

Examining the Lightning network on a protocol level

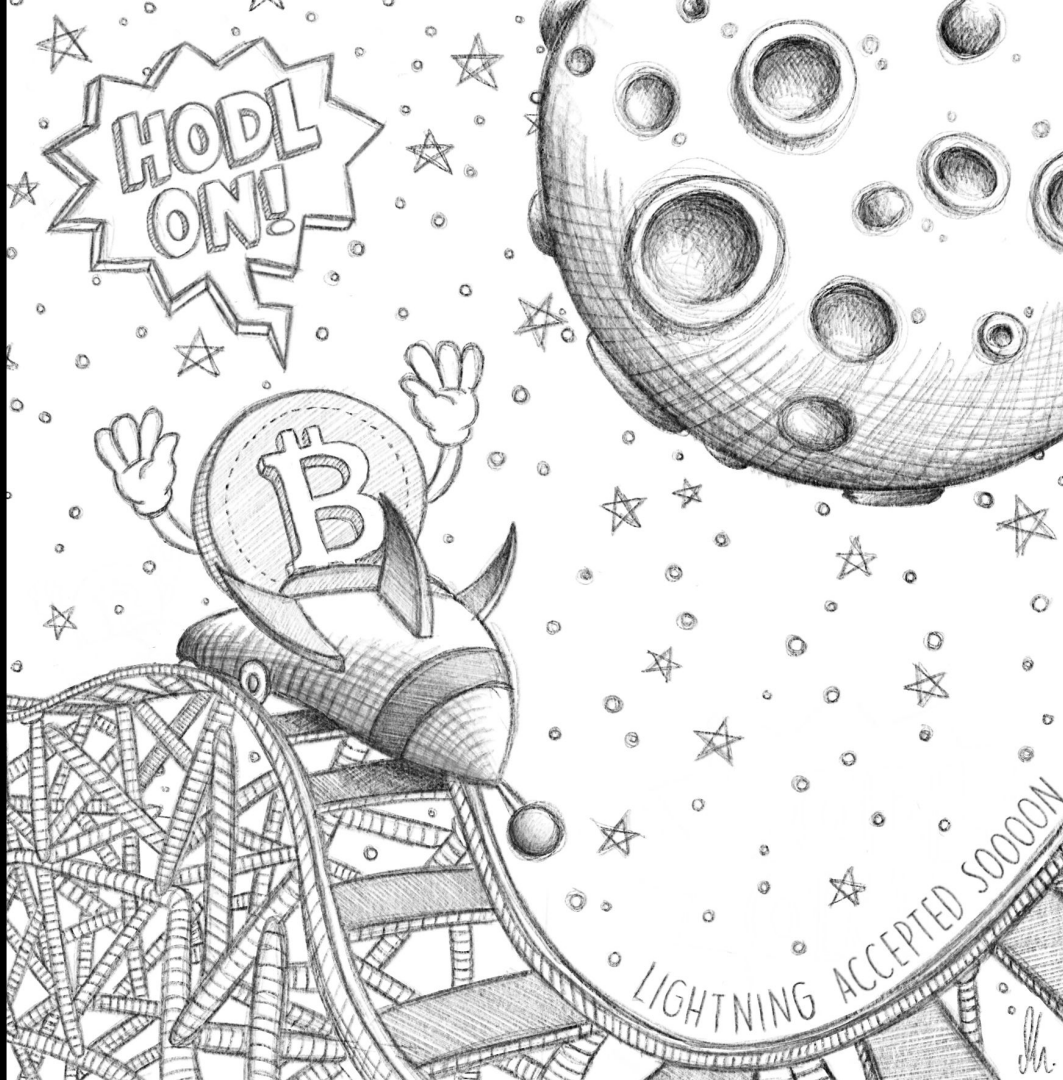
Rene Pickhardt (Data Science Consultant)

<https://www.rene-pickhardt.de> & <https://twitter.com/renepickhardt>

München 19.7.2018

A photograph of two young girls outdoors, engaged in a conversation. The girl on the left has long, dark hair and is wearing a pink and white striped sweater with a white fur collar. She is leaning in and whispering into the ear of the girl on the right. The girl on the right has long, light brown hair and is wearing a bright orange sweater. She is also whispering to someone off-camera to her right. Both girls have their hands covering their mouths, suggesting they are sharing a secret. The background shows trees and a white fence, indicating an outdoor setting like a garden or park.

I've been told I should start any talk with something everybody in the audience should know



HODL ON!

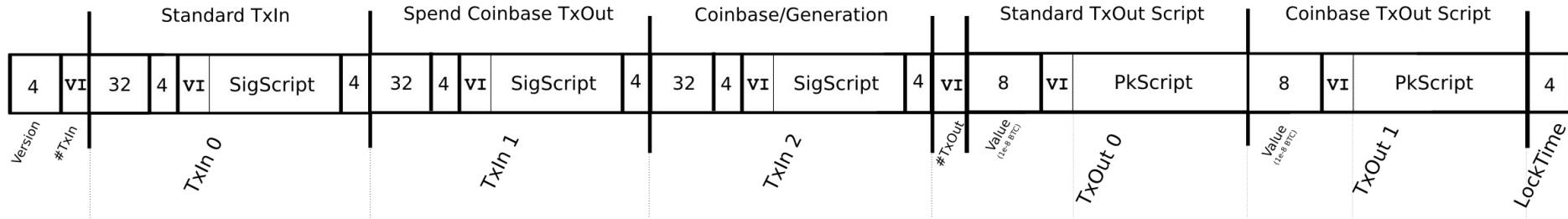
LIGHTNING ACCEPTED SOON

A standard Bitcoin Transaction has 6 data fields

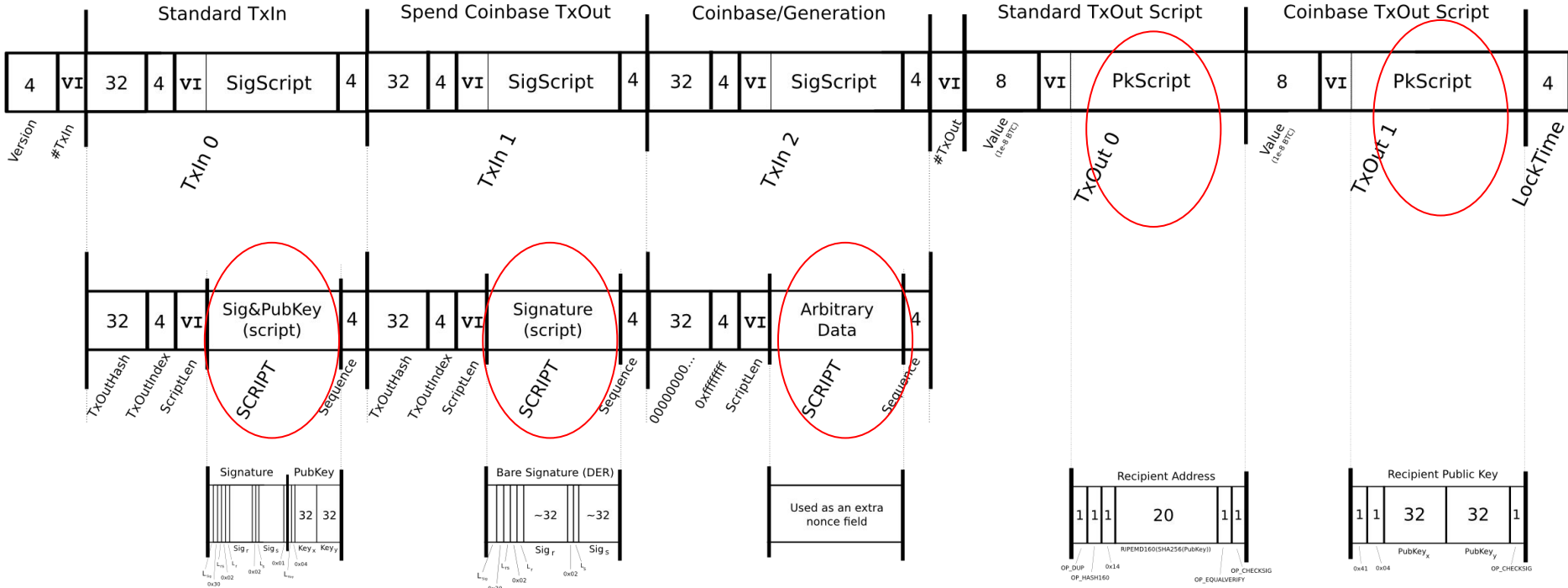
(The following is a brief summary of <https://en.bitcoin.it/wiki/Transaction#Input>)

- Version number (4 Bytes)
- In-Counter (1-9 Bytes)
- List of inputs (depending on the Value of <In-Counter>)
- Out-Counter (1-9 Bytes)
- List of Outputs (depending on the Value of <Out-Counter>)
- Lock_time (4 Bytes)

