Lecture 10 (1): Lightning and Atomic Swaps

Yury Yanovich

November 29, 2018

1. Lightning

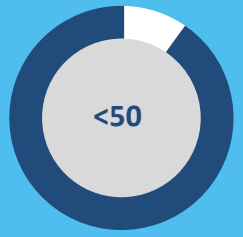
BLOCKCHAIN SCALABILITY PROBLEM

Visa/Mastercard

Peak capacity in 2017

Source: visa.com

Public/Private blockchains



Public blockchains

Based on current performance of Ethereum blockchain

Private blockchains

Based on performance of existing platforms

Lightning Network

For the whole Lightning Network

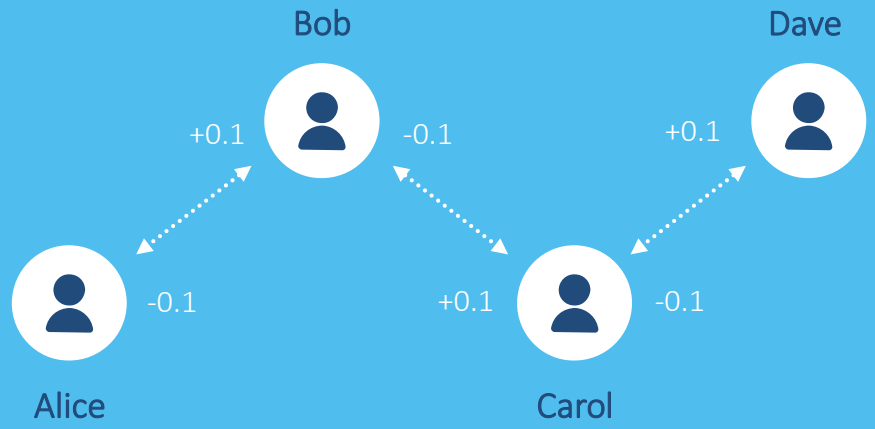
Based on academic research and modeling

All number are provided in transactions per second.

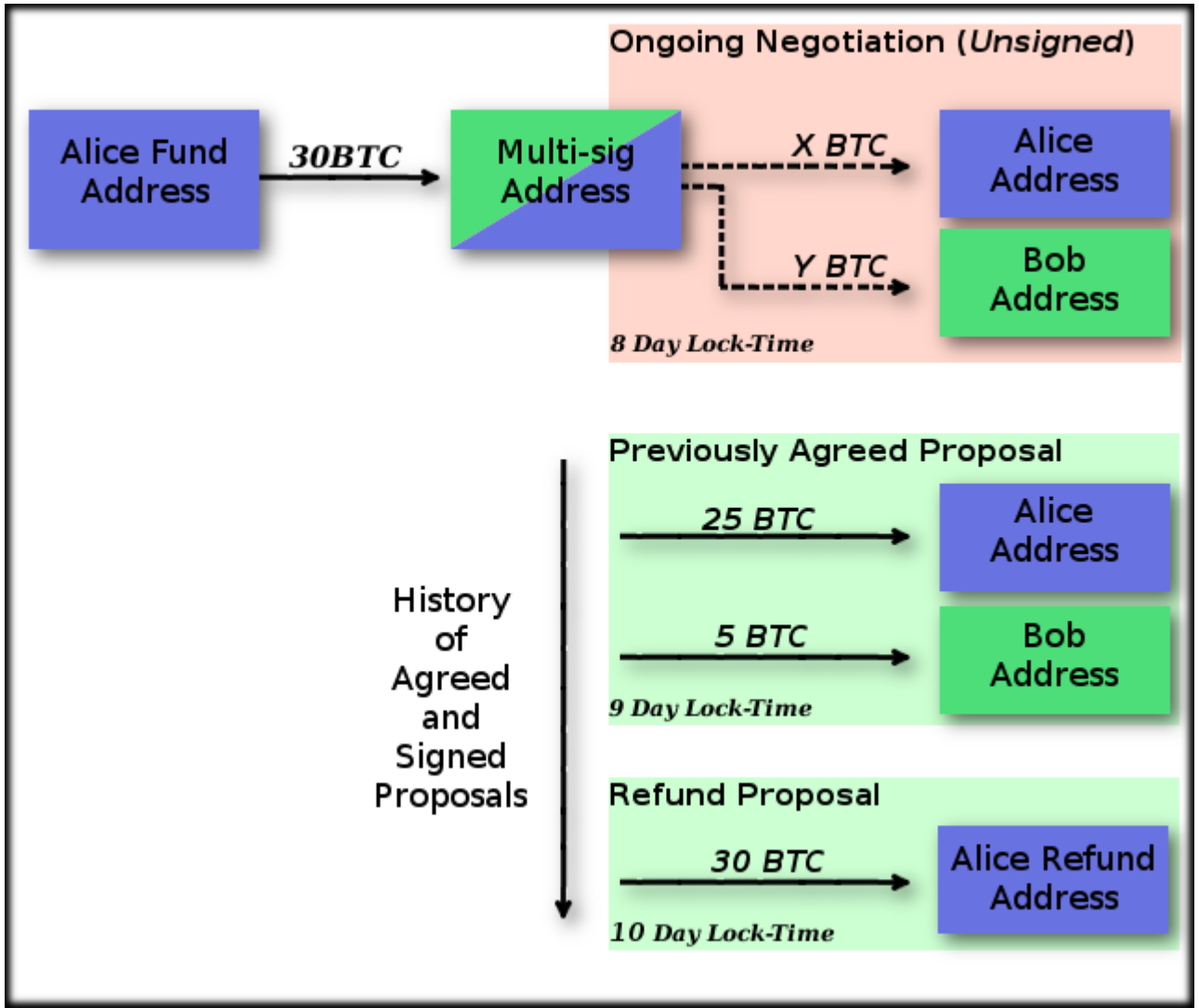
HOW IT WORKS

Network building brick – payment channel

Payments performed through chain of payment channels



Micropayments



Micropayments

Alice is sending value, Bob is receiving it. Alice-side description:

1. Create a public key (K1). Request a public key from Bob (K2).
2. Create and sign but do not broadcast a transaction (T1) that sets up a payment of (for example) 30 BTC (for example) to an output requiring both Bob's private key and one of your own to be used. A good way to do this is use OP_CHECKMULTISIG. The value to be used is chosen as an efficiency tradeoff.
3. Create a refund transaction (T2) that is connected to the output of T1 which sends all the money back to yourself. It has a time lock set for some time in the future, for instance a few hours. Don't sign it, and provide the unsigned transaction to Bob. By convention, the output script is "2 K1 K2 2 CHECKMULTISIG"
4. Bob signs T2 using its private key associated with K2 and returns the signature to Alice. Note that he has not seen T1 at this point, just the hash (which is in the unsigned T2).
5. The Alice verifies Bob’s signature is correct and aborts if not.

Micropayments (2)

6. Alice signs T1 and passes the signature to Bob, which now broadcasts the transaction (either party can do this if they both have connectivity). This locks in the money.
7. Alice then creates a new transaction, T3, which connects to T1 like the refund transaction does and has two outputs. One goes to K1 and the other goes to K2. It starts out with all value allocated to the first output (K1), i.e., it does the same thing as the refund transaction but is not time locked. Alice signs T3 and provides the transaction and signature to Bob.
8. Bob verifies the output to itself is of the expected size and verifies the client's provided signature is correct.
9. When Alice wishes to pay Bob, she adjusts its copy of T3 to allocate more value to the Bob's output and less to its own. She then re-signs the new T3 and sends the signature to Bob. Alice does not have to send the whole transaction, just the signature and the amount to increment by is sufficient. Bob adjusts its copy of T3 to match the new amounts, verifies the signature and continues.

Micropayments (3)

- This continues until the session ends, or the 1-day period is getting close to expiry.
- The lock time and sequence numbers avoid an attack in which the AP provides connectivity, and then the user double-spends the output back to themselves using the first version of TX2.

LIGHTNING NETWORK: GRID OF PAYMENT CHANNELS

Powered by blockchain

LIGHTNING NETWORK (https://lightning.network/)

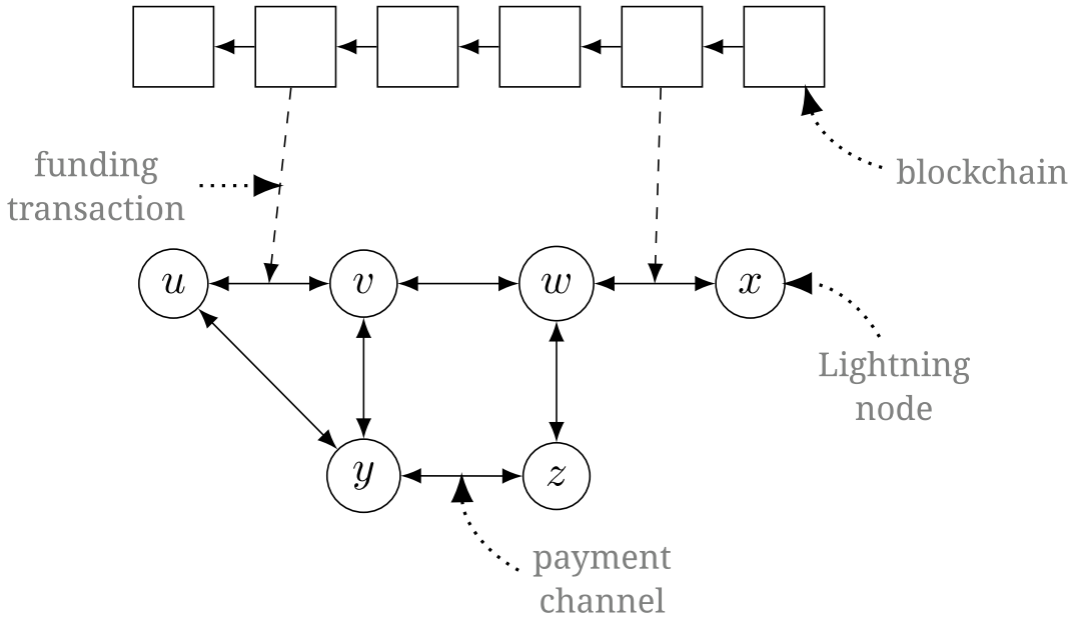
Lightning Network (LN) is a network of bi-directional payment channels, backed-up by blockchain. Channels can be opened between any institutions using the network – from banks and corporations to stores and their customers. That construction allows to do secure instant off-chain payments in Bitcoin and other blockchain-based currencies.

