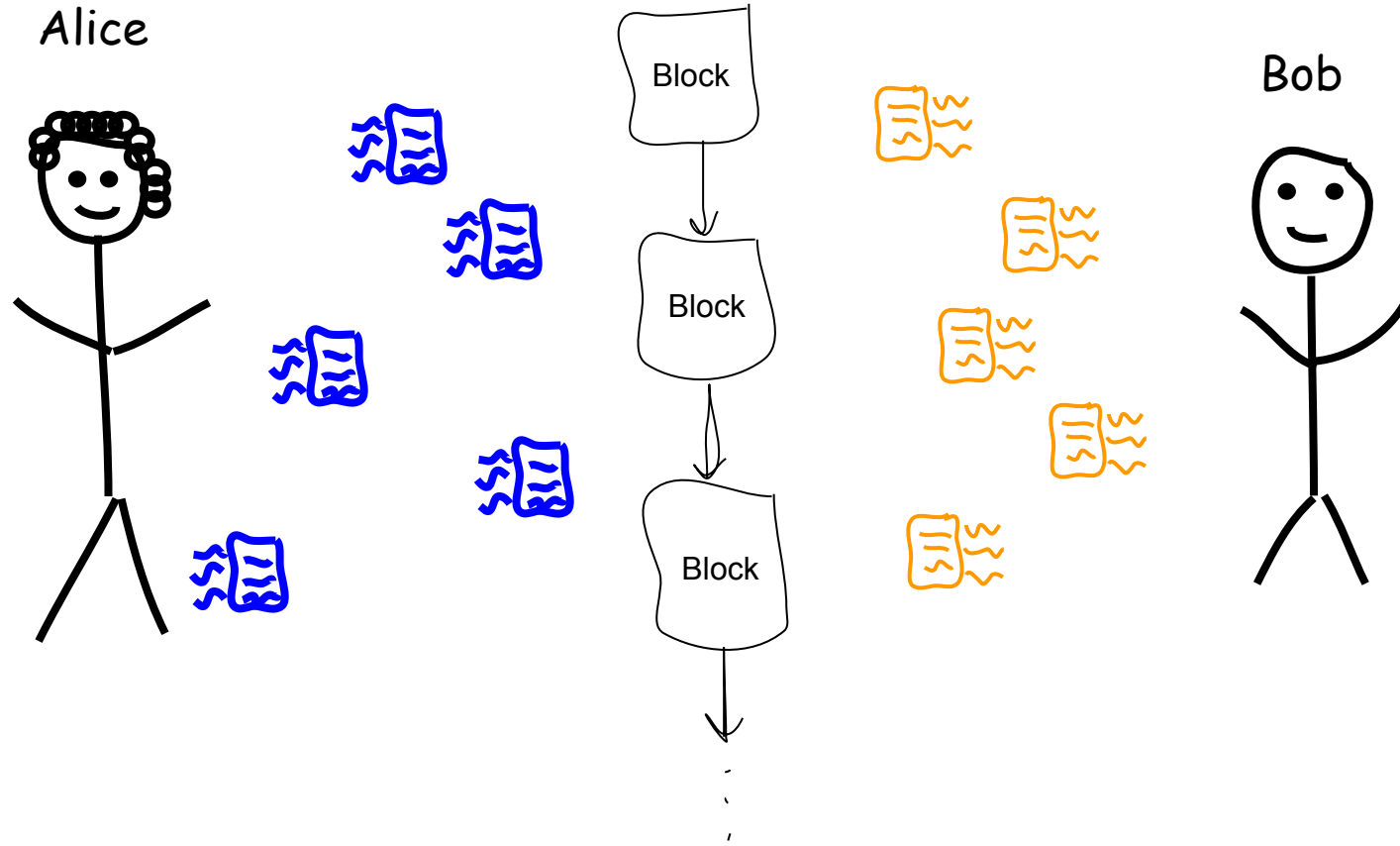
Cross-chain Atomic Swaps

Break...