Bitcoin Transaction with 3 inputs and 2 outputs

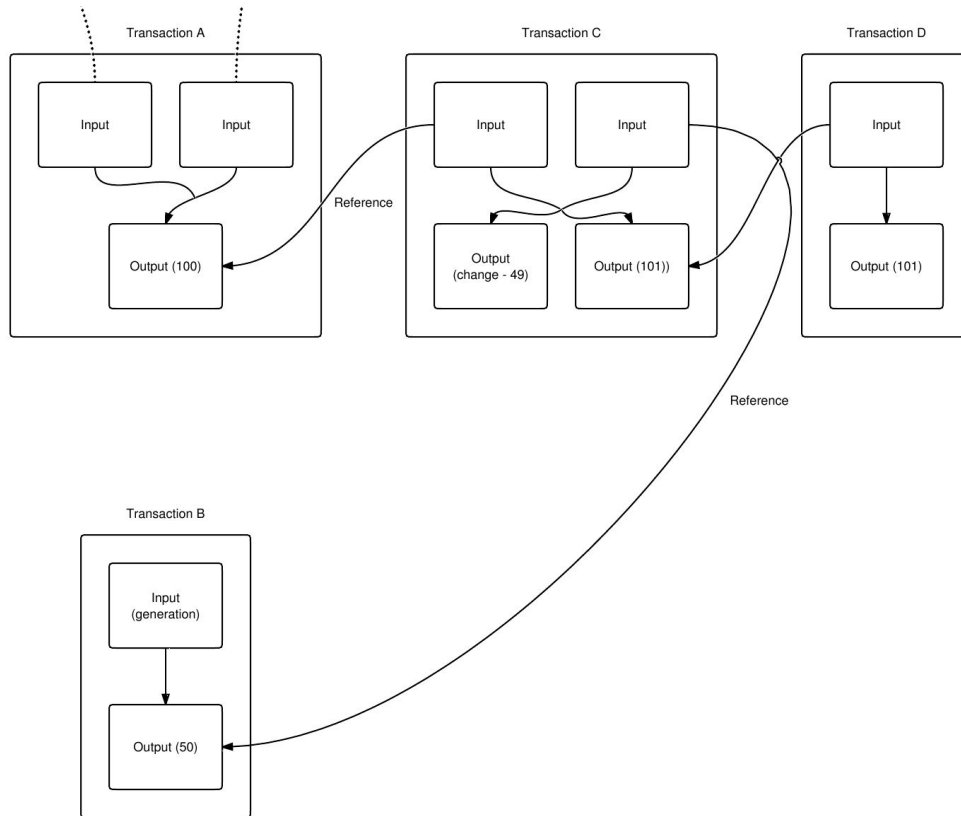


Data consists mainly of executable scripts!



Scripts and DER encoding both use big-endian values, all other serializations use little-endian

A chain of Bitcoin Transactions with 3 UTXO



- Outputs are references by the inputs
- The Script in the output defines how the transaction can be spent
- Owning Bitcoins means being able to spend the output of an unspent transaction
 - Provide an input script
 - Concatenate it with with some outputscript of an unspent transaction
 - The combined Script needs to evaluate to True

Spending a Pay-to-PubkeyHash (standard TX)

- ScriptPubKey (aka the Output Script)
 - OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
- ScriptSig: (aka the Input Script)
 - <sig> <pubKey>
- Explanations
 - OP_CODES are the instructions of the script language within Bitcoin
 - <data> is depicted like html tags with lesser than and greater than signs
 - The complete script is concatenated as Input || Output and then being executed on a Stack machine

<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

Execution of Pay-to-PubkeyHash Script

<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG



Data is pushed on the stack

<sig>

Execution of Pay-to-PubkeyHash Script

~~<sig>~~ <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG



Data is pushed on the stack again



<pubKey>

<sig>

Execution of Pay-to-PubkeyHash Script

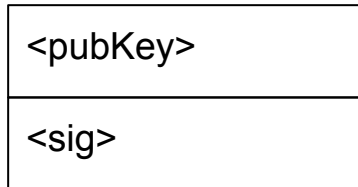
~~<sig>~~ <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG



The OP_CODE OP_DUP is being executed by pushing the top element to the stack again



<pubKey>



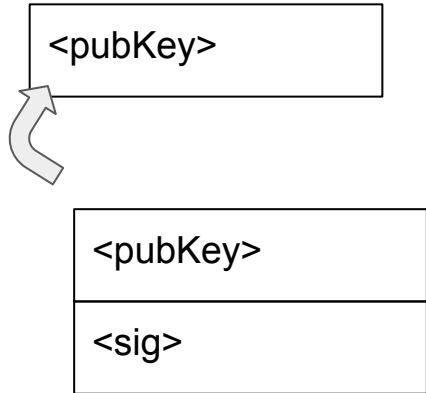
Execution of Pay-to-PubkeyHash Script

~~<sig> <pubKey> OP_DUP~~ OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG



The OP_CODE OP_HASH160 is being executed:

It takes the top element of the Stack



Execution of Pay-to-PubkeyHash Script

~~<sig> <pubKey> OP_DUP~~ OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG



The OP_CODE OP_HASH160 is being executed:

It takes the top element of the Stack
The RIPEMD-160 Hash is calculated



<pubKey>

<pubKeyHash>

By design of the input and output script this will be the Hash of the Public Key (aka the Bitcoin Address)



<pubKey>
<sig>

Execution of Pay-to-PubkeyHash Script

~~<sig> <pubKey> OP_DUP~~ OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

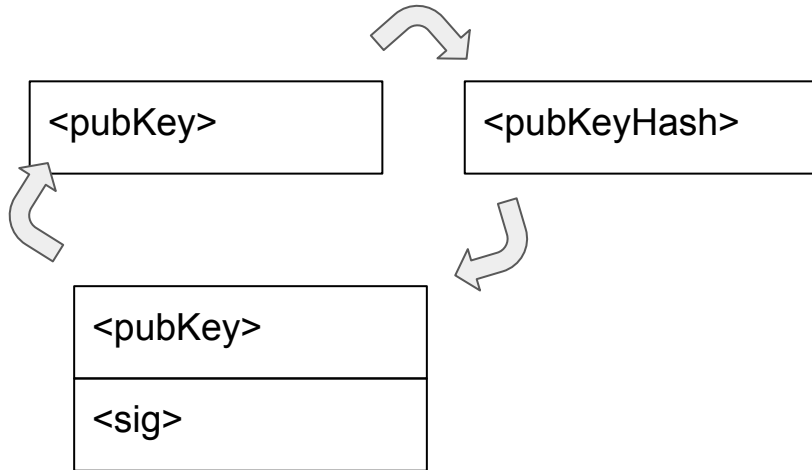


The OP_CODE `OP_HASH160` is being executed:

It takes the top element of the Stack

The RIPEMD-160 Hash is calculated

And the result is pushed back on the stack



By design of the input and output script this will be the Hash of the Public Key (aka the Bitcoin Address)

Execution of Pay-to-PubkeyHash Script

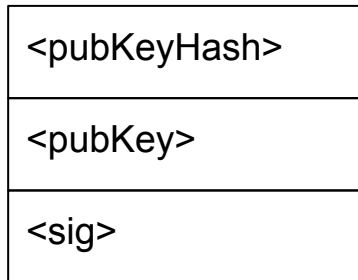
~~<sig> <pubKey> OP_DUP OP_HASH160~~ <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG



Data is pushed on the stack again



<pubKeyHash>



Execution of Pay-to-PubkeyHash Script

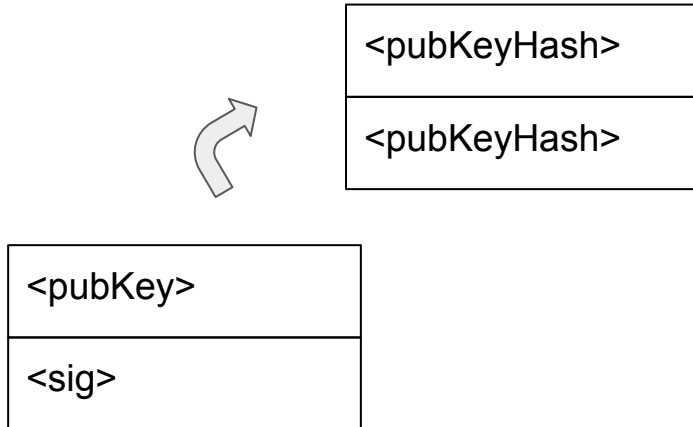
~~<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash>~~ OP_EQUALVERIFY OP_CHECKSIG



OP_EQUALVERIFY checks if the two top elements of the stack are equal.