Lightning

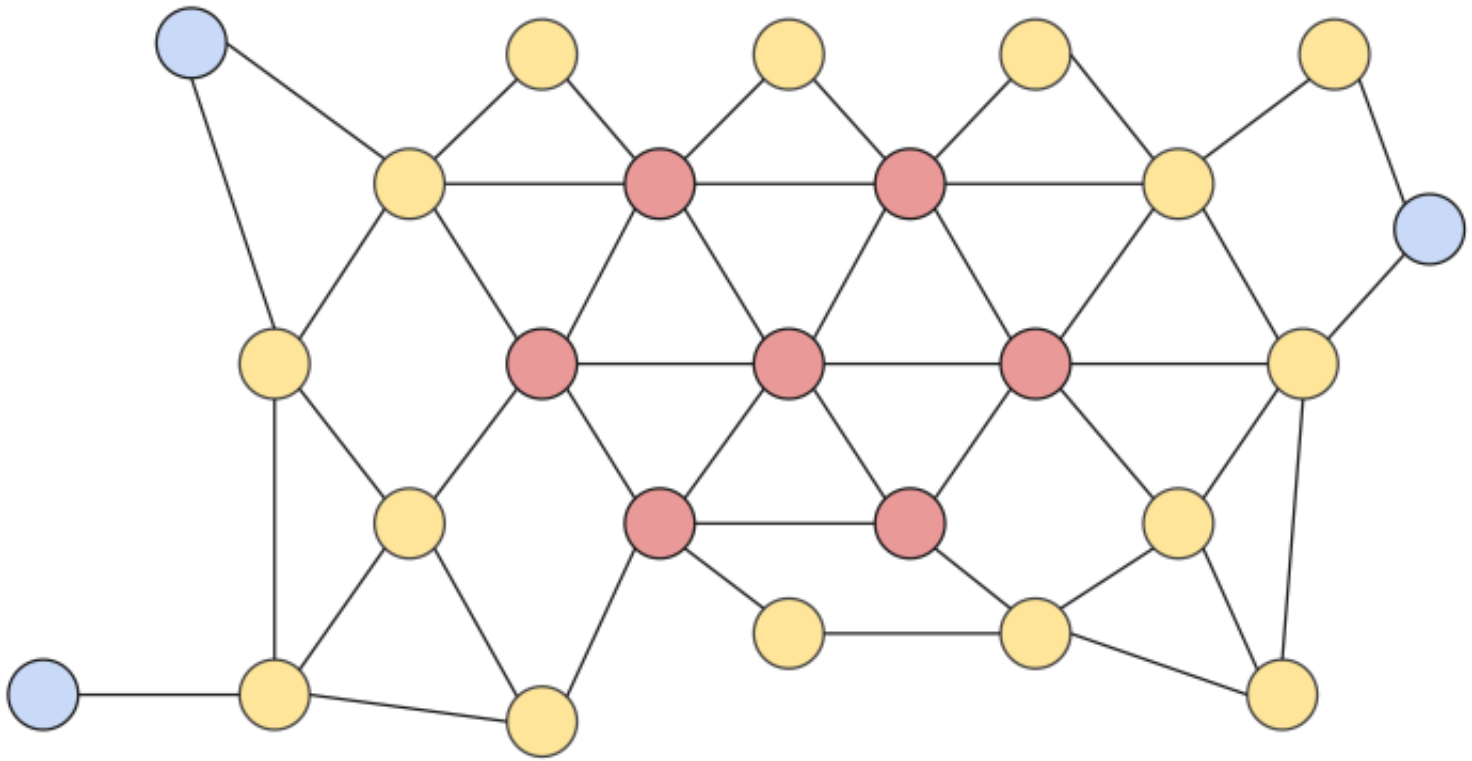
High level view of Lightning Network.

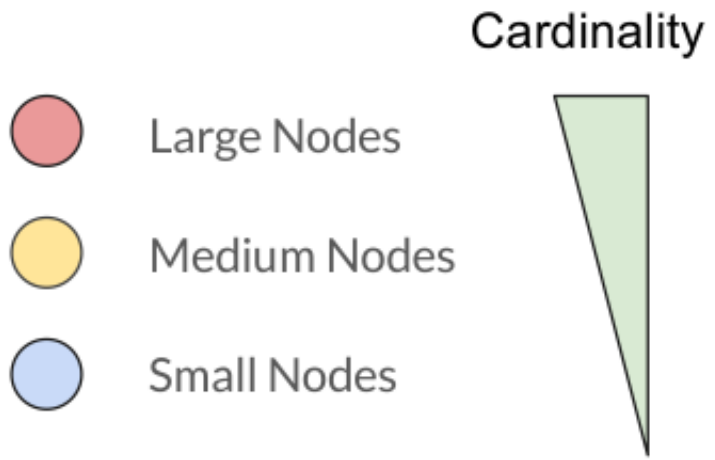
Node y can send payment to node x through nodes v and w (provided channels y – v , v – w , and w – x have enough bitcoin to forward the payment), through z and w, or through u , v and w .

If y routes a payment to x through v and w , intermediate nodes could receive fees for routing a payment: an HTLC payment from y to v would be greater than the HTLC payment from v to w by the value of the fee paid to v , and so on.