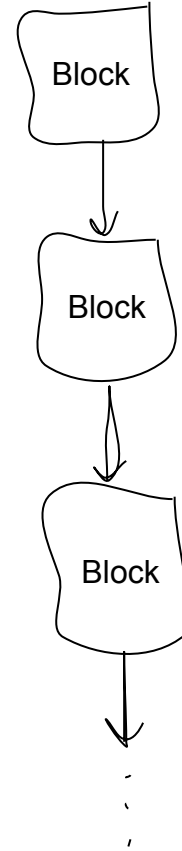
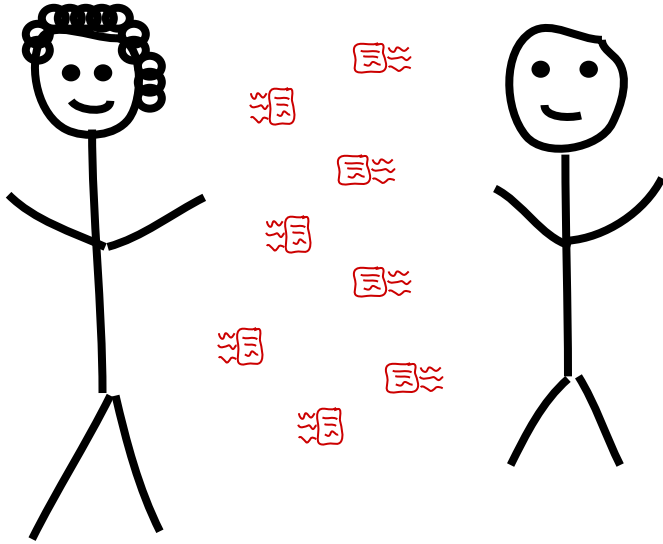
So....

What is Lightning?

# Think of Lightning as a System of Secure IOUs

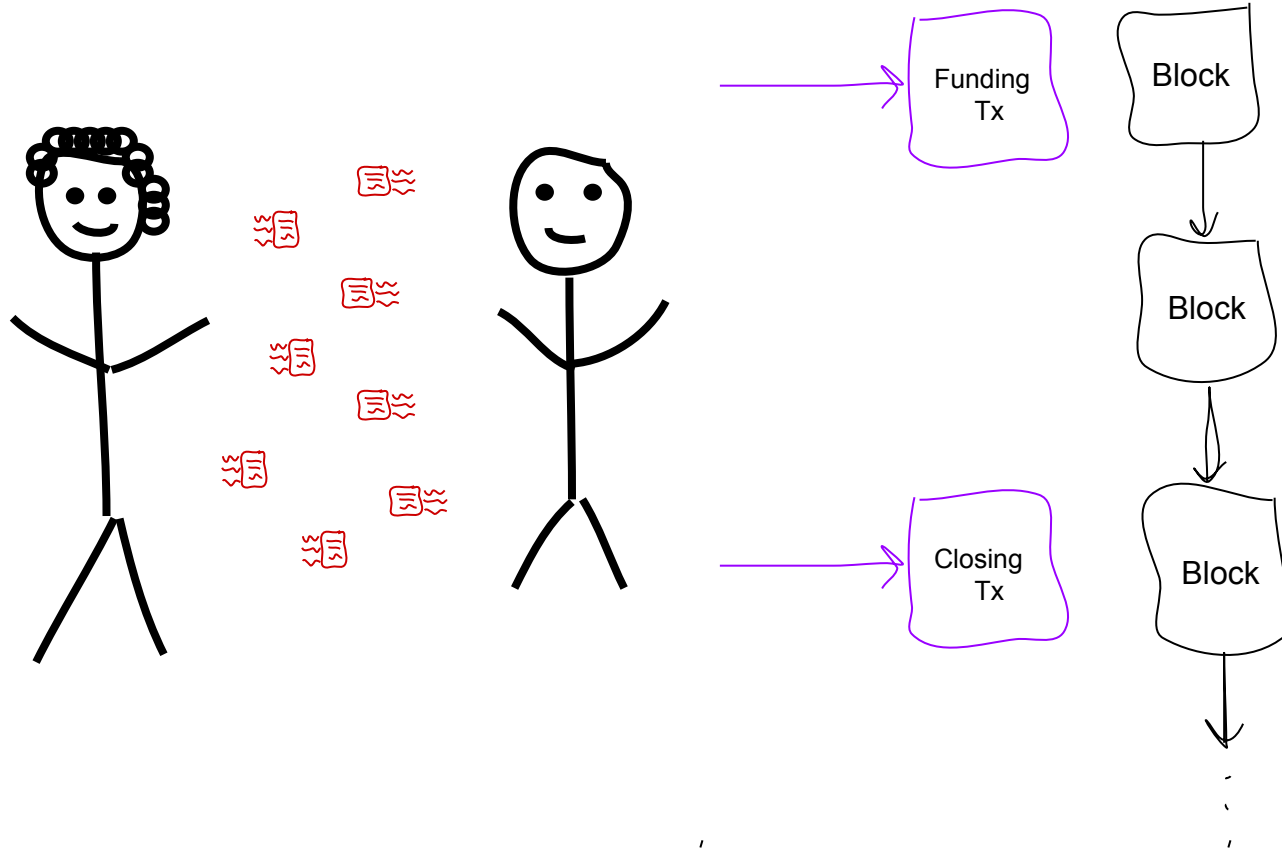


# Think of Lightning as a System of Secure IOUs





# Think of Lightning as a System of Secure IOUs



# What does Lightning Look Like?

## *Incli*

Command-line client used to  
send JSON-RPC requests to  
*Ind* (written in Golang)