While checking the elements will be removed

Like the instruction set in the ALU of the CPU OP_CODES know how much data they need to consume (and will do so or fail otherwise)



Execution of Pay-to-PubkeyHash Script

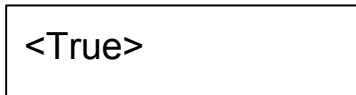
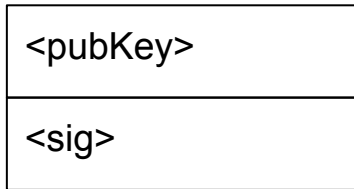
~~<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG~~



OP_CHECKSIG evaluates that the PubKey which was the top element fits to the signature which was the 2. Element on the stack

If this fits together ownership is proven and the Transaction will be added to the block with a new Output script.

After being mined the transaction is now considered to be spent. Double spending cannot take place since the assumption was that the output was not spent yet.



Broadcast / Publish that Transaction for
the network to verify it



The Bitcoin miners will build the next block

- Collect published and valid transactions
 - In particular the hashes of the transactions
 - Most likely those that offer highest mining fees
 - Mining fees are leftovers of inputs which have not assigned to outputs
 - As many as there is space within the block
 - Start with one special transaction which has no input generating the block reward as an output
- Take some additional meta data
 - The hash of the previous block
 - The nonce (and extra nonce as part of the coinbase)
- Compute the merkle tree of all the hashes
- Look out for a hash collision with respect to the mining difficulty
- If your Tx was in: Congratulations you now sent bitcoins

A person with a beard, wearing a purple long-sleeved shirt, blue shorts, and yellow climbing shoes, is climbing a steep, dark grey rock face. They are using a rope for safety. The background shows a rugged mountain landscape with green vegetation and rocky terrain.

Be Brave!

Now is the moment to
ask questions



Let's talk lightning!

A night photograph of a city skyline with a large, bright lightning bolt striking the sky. The lightning bolt is the central focus, branching out across the dark sky. The city lights are visible in the background, and the overall scene is dark and dramatic.

I love taking photographs of lightning bolts...

Well known properties of the Lightning Network

- Backed by the Bitcoin Blockchain (and nothing else)
 - No the Lightning Network is not an ICO!
 - However it can be "easily" extended to other blockchains
- Fast / instant payments
 - Transactions are possible without waiting for blockchain confirmations
- Low fees
 - This makes micro payments possible
- Trustless (actually it trusts the bitcoin blockchain)
 - No need to trust any custodian or third party
 - Or even your channel partner
- Be your own bank
 - You own your funds at any given time and have full control (including all the risk)
- Decentralized (Frequently people fear it could become centralized due to hubs)
 - Existence of super hubs doesn't make it centralized!

Two main components lead to the Lightning Network

- Bidirectional payment channel
 - 3 known methods to construct those
 - **Poon Dryja Channels** (Penalty Revocation based system)
 - Eltoo (SIGHASH_NOINPUT requires Bitcoin softfork)
 - Decker Wattenhofer (invalidation trees)
 - Revocable Sequence Maturity Contracts (RSMC)
 - OP_CHECKSEQUENCEVERIFY aka OP_RELATIVECHECKLOCKTIMEVERIFY
 - Thanks (!) to BIP 112 <https://github.com/bitcoin/bips/blob/master/bip-0112.mediawiki>
- Routing payments through a network of payment channels
 - Hashed time locked contracts (HTLC)
 - Preimages of Hash as Secrets to trigger payments

Purpose of a trustless bidirectional payment channel

- Sending payments between two partners
 - Instantly
 - Like updating a balance sheet
 - Only bound by network traffic over the internet
 - No direct involvement of the blockchain (as long as everyone plays by the rules)
- No need for any partner to trust the other side
 - Again The Blockchain CAN be involved if problems arise
- No fees or overhead when sending payments within the channel
- Payments can go in either direction back and forth

→ How can this technically be possible?

A comparison with HTTP

- HTTP is a Request Response Protocol by design
 - You (the client / Browser) can make a request
 - The server gives you a response
- According to the protocol a server cannot initiate communication with a client

- How can a website (e.g. Twitter) give us a notification that some new data is there?
 - By abusing the protocol
 - aka: Server push, long polling,
 - Took us about 10 years to figure that out

How does a HTTP server push work?

- You load a web page
- Javascript (AJAX) initiates an HTTP request
- Server looks if he has some information for the client
- If yes → response
- If no → defer answering to that request to some time in the future
 - If new information for the client is available
 - If client opens a new page and can't receive that response anymore
 - Just as a heartbeat mechanism to ask the client to make a new request
- From an end user perspective the server has initiated a conversation
 - The end user is usually not aware of the fact that an AJAX Request is outstanding

More details at: https://en.wikipedia.org/wiki/Push_technology

Payment channels are constructed by deferring the publication or broadcast of some TXs to the future

- Bitcoin is not a request response protocol
 - Rather a broadcast / gossip protocol
- What exactly is being deferred to the future?
 - Transactions spending outputs are held back from being broadcasted
 - Later Transactions are purposely tried to be double spent
 - Obviously only one of the transactions can be actually spent
 - Lightning is about making sure it is the correct one
 - We look at this in the next couple slides
- Payment channels are to Bitcoin what HTTP Server Push is to HTTP
 - Just a non obvious use of the Bitcoin protocol
 - It took us a little while to figure this feature of Bitcoin out (as for HTTP)

A payment channel is a 2-2 multisig Address together with some (smart?) contract

- A 2-2 multisig Address means that the output script of a transaction requires 2 keys in order to be able to spend the transaction
- It is a form of Pay to Script Hash
 - scriptPubKey (OUTPUT): OP_HASH160 <scriptHash> OP_EQUAL
 - scriptSig (INPUT): ...<signatures>... <serialized script>
- Case of a m-of-n multi-signature
 - The OUTPUT script (scriptPubKey) as above
 - OP_HASH160 <scriptHash> OP_EQUAL
 - The INPUT Script to spend the scriptPubKey looks like this:
 - scriptSig: 0 <sig1>...<serialized script>
 - Script: OP_m <pubKey1> ... OP_n OP_CHECKMULTISIG

**U
N
P
U
B
L
I
S
H
E
D**

Output x
(100 mBTC)
Alice's Key

Reference

Funding Transaction

Input (100 mBTC)


Output 0
(100 mBTC)
Alice's & Bob's Key


Inside


ScriptPubKey:
OP_HASH160<scriptHash> OP_EQUALVERIVY

Legend for colors:

 Broadcasted and mined

 Not boradcasted

 Cannot by minded / should not be broadcasted

 Should be broadcasted

Output x
(100 mBTC)
Alice's Key

Reference

Funding Transaction

Input (100 mBTC)

Output 0
(100 mBTC)
Alice's & Bob's Key

Inside

ScriptPubKey:
OP_HASH160 <scriptHash> OP_EQUALVERIVY

sigScript:
<sig1>... <serializedScript>
Script:
OP_2 <Alice><Bob> OP_2 OP_CHECKMULTISIG

Inside

Commitment Transaction 1

Input (100 mBTC)

Output 0
(70 mBTC)
Alice's Key

Output 1
(30 mBTC)
Bob's Key

Reference

U
N
P
U
B
L
I
S
H
E
D

Output x
(100 mBTC)
Alice's Key

Reference

Funding Transaction

Input (100 mBTC)

Output 0
(100 mBTC)
Alice's & Bob's Key

Inside

ScriptPubKey:
OP_HASH160 <scriptHash> OP_EQUALVERIVY

P
U
B
L
I
S
H
E
D

Reference

Inside

sigScript:
<sig1>... <serializedScript>
Script:
OP_2 <Alice><Bob> OP_2 OP_CHECKMULTISIG

Commitment Transaction 1

Input (100 mBTC)