Payment Channel Network





Payment Example 1

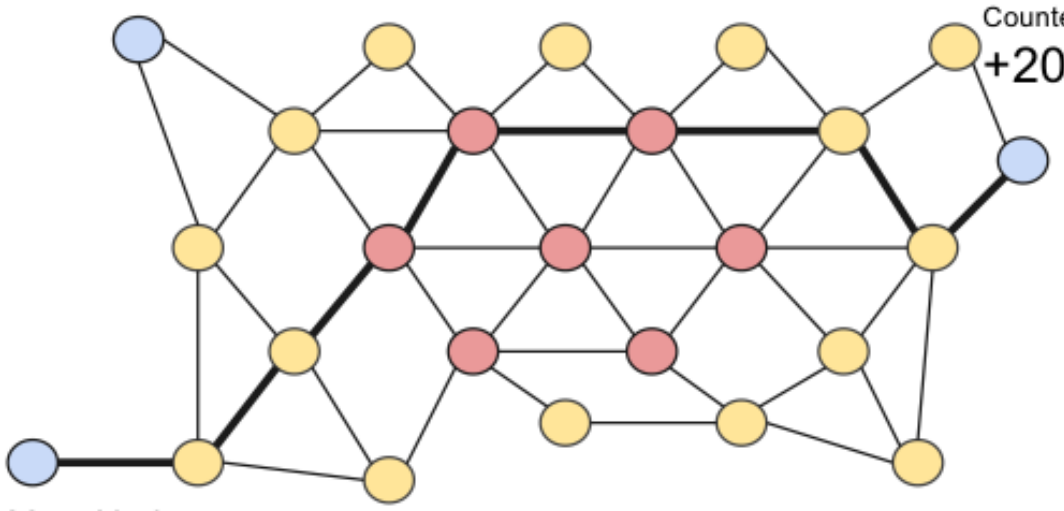
On-chain

Counter Party's Hot Wallet Address:
1F1tAaz5x1HUXrCNLbtMDqcw6o5GNn4xqX

200uBTC

Your Hot Wallet Address:
1ExAmpLe0FaBiTco1NADr3sSV5tsGaMF6hd

Lightning Network



Payment Example 2

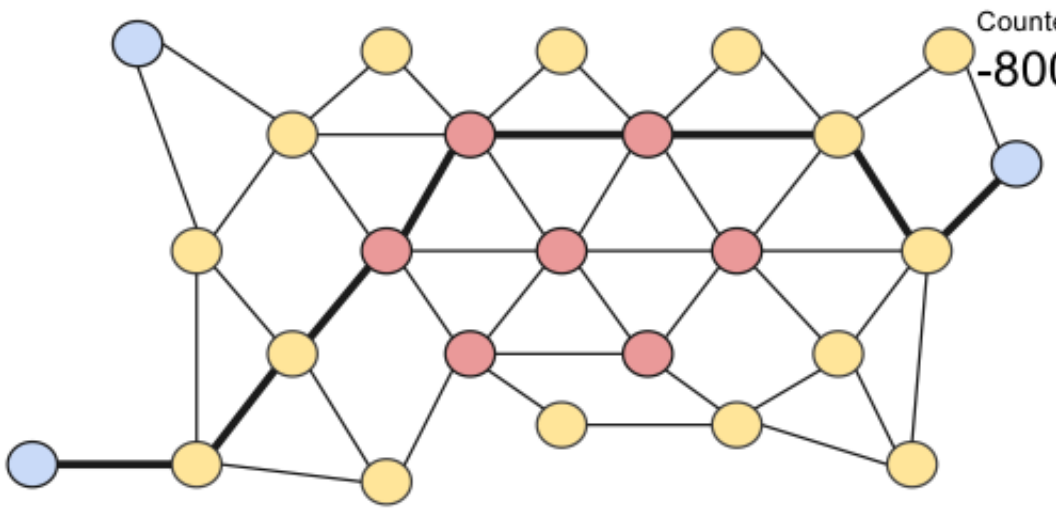
On-chain

Counter Party's Hot Wallet Address:
1F1tAaz5x1HUXrCNLbtMDqcw6o5GNn4xqX

800uBTC

Your Hot Wallet Address:
1ExAmpLe0FaBiTco1NADr3sSV5tsGaMF6hd

Lightning Network

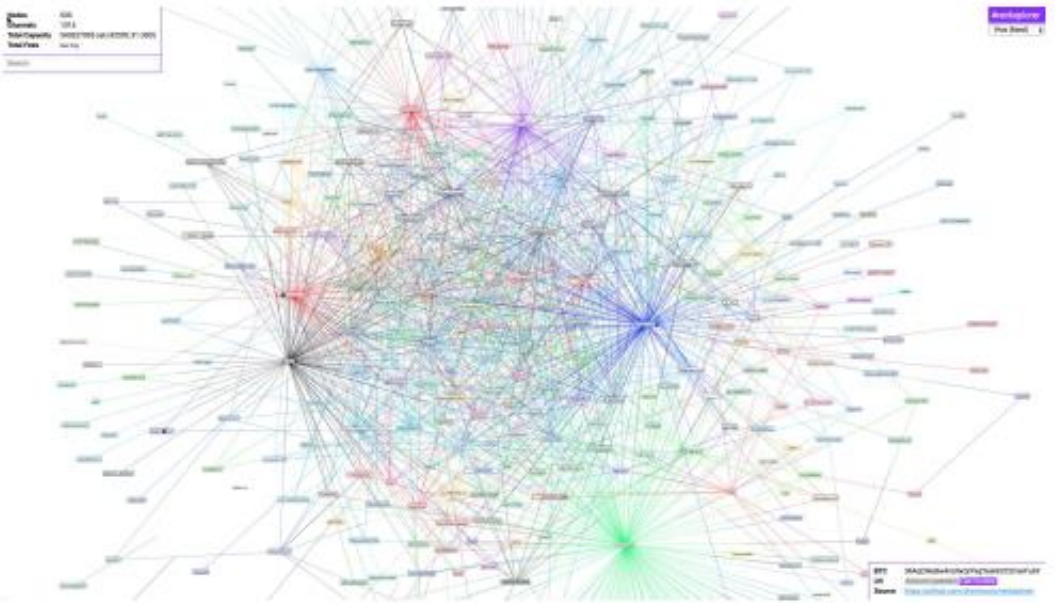


How many Lightning Nodes are available?

Testnet - https://explorer.acinq.co/

1720 Nodes / 5673 Channels

Mainnet - https://lnmainnet.gaben.win/

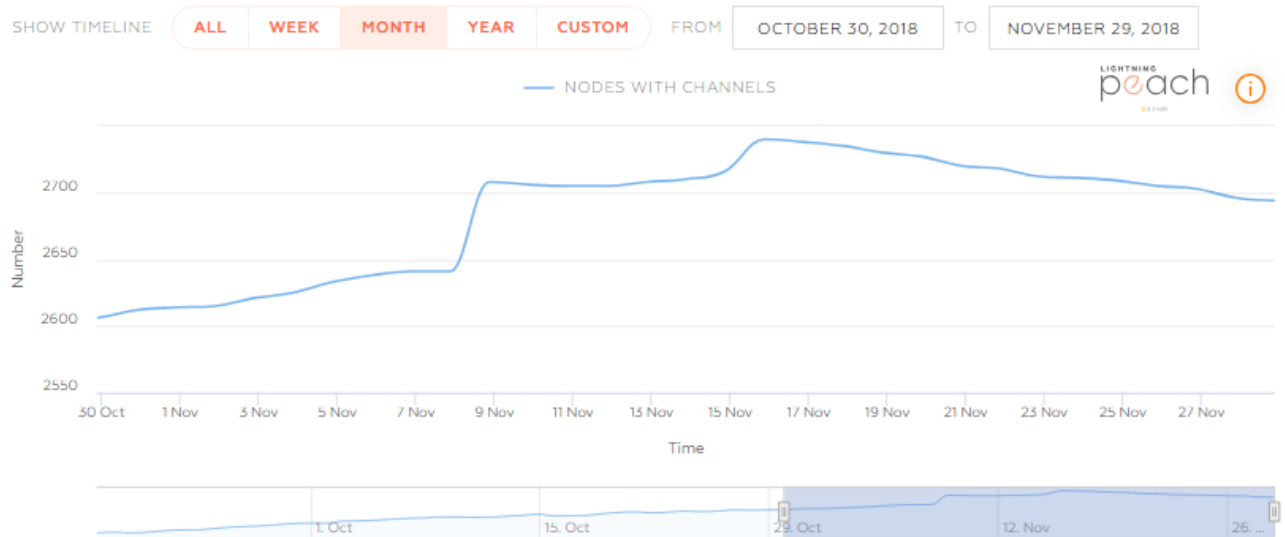


561 Nodes / 1401 Channels (As of 07Feb2018)

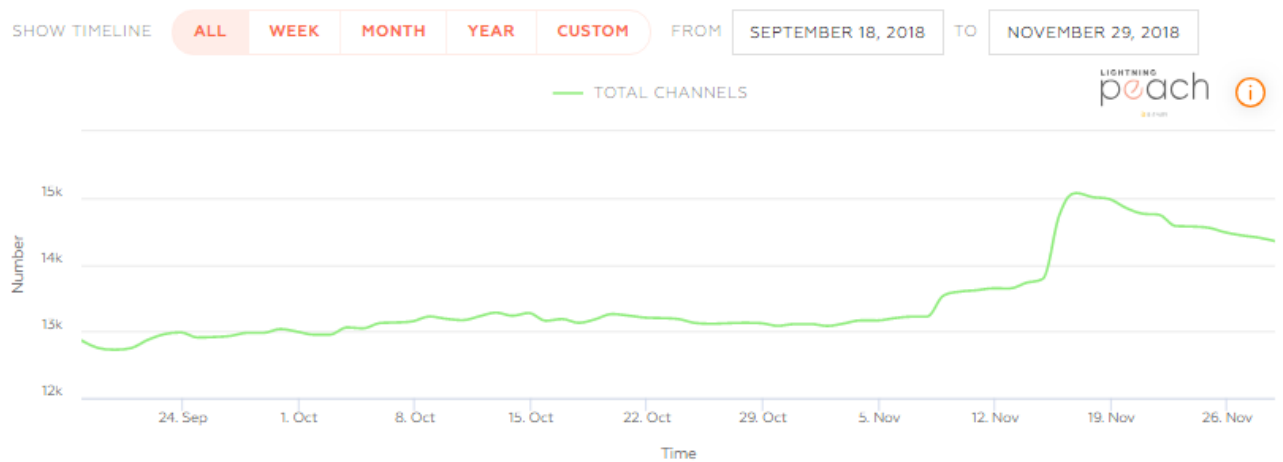
LIGHTNING peach
BITFURY

LN MONITOR | GET THE WALLET

Number of nodes



Number of channels

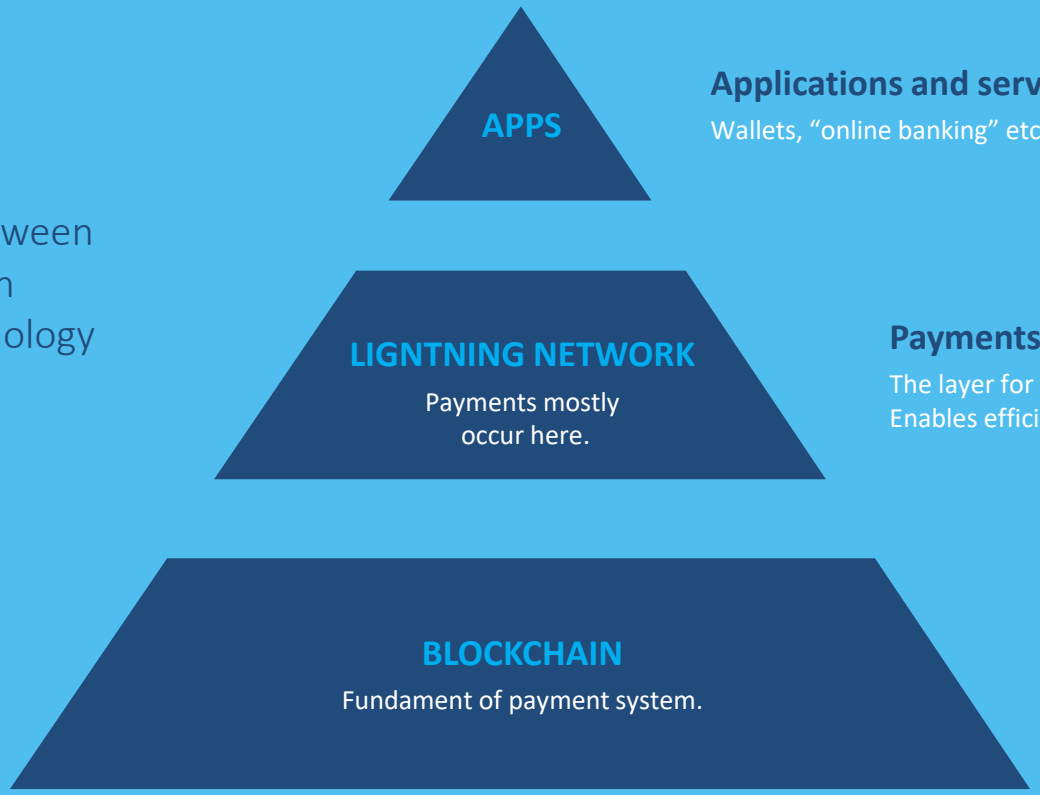


https://lightningpeach.com/ln-monitor

ROLE OF LIGHTNING NETWORK IN BLOCKCHAIN INFRASTRUCTURE

BIG PICTURE

Lightning Network is a layer between end users and blockchain, which allows broader use of the technology and solves scalability issue.



Applications and services
Wallets, “online banking” etc.

Payments level
The layer for optimal transaction processing.
Enables efficient micro transactions.

Trust enforcing level
Used for large transactions and to resolve disputable situations.