## *Ind*

Lightning-aware node which  
handles channel management,  
private/public key storage, and  
peer communication. Used  
*btcd* to interact with blockchain  
(both are written in Golang)

# What does Lightning Look Like?

- Download + install *Ind* + *Incli*
  - <https://github.com/lightningnetwork/ln/blob/master/docs/INSTALL.md>

# What does Lightning Look Like?

- Download + install *Ind* + *Incli*
  - <https://github.com/lightningnetwork/ln/blob/master/docs/INSTALL.md>
- Create *Ind* wallets
  - *Incli create*
  - *Inccli unlock*

# What does Lightning Look Like?

- Download + install *Ind* + *Incli*
  - <https://github.com/lightningnetwork/ln/blob/master/docs/INSTALL.md>
- Create *Ind* wallets
  - *Incli create*
  - *Inccli unlock*
- Fund Wallets

# What does Lightning Look Like?

- Download + install *Ind* + *Incli*
  - <https://github.com/lightningnetwork/Ind/blob/master/docs/INSTALL.md>
- Create *Ind* wallets
  - *Incli create*
  - *Inccli unlock*
- Fund Wallets
- Create Payment Channel
  - *Incli connect*  
*033f05189c200b946097ed0a81e1d47ec1b83ba5343670d1500659ac74be21de51@159.203.125.125*
  - *Inccli openchannel*  
*--node\_key=033f05189c200b946097ed0a81e1d47ec1b83ba5343670d1500659ac74be21de51 --local\_amt=100000000*

# What does Lightning Look Like?

- Send funds!
  - *Incli addinvoice --value=1000*
    - *"pay\_req":*  
*"Intb100u1pdza8wmp570kl3xp8psz4enjcxrlv66xmssj044hfe37nntd9waq0rwjs0hlqdqqcqzysdrasw9ufy30q9dv9awa0tw2lurm9d9w4updpk4dlx4hz4jugf65454usjmyu9w6vrj6tny7fw4uz56x3x2q6c6q2wgwfq9hapyz4ugghedxqu"*
  - *Incli sendpayment*  
*--pay\_req=Intb100u1pdza8wmp570kl3xp8psz4enjcxrlv66xmssj044hfe37nntd9waq0rwjs0hlqdqqcqzysdrasw9ufy30q9dv9awa0tw2lurm9d9w4updpk4dlx4hz4jugf65454usjmyu9w6vrj6tny7fw4uz56x3x2q6c6q2wgwfq9hapyz4ugghedxqu*
- Side-note: Lightning in it's current form is pull-based, you can't send funds to nodes without them first requesting you send them (different from underlying Bitcoin way of sending, which is push-based)

# What does Lightning Look Like?

- Close the channel
  - *Incli closechannel*  
*--funding\_txid=5fb65faf5ef665284bd82db5077b26d8eb3617cdb7e143a6bbef4e6301b41221*  
*--output\_index=0*



What does Lightning Look Like?