Output 0
(70 mBTC)
Alice's Key

Output 1
(30 mBTC)
Bob's Key

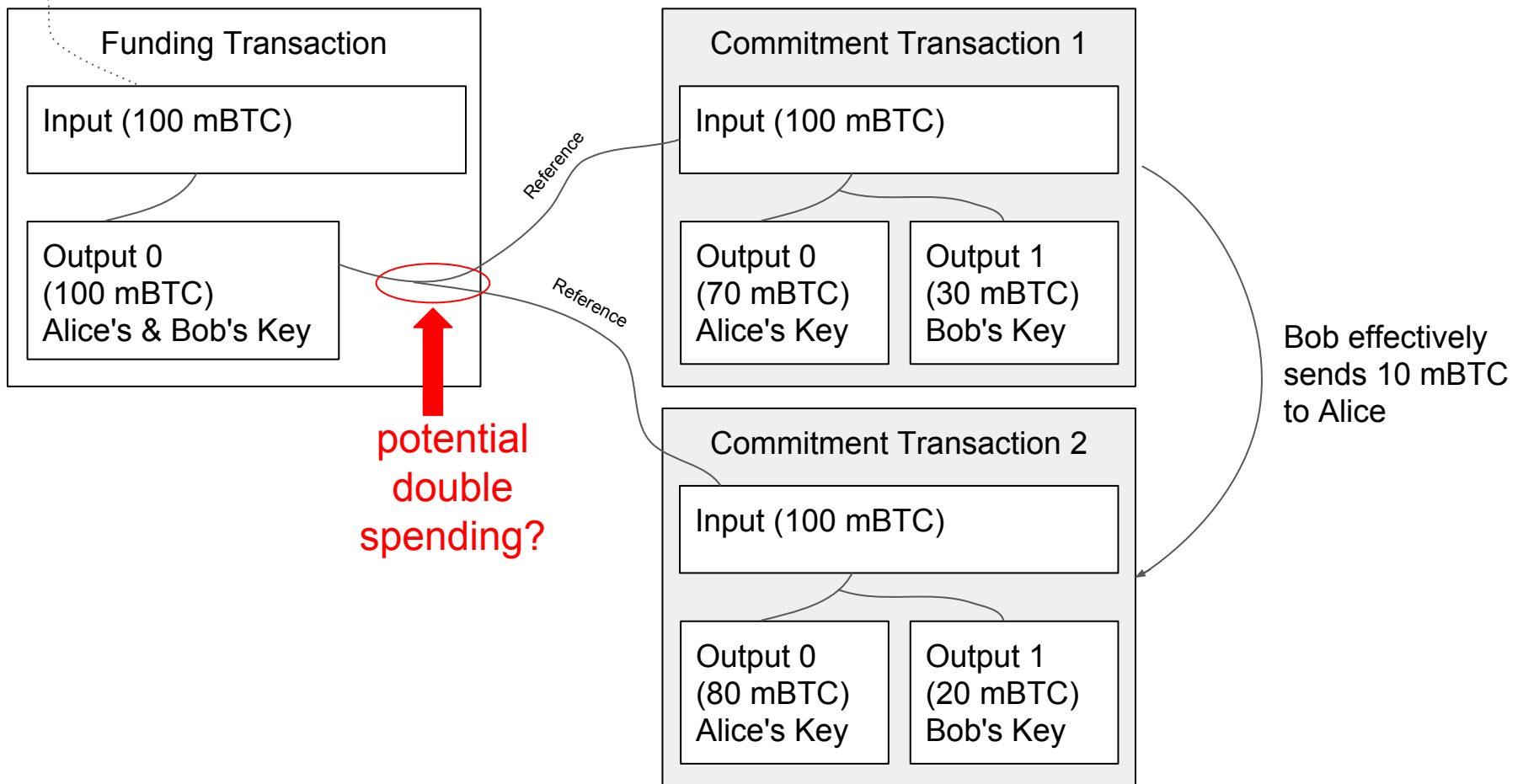
U
N
P
U
B
L
I
S
H
E
D

This Commitment Transaction (CTX1) encodes the balance sheet of the channel.

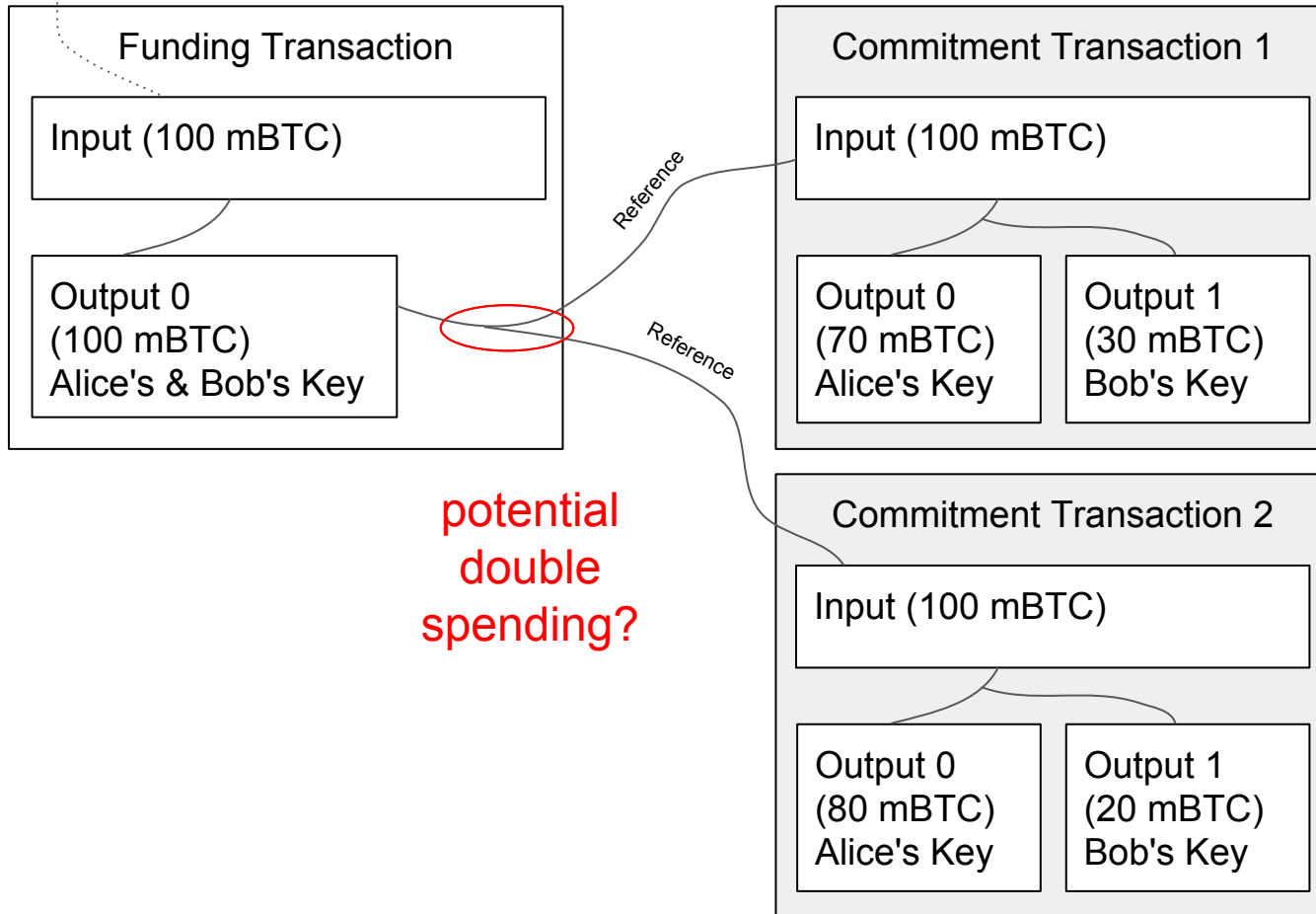
It is called commitment because both sides have already been committed to it.

They can be broadcasted to the blockchain at any time

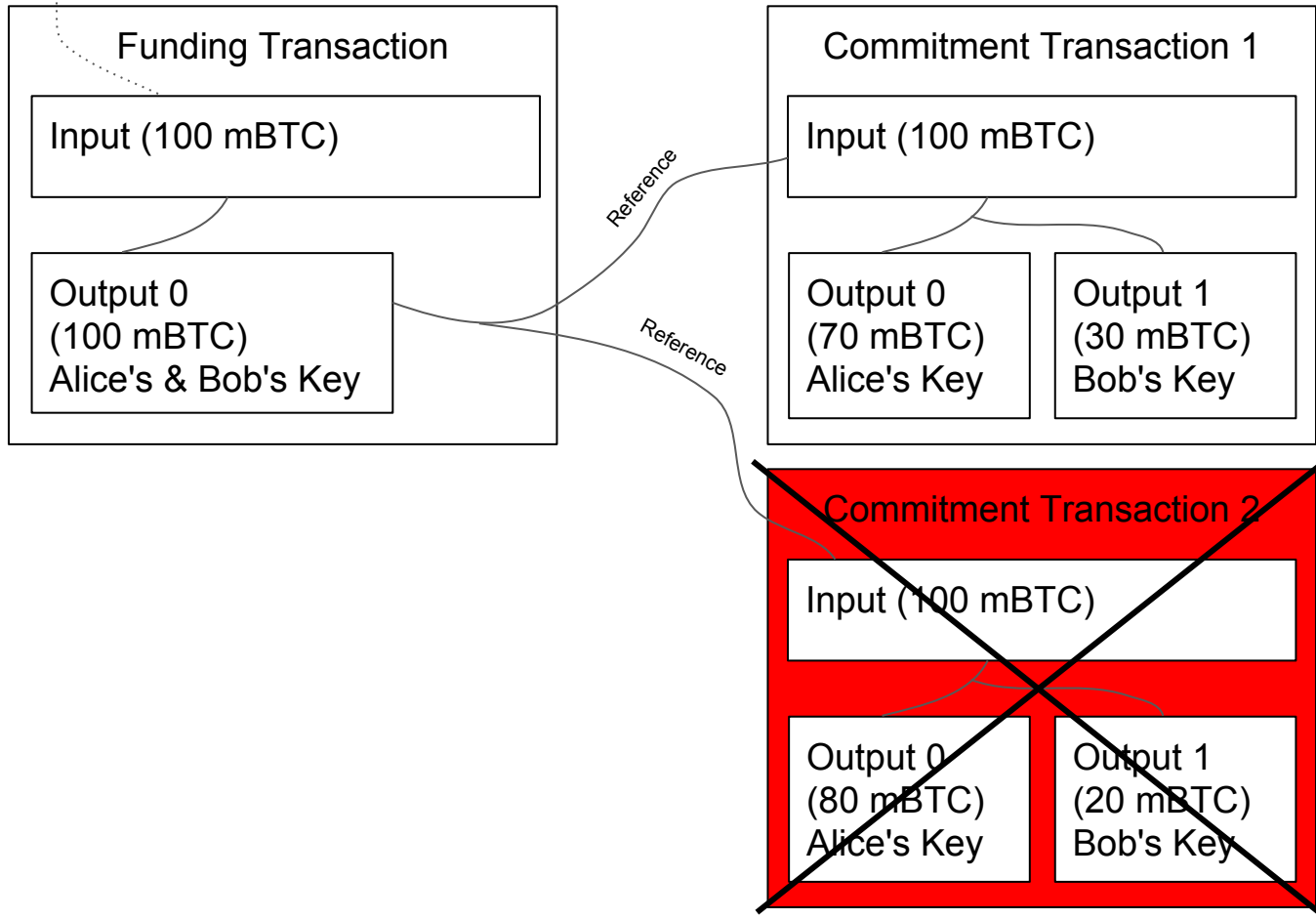
With CTX2 Bob updates the channel Balance