COMPARISON

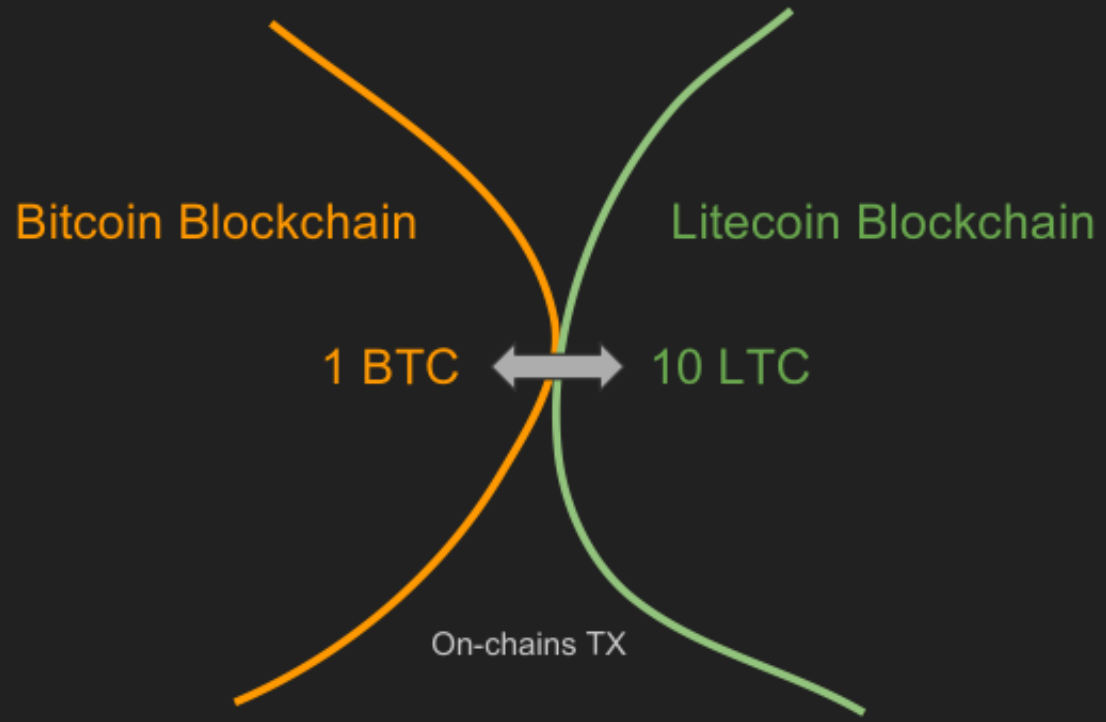
	Current payment systems	Blockchains	Lightning Network
SECURITY	Fraud is a significant issue, especially for Internet payments	Secure transactions	Secure transactions
SPEED	Settlement in a few days	Settlement in a few minutes	Clearing in seconds, settlement in minutes on blockchain
SCALABILITY	65 000 tps	50 / 5 000 tps	100 000 tps
MICRO TRANSACTIONS	Cost inefficient	Not very efficient to deploy due to fees	Supported and very efficient
CONFIDENTIALITY	Partial	Weak	Full
SUPPORT FOR DIFFERENT CURRENCIES	Yes	Single currency	Many different cryptocurrencies
OPEN SOURCE	No	Yes	Yes

Stage 1 : Segwit

23 AUG 2017

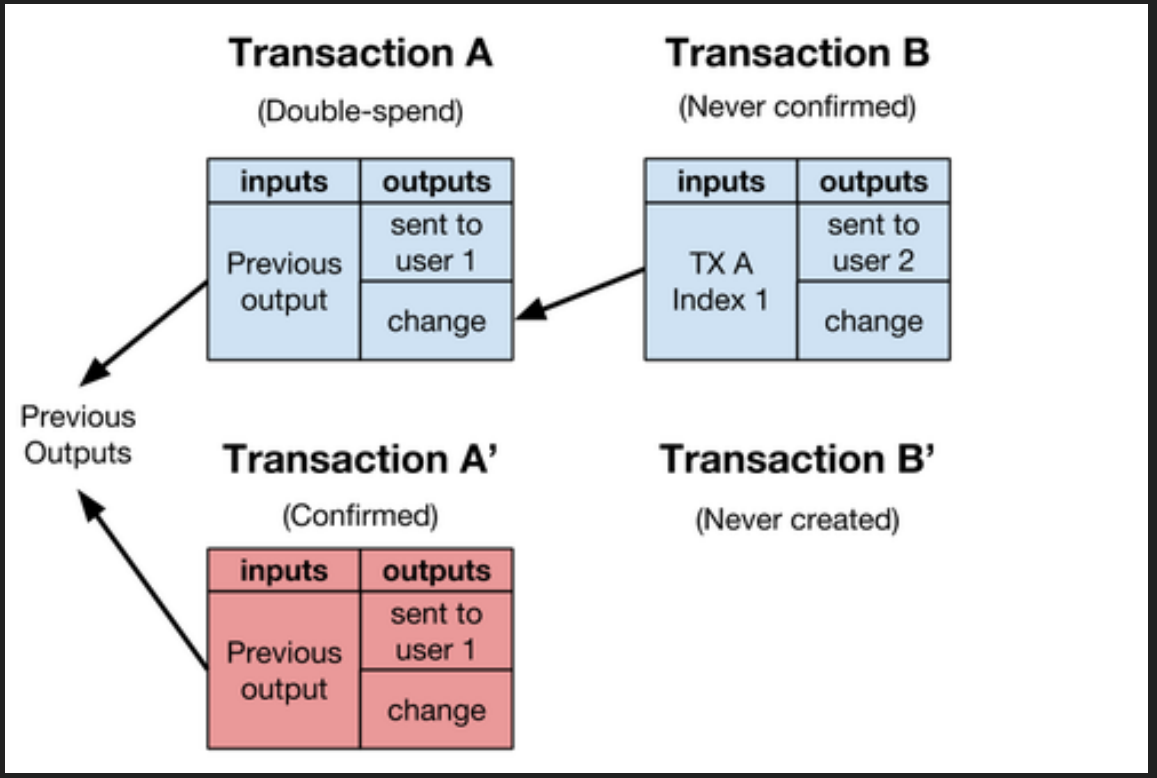
Segwit : Atomic Swap

many other coins support Segwit
(LTC, VTC, XMY, DCR, SYS, NAV, VIA…)



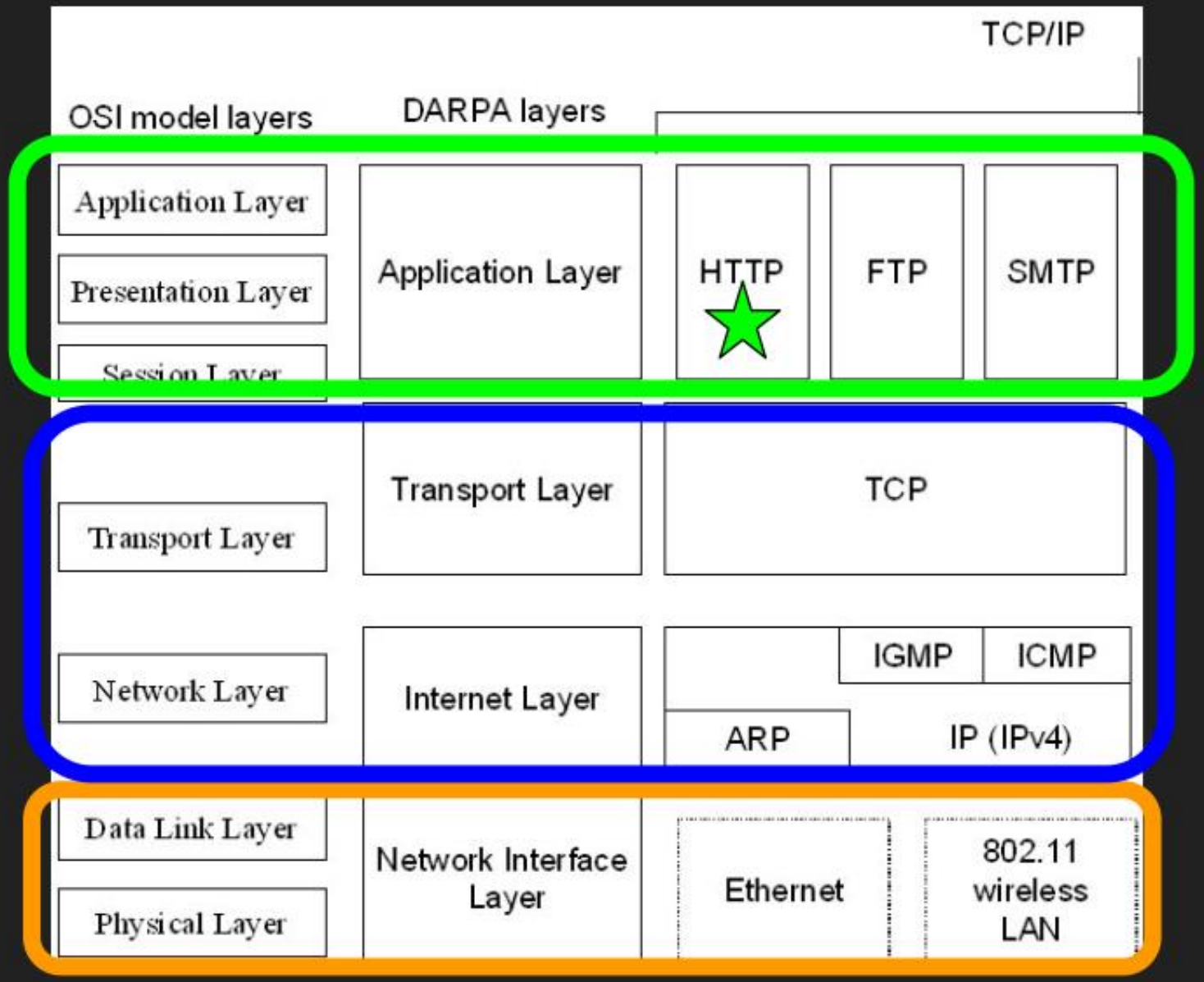
Segwit : Resolves malleability

- Segwit solves a minor bug (malleability) since the beginning of Bitcoin
- ⇒ LN POSSIBLE