*Demo!*

# What does Lightning Look Like?

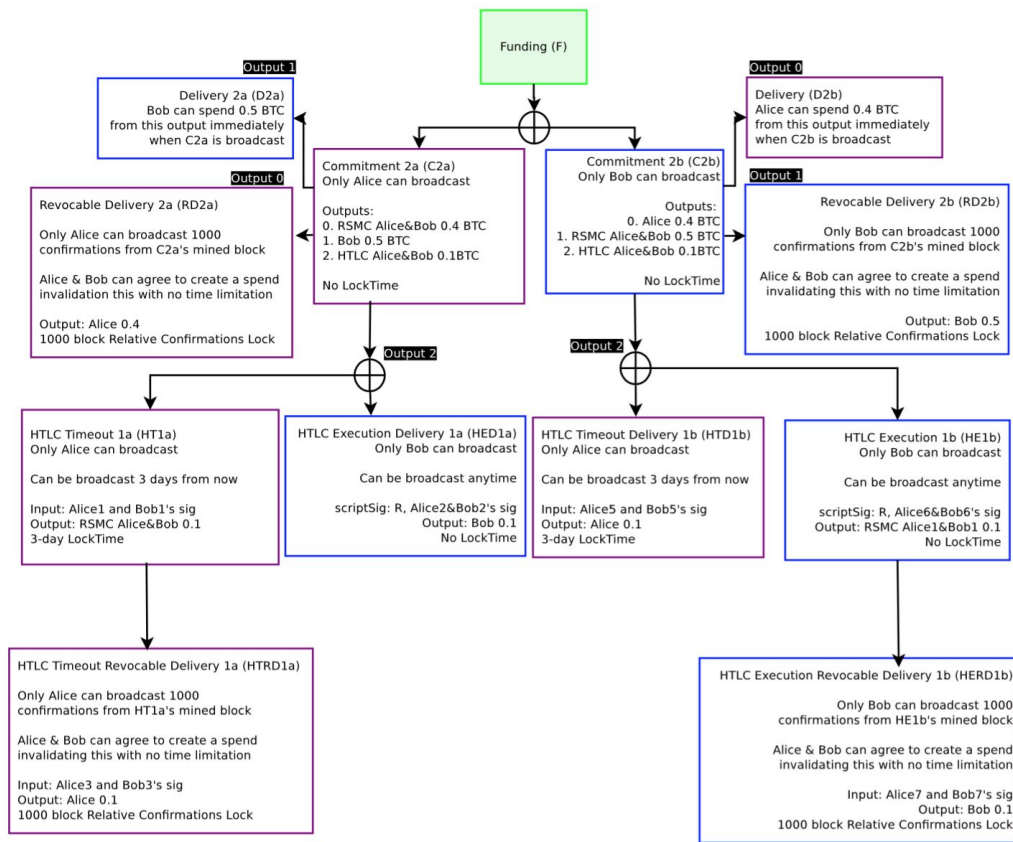
*Demo!*

*...come up and talk afterwards*

Break...

# How Does Lightning Work?

# How Does Lightning Work?



# How Does Lightning Work?

- Smart Contracts (Bitcoin SCRIPT language)
- Multisignature addresses
  - Funding transaction
  - Commitment Transactions
- Time-locked Transactions
- Revocation Transactions
- Hash Time-locked Transactions

# Smart Contracts (SCRIPT)

- Addresses are not what you think they are
  - Addresses are mini programs which, if upon ending return TRUE, prove ownership of coin
  - Address -> scriptPubKey (like a callback)
  - Ownership -> scriptSig (like arguments to the callback)

```
scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY  
OP_CHECKSIG
```

```
scriptSig: <sig> <pubKey>
```

# Funding Transaction

- Used to open/fund the the Alice-Bob channel
  - Alice and Bob each generate a keypair
  - Broadcast the transaction to the blockchain, wait for it to confirm
  - Happens after Commitment transactions are exchanged
- 2-2 multisig address
  - Requires 2 signatures, one from Alice, one from Bob, in order to be valid
  - Ensures both Alice and Bob must agree before moving any funds
  - Uses P2SH (Pay to Script Hash) address types

```
scriptPubKey: 2 <alicePubKey> <bobPubKey> 2 OP_CHECKMULTISIG
```

```
scriptSig: OP_0 <aliceSignature> <bobSignature> 2  
<alicePubKey> <bobPubKey> 2 OP_CHECKMULTISIG
```



# Commitment Transaction

- Represents the current state of the channel
  - How much coin do Alice and Bob have
- Protects Alice and Bob from being hurt if either is uncooperative
- If broadcasted, broadcaster must wait some time before claiming funds, but other channel member can claim their funds immediately

OP\_IF

# Penalty transaction

<revocationkey>

OP\_ELSE

`to\_self\_delay`

OP\_CSV

OP\_DROP

<local\_delayedkey>

OP\_ENDIF

OP\_CHECKSIG

# Future Study

Master resource list (you will know more than me after watching/reading all of this)

- <http://dev.lightning.community/resources/>

Mid-level explanation of Lightning

- <http://dev.lightning.community/tutorial/01-Incli/index.html>

Summaries of major BOLTs (Lightning's version of BIPs) by core dev Rusty Russell

- [https://medium.com/@rusty\\_lightning/the-bitcoin-lightning-spec-part-1-8-a7720fb1b4da](https://medium.com/@rusty_lightning/the-bitcoin-lightning-spec-part-1-8-a7720fb1b4da)

Tutorial on how to setup 3 lightning nodes and make payments

- <http://dev.lightning.community/tutorial/01-Incli/index.html>

API docs for *Ind*

- <http://api.lightning.community/>

Original Lightning Network Whitepaper

- <https://lightning.network/lightning-network-paper.pdf>

# Further Future Study

## Core Lightning Developer Blog

- <https://rusty.ozlabs.org/?p=462>

## Good SO post explaining Lightning

- <https://bitcoin.stackexchange.com/questions/48283/what-is-the-lightning-network-proposal-what-problem-is-it-trying-to-solve>

## Transaction Timelock Explanation

- <https://en.bitcoin.it/wiki/Timelock>

## 2015 Explanation of Lightning Network (useful when trying to understand what's changed)

- <https://github.com/ElementsProject/lightning/blob/master/doc/deployable-lightning.pdf>

# Further Watching

- <https://www.youtube.com/watch?v=7FWKc8IM4Ek>