Anybody seeing a problem with this?

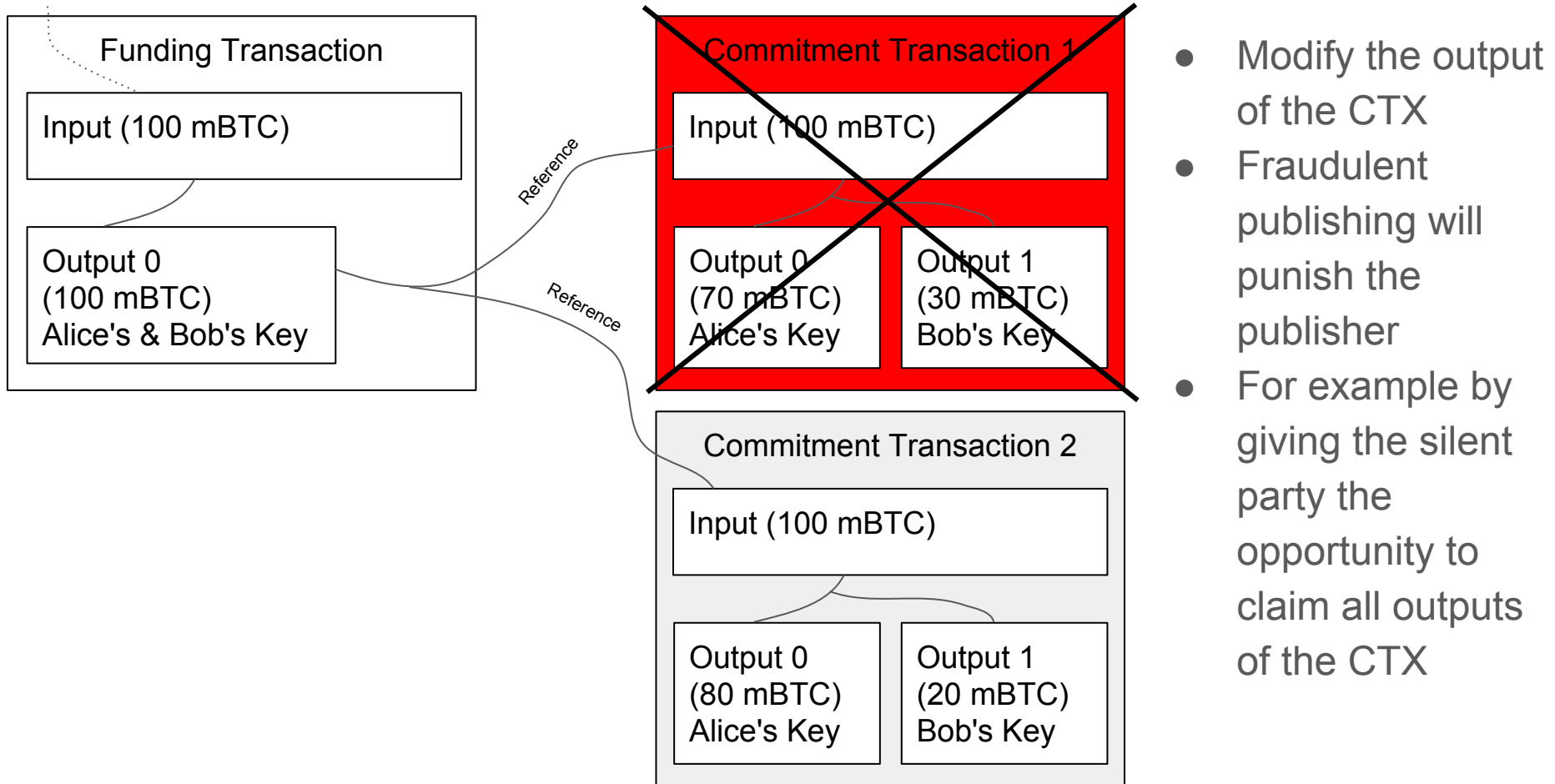


Bob broadcasts CTX1 which is successfully mined



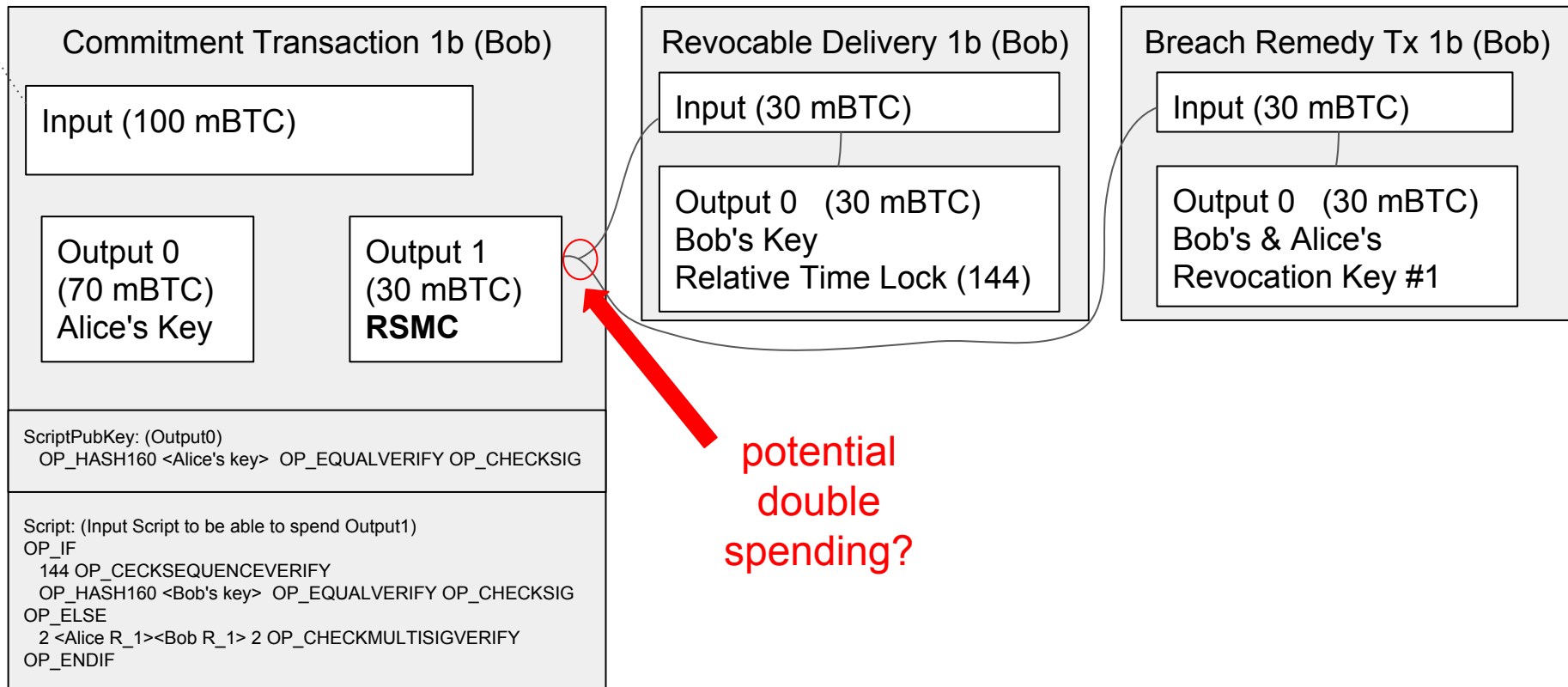
Bob just effectively stole 10 mBTC from Alice