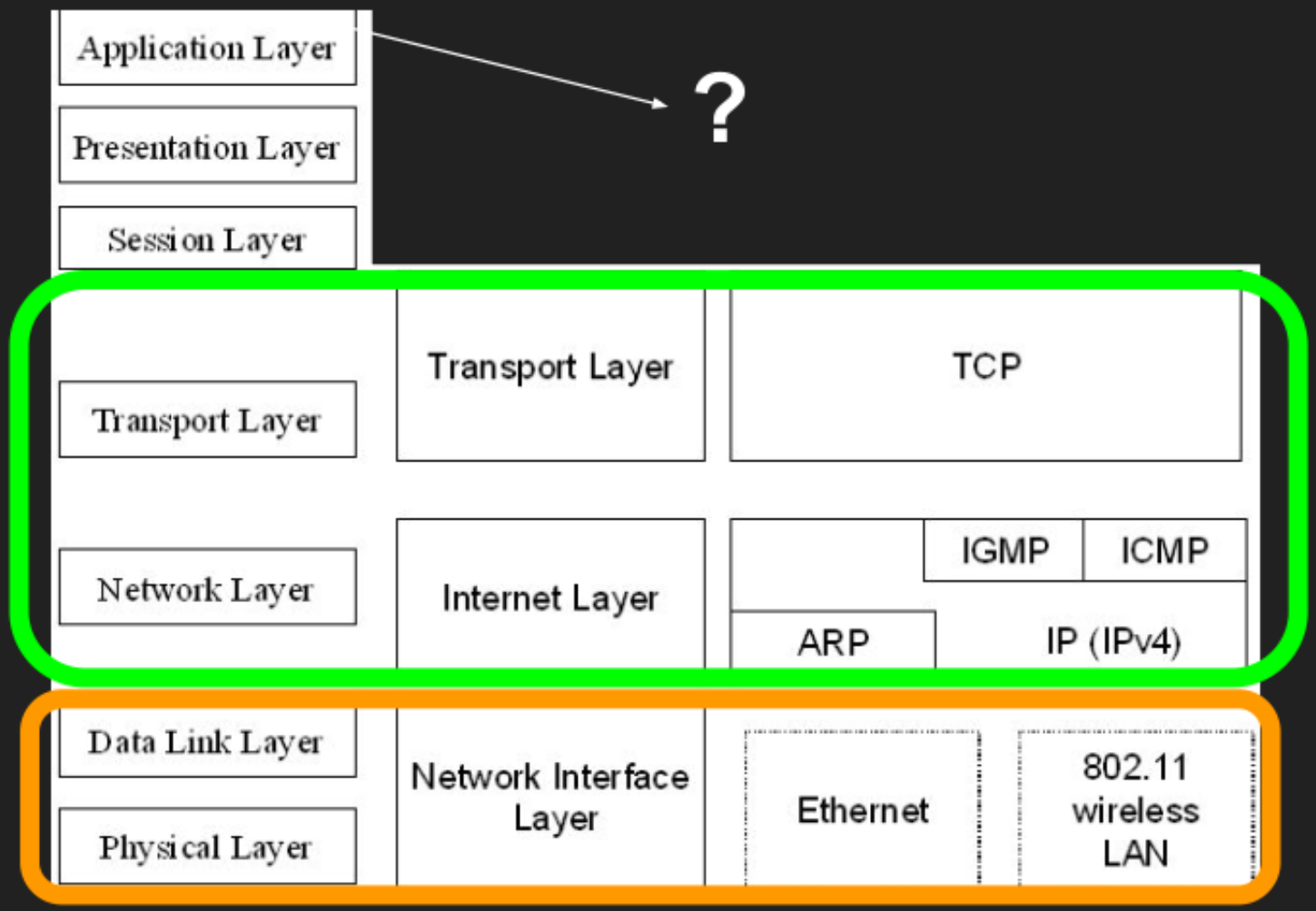


Stage 2 :
Second Layer

Internet


TCP/IP
OSI model layers
DARPA layers
Application Layer
Presentation Layer
Session Layer
Application Layer
HTTP
FTP
SMTP
Transport Layer
Transport Layer
TCP
Network Layer
Internet Layer
IGMP
ICMP
ARP
IP (IPv4)
Data Link Layer
Physical Layer
Network Interface Layer
Ethernet
802.11 wireless LAN

Bitcoin : layers



Internet of Value/Money (aka application layer)

Lightning Network / Segwit 2017+ (aka TCP/IP layer)

Blockchain use 2009-2017 (aka physical layer)

Lightning Network : Origin

Lightning Network : Code

The Bitcoin Lightning Network:
Scalable Off-Chain Instant Payments

Joseph Poon
joseph@lightning.network

Thaddeus Dryja
rx@awsomnet.org

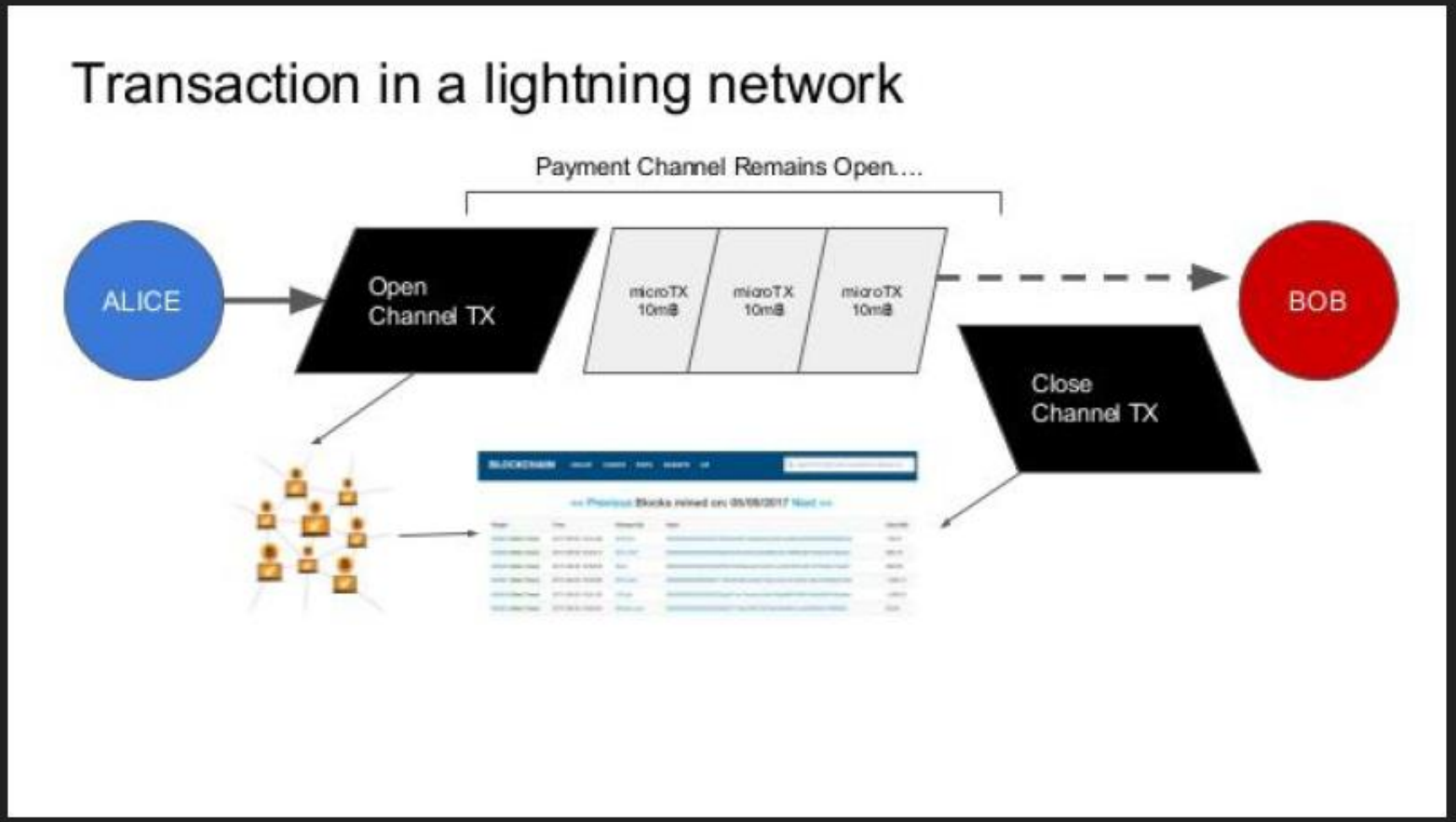
January 14, 2016

DRAFT Version 0.5.9.2

Abstract

The bitcoin protocol can encompass the global financial transaction volume in all electronic payment systems today, without a single custodial third party holding funds or requiring participants to have anything more than a computer using a broadband connection. A decentralized system is proposed whereby transactions are sent over a network of micropayment channels (a.k.a. payment channels or transaction channels) whose transfer of value occurs off-blockchain. If Bitcoin transactions can be signed with a new sighash type that addresses malleability, these transfers may occur between untrusted parties along the transfer route by contracts which, in the event of uncooperative or hostile participants, are enforceable via broadcast over the bitcoin blockchain in the event of uncooperative or hostile participants, through a series of decrementing timelocks.

Lightning Network: Channel



Lightning Network is on BOLT layer-2 protocol

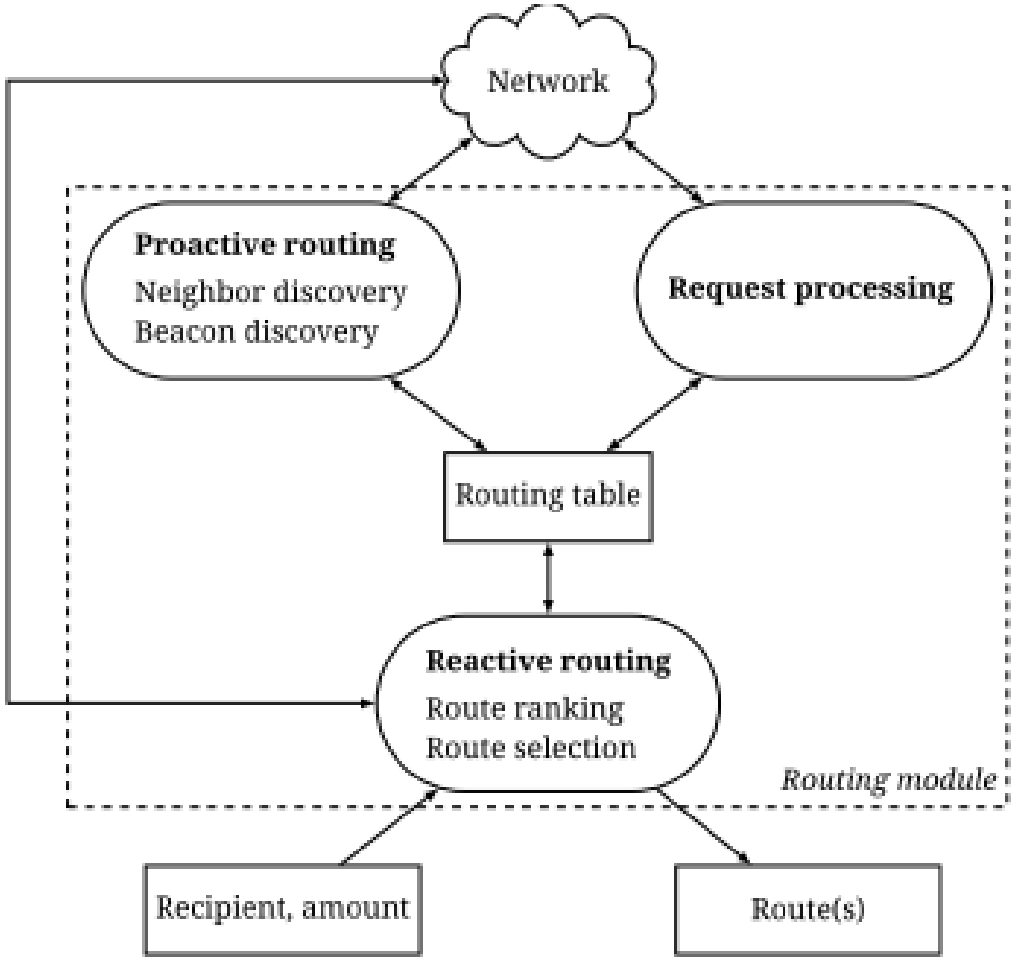
Lightning Network is a second layer technology on top of Bitcoin network protocol.