Goal: Make old CTXs somehow unpublishable



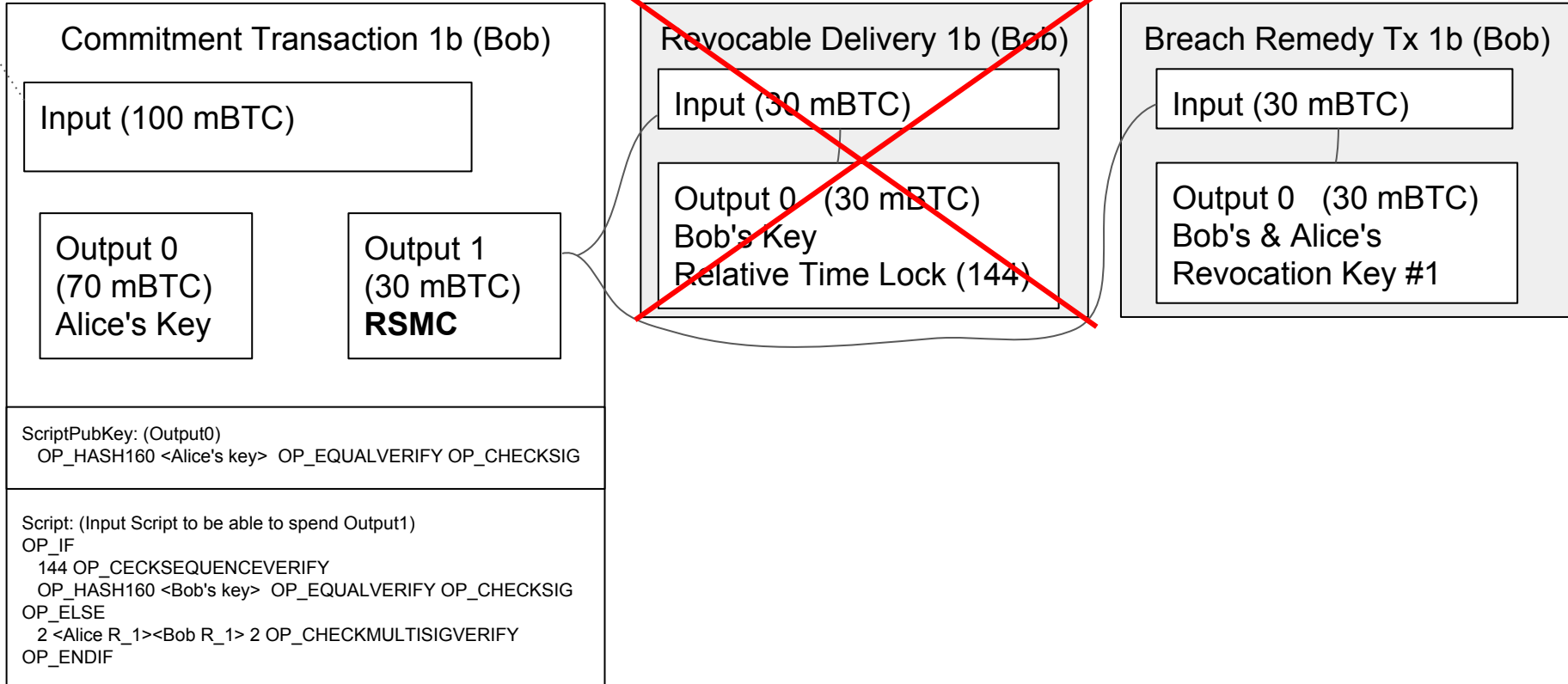
We modify the output scripts of CTXs that it can be spent by either one of the two Transactions

Reference



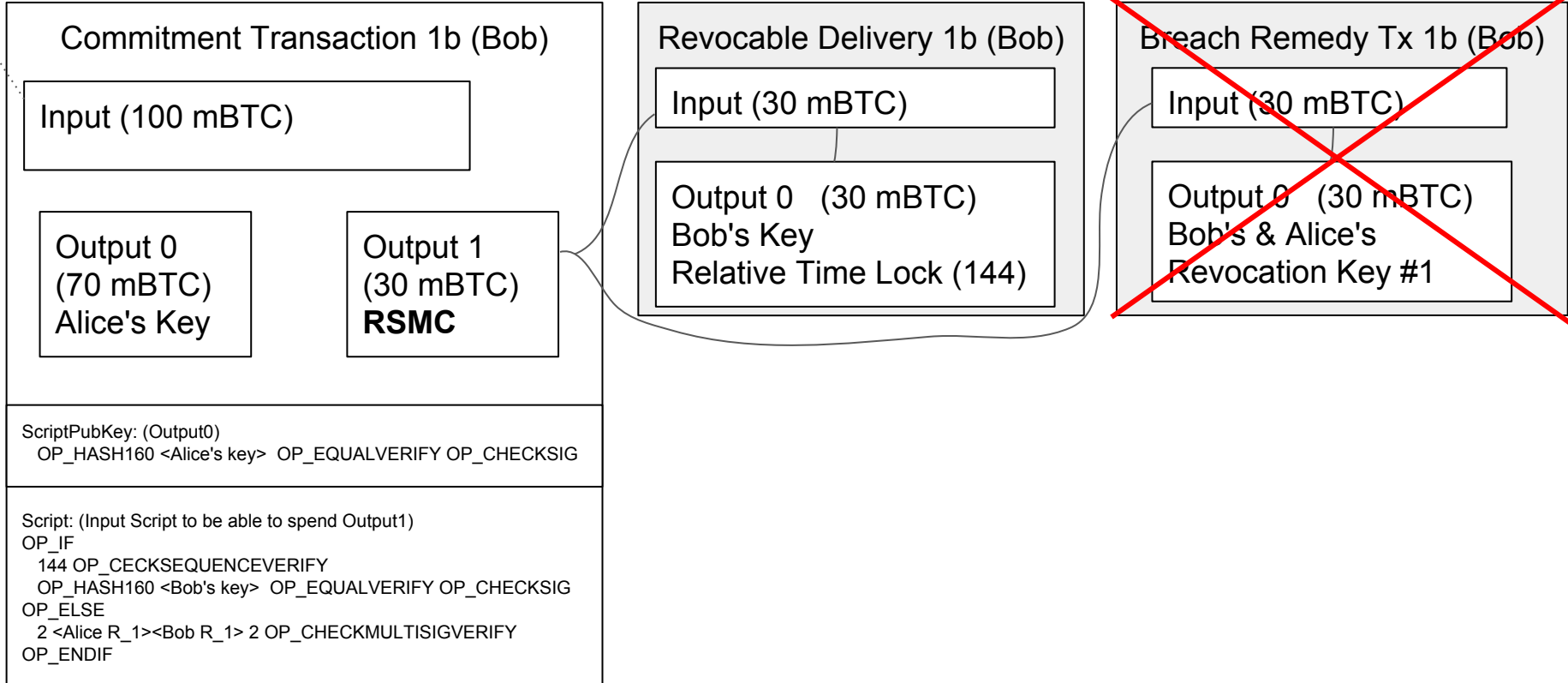
Within 144 Blocks after CTX1b is mined only Alice can spend Output1 (if she has Bob's Revocation Key)

Reference



After 144 Blocks Bob can redeem his 10 mBTC & Alice can't spent BRTX1b without Bob's Key

Reference



Commitment Transaction 1b (Bob)

Input (100 mBTC)

Output 0
(70 mBTC)
Alice's Key

Output 1
(30 mBTC)
RSMC

ScriptPubKey: (Output0)
OP_HASH160 <Alice's key> OP_EQUALVERIFY OP_CHECKSIG

Script: (Input Script to be able to spend Output1)
OP_IF
144 OP_CKCKSEQUENCEVERIFY
OP_HASH160 <Bob's key> OP_EQUALVERIFY OP_CHECKSIG
OP_ELSE
2 <Alice R_1><Bob R_1> 2 OP_CHECKMULTISIGVERIFY
OP_ENDIF

Revocable Delivery 1b (Bob)

Input (30 mBTC)

Output 0 (30 mBTC)
Bob's Key
Relative Time Lock (144)

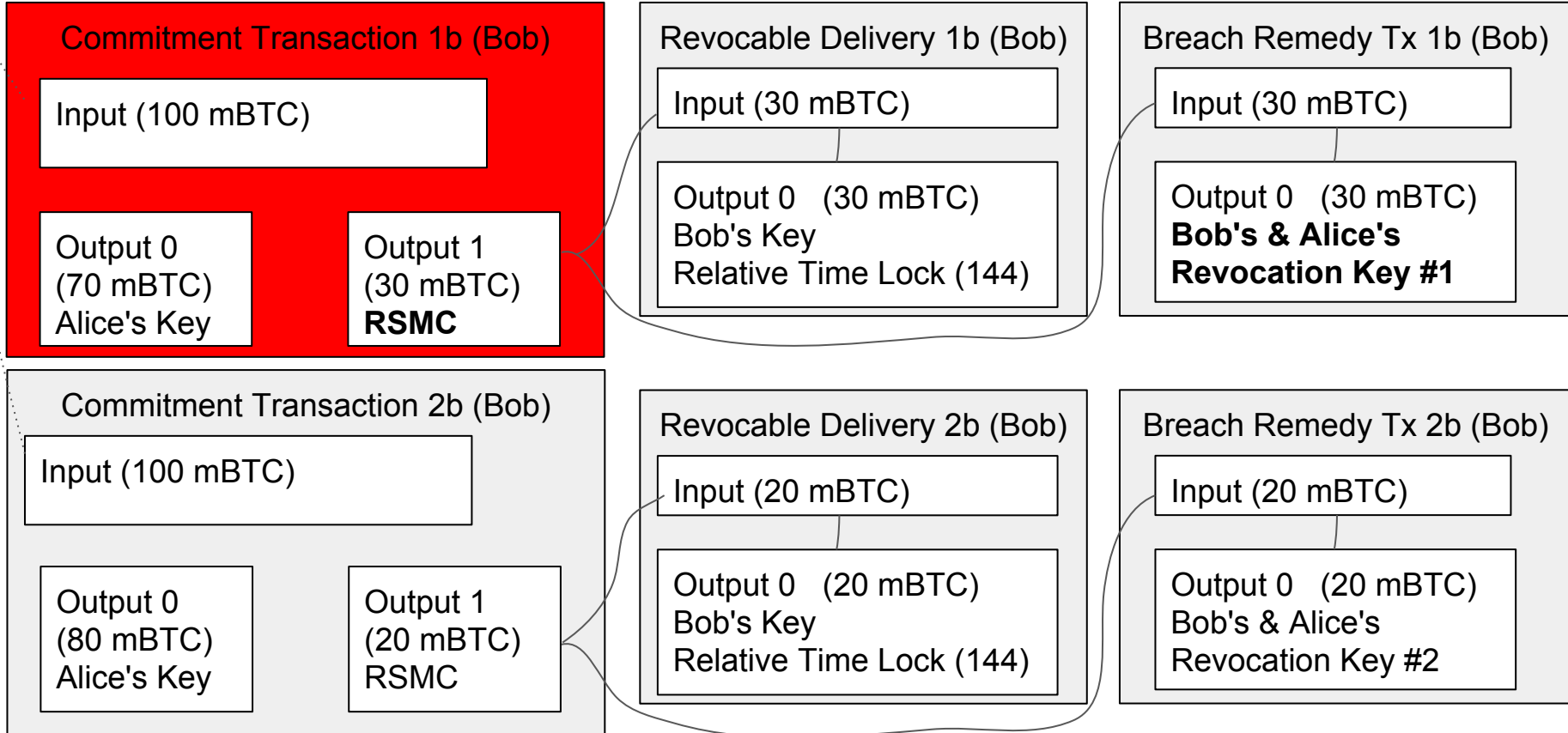
~~Breach Remedy Tx 1b (Bob)~~

~~Input (30 mBTC)~~

~~Output 0 (30 mBTC)
Bob's & Alice's
Revocation Key #1~~

A new Commitment Transaction is only signed if the other party reveals their previous revocation Key

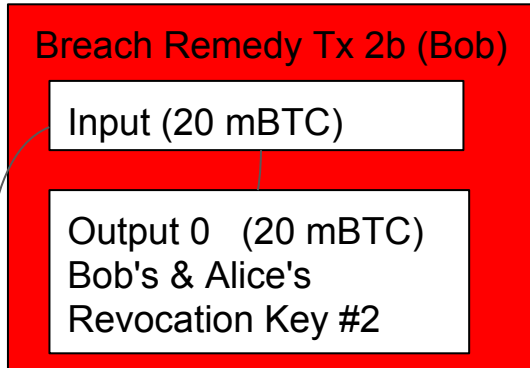
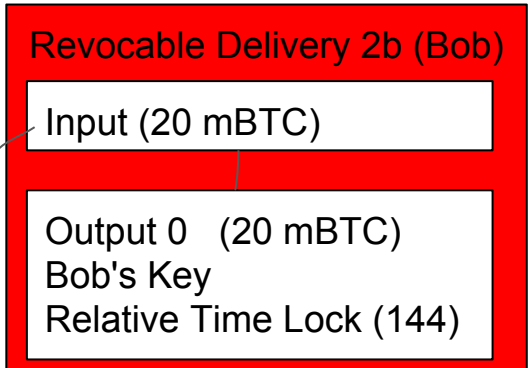
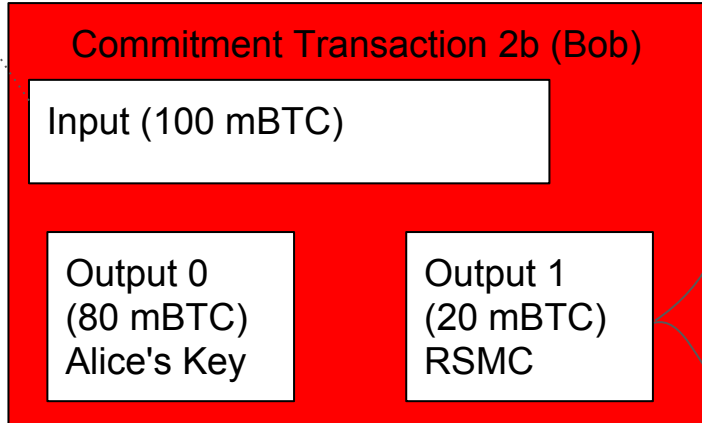
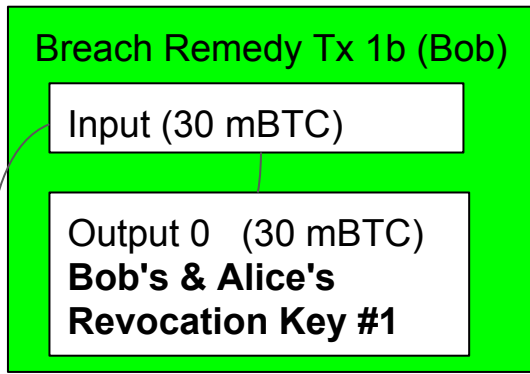
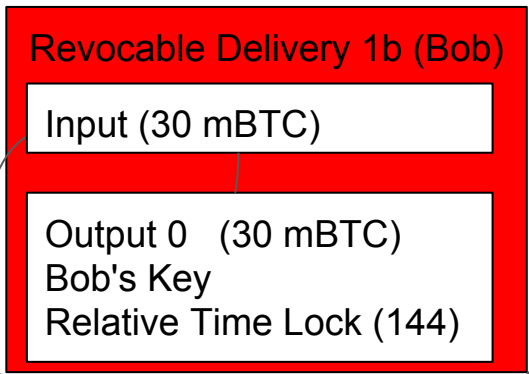
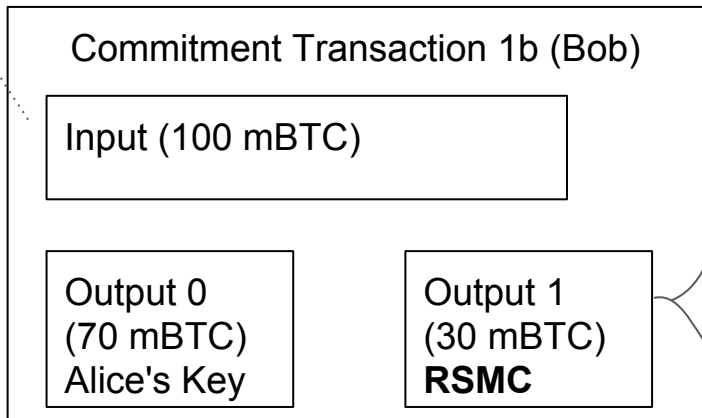
Reference



Assume CTX1b is published and mined. Alice should immediately publish BRTX1b to punish Bob's breach