1. BOLT #1: Base Protocol
2. BOLT #2: Peer Protocol for Channel Management
3. BOLT #3: Bitcoin Transaction and Script Formats
4. BOLT #4: Onion Routing Protocol
5. BOLT #5: Recommendations for On-chain Transaction Handling
6. BOLT #7: P2P Node and Channel Discovery
7. BOLT #8: Encrypted and Authenticated Transport
8. BOLT #9: Assigned Feature Flags
9. BOLT #10: DNS Bootstrap and Assisted Node Location
10. BOLT #11: Invoice Protocol for Lightning Payments

https://github.com/lightningnetwork/lightning-rfc

Lightning: routing

The place of the routing module in the LN node's business logic.



Route selection

- Routing Table
- Neighborhood Discovery
- Ranking
  - Static: based on static information (i.e., on the information from routing tables only).
  - Dynamic: based on dynamic (and, possibly, static) information. In order to obtain information for dynamic ranking, an onion wrapped polling message is sent through every route among candidate routes collecting up-to-date information on channels and nodes in the route.

2. Atomic Swaps

What is a “Swap”?

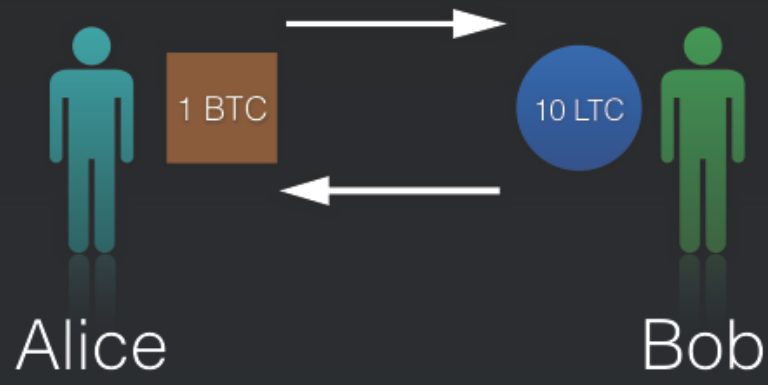
Other word for “Exchange”:

Example:

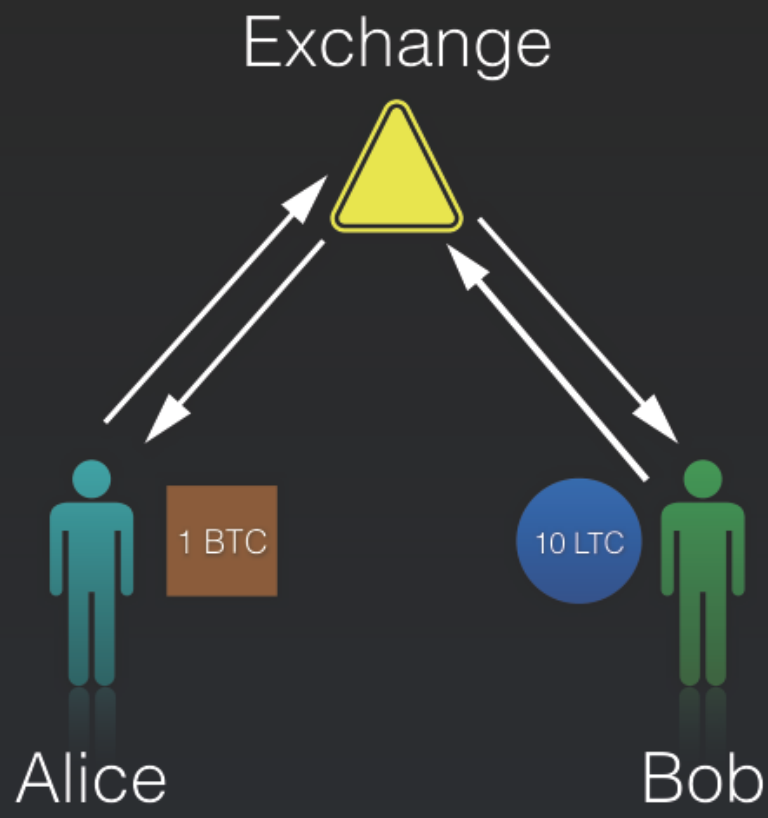
Alice has BTC
Bob has LTC

They negotiate with each other and both agree to exchange:

Alice wants to give Bob 1 BTC in return for 10 LTC



What is a “Swap”?



Current Situation:

- Need for trusted 3rd parties
  –> Centralized Exchanges

- What if…

  they somehow could exchange
  “*exactly at the same time*” ?

Atomic Swap

Atomicity:

> "*An **atomic transaction** is an **indivisible** […] **series** of […] **operations** such that **either all occur, or nothing** occurs.*"

Atomic Swap

- Basic Idea:
  Create transactions, on both chains
- Add a spending condition, which only can get true on both chains simultaneously (even if chains are totally unrelated)

Atomic Swap

- Already in 2011 Mike Hearn proposed an idea to achieve this (see Bitcoin Wiki, Article "Contract", Example 5): *"Trading across chains"*

Atomic Swap

The basic idea is the following:

- Initiator (i.e. Alice) thinks of a random secret ("**S**") example:

  "***correct horse battery staple***"

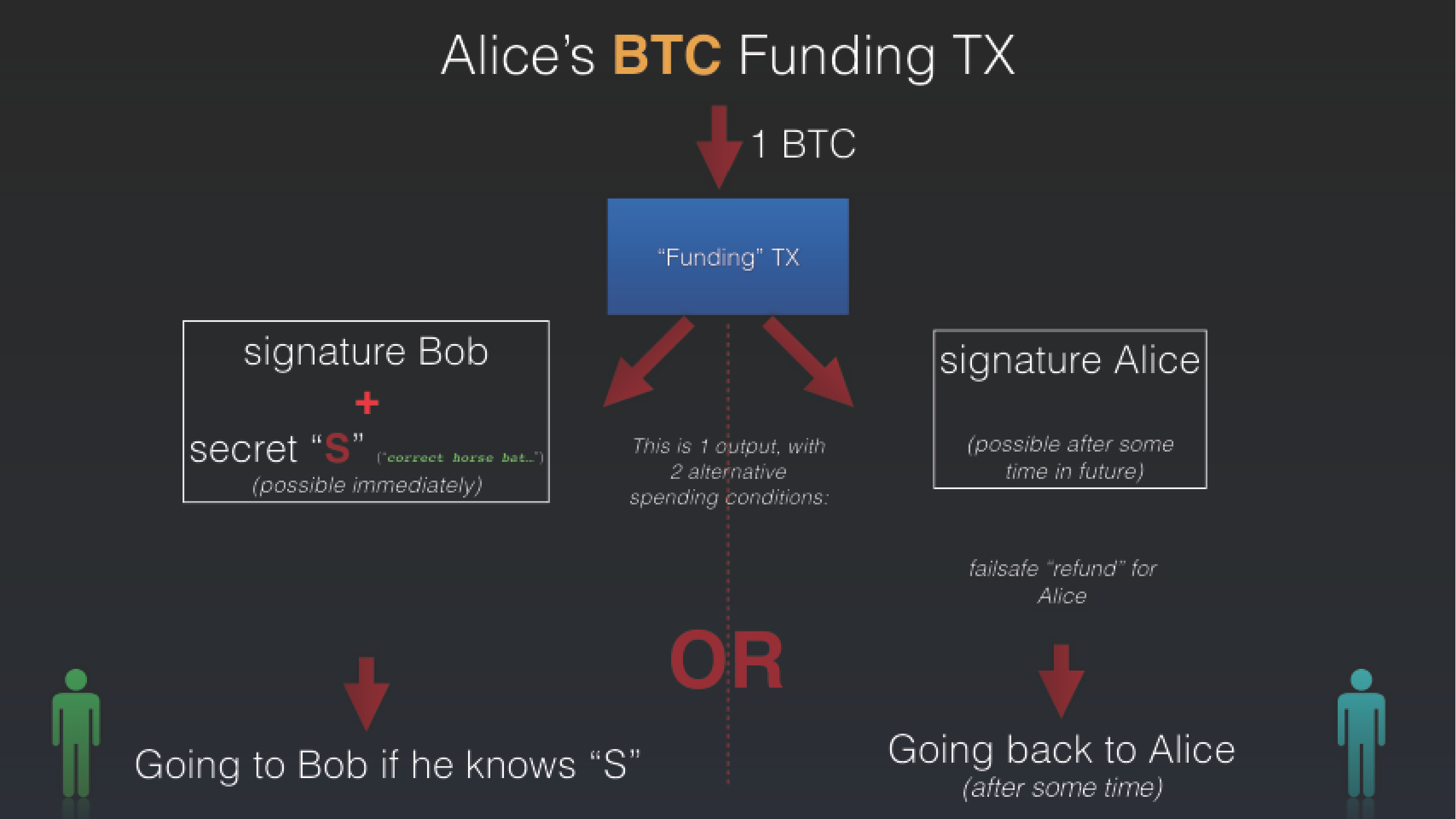
- She calculates the hash ("**H**") of the secret "S":

  "***2259cd5b42ae4d70deaa3d8d2ead2bb32ed3677b***"

Atomic Swap

- Then Alice sends her funds (1 BTC) into a “Contract TX” (or “Funding” TX) on the BTC chain, locking the output like this:

Alice's **BTC** Funding TX



Alice's **BTC** Funding TX

1 Output (*ie. 1 BTC*)
Can be spent **EITHER**:

- script:

```
IF
   <Key_Bob> CHECKSIGVERIFY
   HASH160 <H> EQUALVERIFY
ELSE
   <Key_Alice> CHECKSIGVERIFY
   <Alice_Timeout> CHECKLOCKTIMEVERIFY
ENDIF
```

by Bob (*Key_Bob*) if he knows the secret "**S**" which will hash to the value "**H**"

OR:

by Alice (*Key_Alice*) at some time in future (failsafe refund)

Alice's **BTC** Funding TX

*This type of Tx are called "**HTLC**":*
*"**Hash-time-locked contract**"*

1 Output (*ie. 1 BTC*)
Can be spent **EITHER**:

- script:

```
IF
   <Key_Bob> CHECKSIGVERIFY
   HASH160 <H> EQUALVERIFY
ELSE
   <Key_Alice> CHECKSIGVERIFY
   <Alice_Timeout> CHECKLOCKTIMEVERIFY
ENDIF
```

by Bob (*Key_Bob*) if he knows the secret "**S**" which will hash to the value "**H**"

OR:

by Alice (*Key_Alice*) at some time in future (failsafe refund)

- spending input data:

```
1 <sig Bob> <secret S>
```

OR:

```
0 <sig Alice> (after some time)
```

Bob's **LTC** Funding TX

1 single Output (*ie. 10 LTC*)
Can be spent **EITHER**:

- script:

```
IF
   <Key_Alice> CHECKSIGVERIFY
   HASH160 <H> EQUALVERIFY
ELSE
   <Key_Bob> CHECKSIGVERIFY
   <Bob_Timeout> CHECKLOCKTIMEVERIFY
ENDIF
```

by Alice (*Key_Alice*) if she knows (and provides) the secret "**S**" which will hash to the value "**H**"

OR:

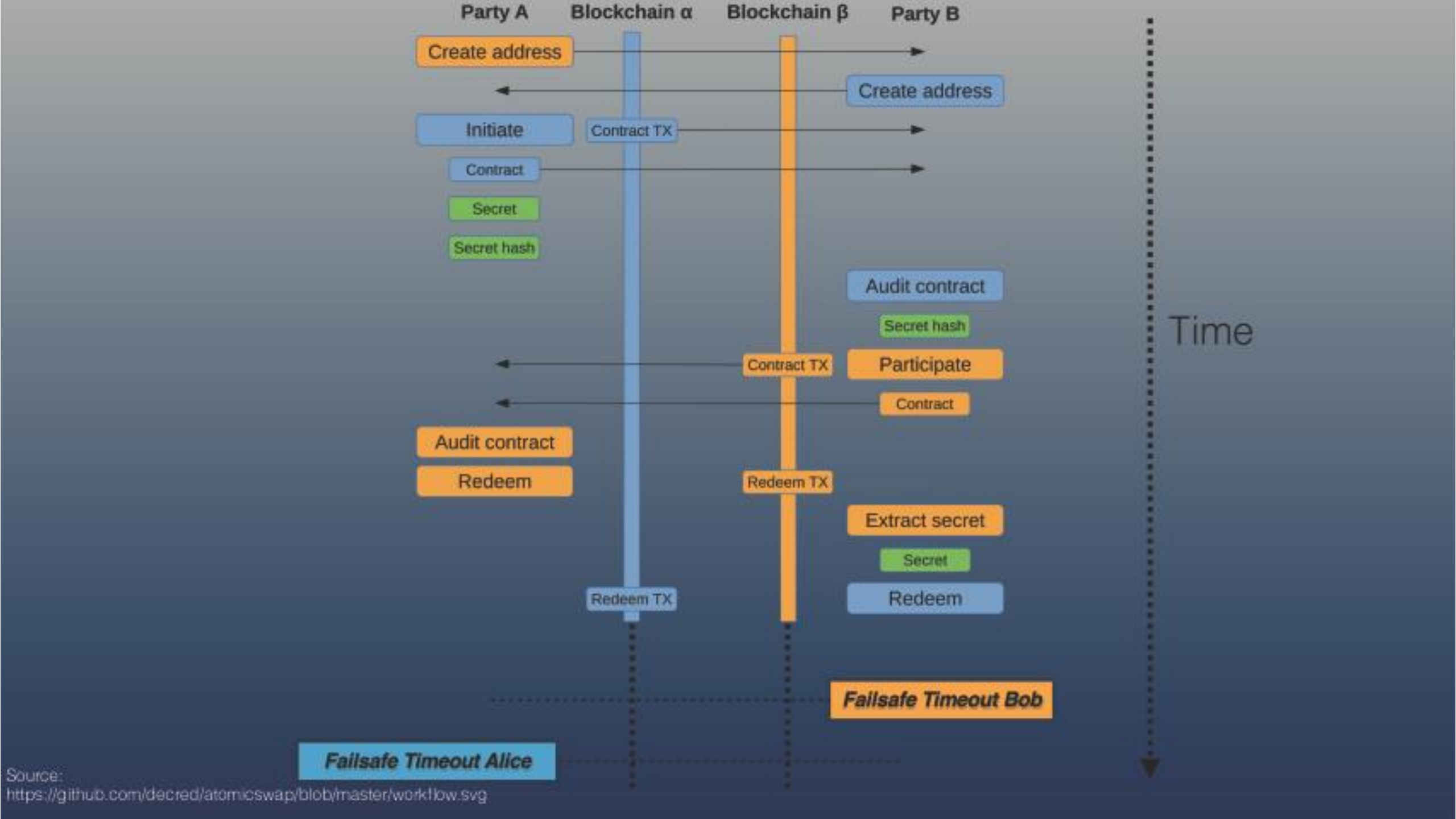
by Bob (*Key_Bob*) at some time in future (failsafe refund)

- spending input data:

```
1 <sig Alice> <secret S>
```

OR:

```
0 <sig Bob> (after some time)
```



Source:
https://github.com/decred/atomicswap/blob/master/workflow.svg

What do the 2 chains need?

- possibility to somehow time-lock funds (CLTV or CSV on Bitcoin-like chains)
- support the same hashing algorithm in the evaluating script
- branching support in scripts (if / else) to realize failsafe path
- ability to check hashes and signatures in evaluating script

What do the 2 chains need?

- this is true for most Bitcoin-like chains
- also for smart contract chains (which allow to program conditions totally free), i.e. Ethereum..
- not possible on chains which don't allow to express spending conditions based on hash pre-image

Secret size attack

Remember, our secret:

- "***correct horse battery staple***"
  which hashes to:
  "***2259cd5b42ae4d70deaa3d8d2ead2bb32ed3677b***"

- Is there a limit for the maximum possible length of a secret?

- For Bitcoin: Yes! 520 Bytes (source)

```
20  // Maximum number of bytes pushable to the stack
21  static const unsigned int MAX_SCRIPT_ELEMENT_SIZE = 520;
```

- What if this limit is different between two chains?

source: https://gist.github.com/markblundeberg/7a932c98179de2190049f5823907c016, Dr. Mark B. Lundeberg, Feb 15, 2018

Secret size attack

Example:

- Imagine evil attacker "**Eve**" 😈 owns **FantasyCoin "FC"** which allows max. 300 bytes-sized script elements
- Eve and Alice agree to trade **10000 FC against 10 BTC**
- Eve creates a **secret** which is **>300 bytes but <520 bytes long** and hashes it into 160 bytes
- Eve proceeds as discussed before (locks her FC into the Funding TX, informs Alice)
- As soon as Alice has locked her 10 Bitcoin in her Funding TX, Eve can claim them (as planned, because she as initiator knows the secret)
- But when Alice now wants to claim her 10000 FC in return, she cannot: although she now knows the secret, she cannot use it, as it's too large to be used in a FC coin script.

Secret size attack

Fortunately, there is an easy solution:

- Add conditon into the script which commits to the length of the secret
- For Bitcoin Script: add "**OP_SIZE xx OP_EQUALVERIFY**" commands. This way the size will be public beforehand:

```
IF
   <Key_Eve> CHECKSIGVERIFY
   SIZE 28 EQUALVERIFY
   HASH160 <H> EQUALVERIFY
ELSE
   <Key_Alice> CHECKSIGVERIFY
   <Bob_Timeout> CHECKLOCKTIMEVERIFY
ENDIF
```

Tools for on-chain swaps

- **“decred”**: https://github.com/decred/atomicswap/

- Commanline tools (for each supported chain) to create onchain atomic swap transactions and perform all steps in the protocol:

```
Commands:
  initiate <participant address> <amount>
  participate <initiator address> <amount> <secret hash>
  redeem <contract> <contract transaction> <secret>
  refund <contract> <contract transaction>
  extractsecret <redemption transaction> <secret hash>
  auditcontract <contract> <contract transaction>
```

```
$ btcatomicswap --testnet --rpcuser=user --rpcpass=pass initiate n31og5QGuS28dmHpDH6PQD5wmVQ2K2spAG 1.0
Secret:      3e0b064c97247732a3b345ce7b2a835d928623cb2871c26db4c2539a38e61a16
Secret hash: 29c36b8dd380e0426bdc1d834e74a630bfd5d111
```

- support several chains: **BTC, BCH, LTC, Monacoin, Particl, Polis, Vertcoin, Viacoin, Zcoin** (Ethereum contract currently WIP: *AtomicSwap.sol)*

Alternative protocol

- BarterDEX by jl777 (used in the Komodo platform): https://komodoplatform.com/decentralized-exchange/
- Realizes atomic swaps with key pairs chosen using the "Cut and choose" principle, as proposed by TierNolan in bitcointalk forum in 2016: https://bitcointalk.org/index.php?topic=1340621.msg13828271#msg13828271 and https://bitcointalk.org/index.php?topic=1364951
- Whitepaper: https://github.com/KomodoPlatform/KomodoPlatform/wiki/barterDEX-Whitepaper-v2
- More complex, takes 7 steps, needs deposits as incentive, needs percental fees to mitigate incentive to exploit the cut-and-choose process, …). But works also if one of the 2 chains has no support for CHECKLOCKTIMEVERIFY
- https://dexstats.info/

Same-chain token swaps

For the sake of completeness:
Not cross-chain, but these are also examples of atomic swaps:

- many ICO contracts on the Ethereum blockchain (atomically exchange ETH against tokens)
- Onchain ERC-20 Exchanges (like EtherDelta, IDEX, 0x, etc..) atomically exchange ERC-20 Tokens

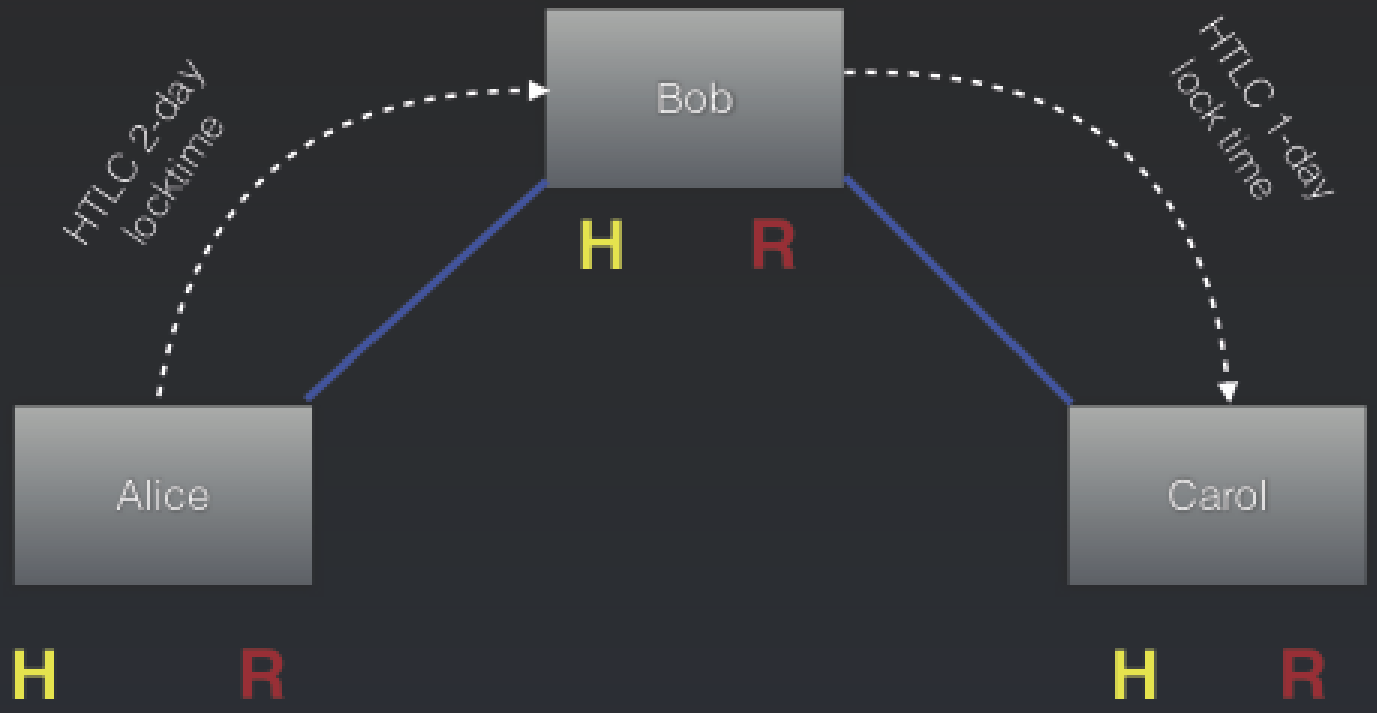
Recap onchain swap

- Needs an out-of-band communication channel (i.e. Alice and Bob need to communicate outside the protocol (to find each other, for negotiation, updates, etc…)
- All transactions happen on-chain (and need time for confirmation)
- Need to pay appropriate fees on both chains, and set reasonable refund timeouts
- Needs 2 TX (on both chains)

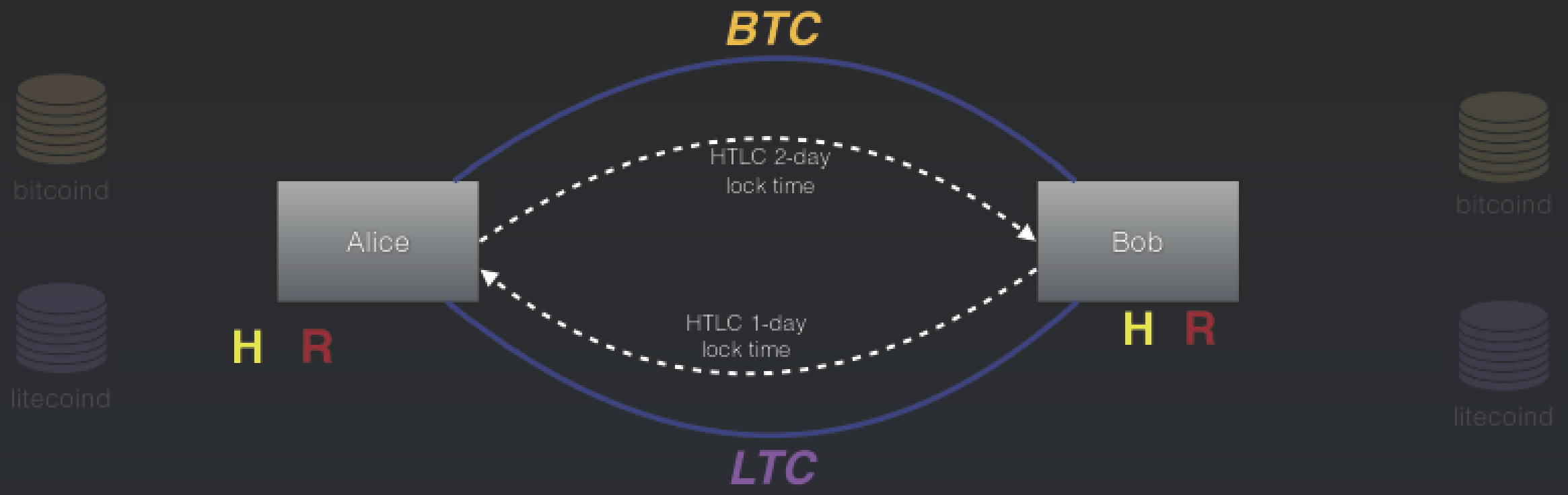
Lightning

- Now, we have lightning.. Yay! 😃⚡⚡⚡
- A lightning payment consists of a series of TXs happening atomically over a route of channels
- Very similar type of transactions (also HTLCs, slightly different for channel maintenance, but same base principle)

Recap: One-hop-Lightning payment example



The same but adapted for Swap
(Alice is both, sender *and* receiver)



Status Lightning swaps

- the Lightning Paper mentions “Cross-Chain Payments” as a further possible use case for the Lightning Network, but no further details yet
- The lightning protocol specifications (BOLTs) also don’t mention swap related messages (yet)
- ***Conner Fromknecht*** from Lightning Labs did a proof-of-concept and performed a BTC/LTC swap on testnet using a modified “lnd” lightning daemon
- See blog article: https://blog.lightning.engineering/announcement/2017/11/16/ln-swap.html (proof-of-concept branch on GitHub and a howto, howto2 to recreate his demo)

Details proof of concept

modified “lnd” daemon (lnd already supports Bitcoin and Litecoin, but not simultaneously yet)

- added new custom cli-commands / parameters:
  - **`--ticker`** (to specify currency)
  - **`queryswaproutes`** (outputs a cross-chain route, used as input for sendtoroute, to explicitly send the payment over this route)Exchange rate statically configured
- exchange rate statically configured
- Basically, swap works like this:

```
lncli … queryswaproutes --in_amt=100000 --in_ticker=LTC --out_ticker=BTC
lncli … addinvoice --value=100000 --ticker=LTC    —> payment_hash
lncli queryswaproutes … | lncli sendtoroute --payment_hash <hash from
invoice>
```

Summary

- on-chain swaps already possible (and tooling getting better)
- Lightning supports cross-chain swaps perfectly by principle, it also seems like a natural fit for lightning. But not fully there yet.

  There still is the need for concrete specifications to be defined (gossip protocol, exchange rate publishing, advertising of supported currency swaps, etc..)

References

- "*Example 5: Trading across chains*", *2011*, https://en.bitcoin.it/wiki/Contract#Example_5:_Trading_across_chains
- "*Alt chains and atomic transfers*", 2013, https://bitcointalk.org/index.php?topic=193281.msg2003765#msg2003765
- "*Malleability, deposits and atomic cross chain transfers*", 2014, https://bitcointalk.org/index.php?topic=515370.0
- "*ACCT using CLTV - More Effective than a sleeping pill!*", 2016 https://bitcointalk.org/index.php?topic=1340621.0
- BIP 0199: "*Hashed Time-Locked Contract transactions*"
- "BarterDEX – A Practical Native DEX", 2016, https://github.com/KomodoPlatform/KomodoPlatform/wiki/BarterDEX-%E2%80%93-A-Practical-Native-DEX
- "*Advisory: secret size attack on cross-chain hash lock smart contracts*", Dr. Mark B. Lundeburg, 2018, https://gist.github.com/markblundeberg/7a932c98179de2190049f5823907c016
- "Connecting Blockchains: Instant Cross-Chain Transactions On Lightning", 2017, https://blog.lightning.engineering/announcement/2017/11/16/ln-swap.html

Thank you for your attention!!!