Reference

IMPOSSIBLE
DOUBLE
SPENDING



Some remarks on presented simplifications

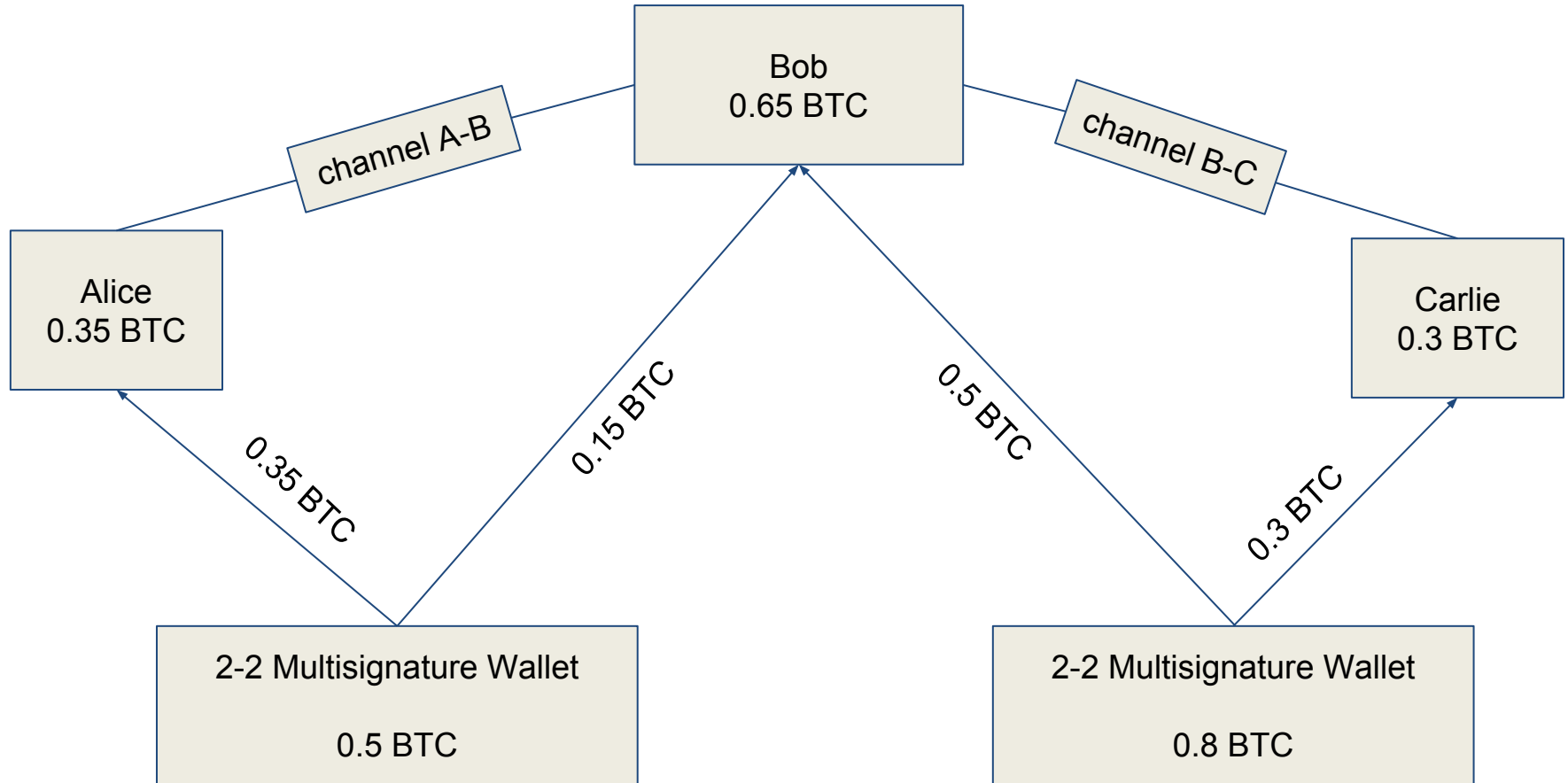
- In order to be able to ascribe blame both sides get their own CTXs
 - Commitment Tx 1b, 2b, 3b,... (for Bob)
 - Commitment Tx 1a, 2a, 3a,... (for Alice)
- Transactions for every channel update
 - The Revocation Deliver TXs and the Breach Remedy TXs are adapted accordingly
- For every Update of the CTXs new Revocation Keys have to be created
 - Otherwise revealing the key once would not help for future updates
 - Future Breach Remedy Transactions could be spent directly
 - They are created from a Hierarchical Deterministic Wallet
- The Commitment Transactions will have even more outputs to support HTLCs
 - HTLC outputs will be needed for routing third party payments through one's channel
- Reading suggestion for a payment channel construction with less Overhead
 - eltoo: A Simple Layer2 Protocol for Bitcoin
 - Christian Decker, Rusty Russell (Blockstream)
 - Olaoluwa Osuntokun (Lightning Labs)
 - Summary at <https://www.rene-pickhardt.de/thoughts-about-eltoo-another-protocol-for-payment-channel-management-in-the-lightning-network/>

Summary of what we have achieved now

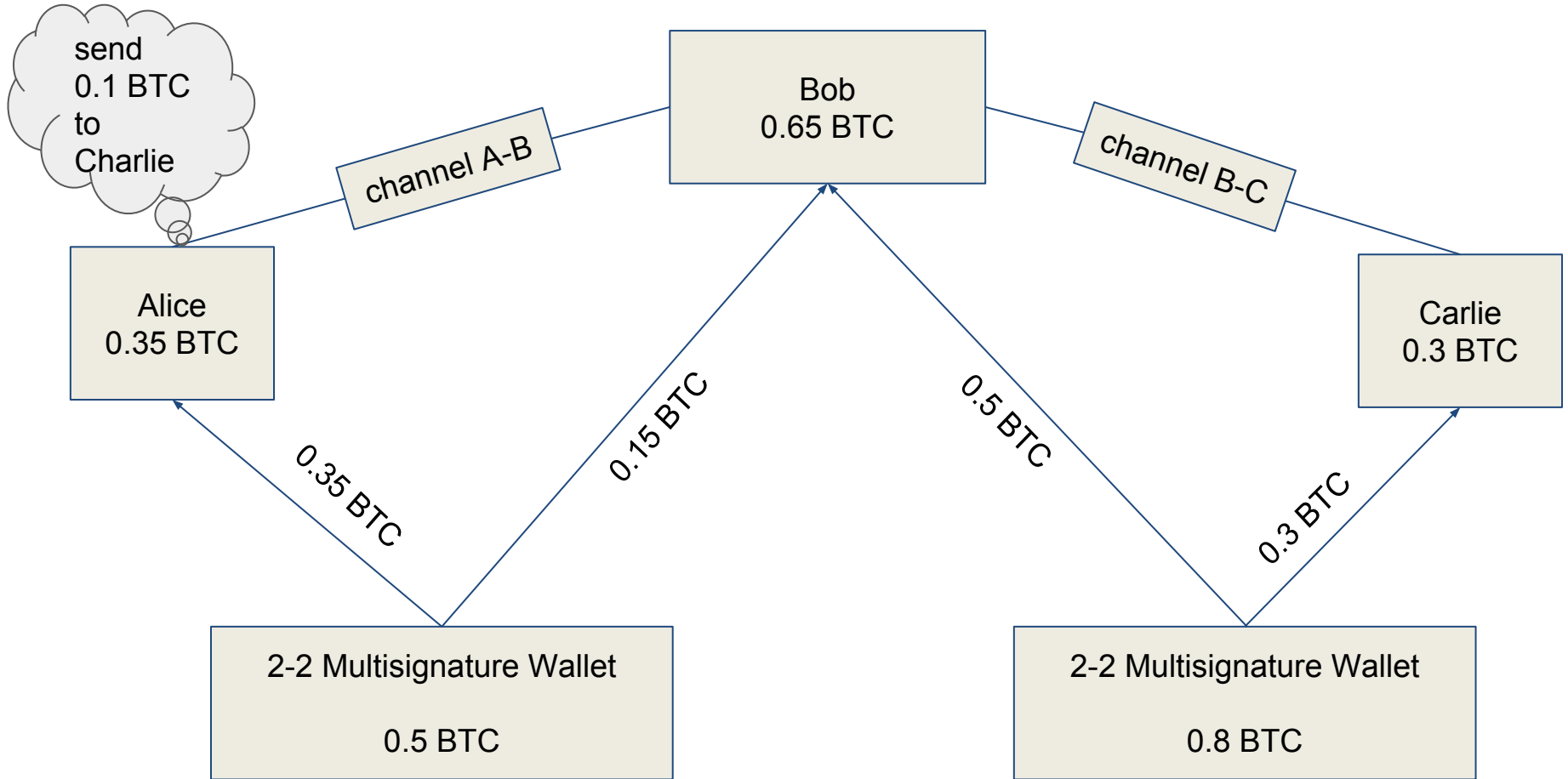
- Bidirectional payment channel is opened with one on chain transaction
- Channel can stay open for arbitrary time
- The newest Commitment Transaction encodes the balance of a channel
- Old Commitment Transactions are invalidated to prevent broadcasting them
 - Achieved by sharing private keys to effectively allow the other side to spend the output
- The balance sheet can instantly be updated
 - Effectively allowing each party to send funds to the other party
 - Both parties need to collaborate for this to happen
 - Blockchain doesn't need to get involved
 - No fees for updating the balance within the payment channel

Where is the (Lightning) Network of payment channels?

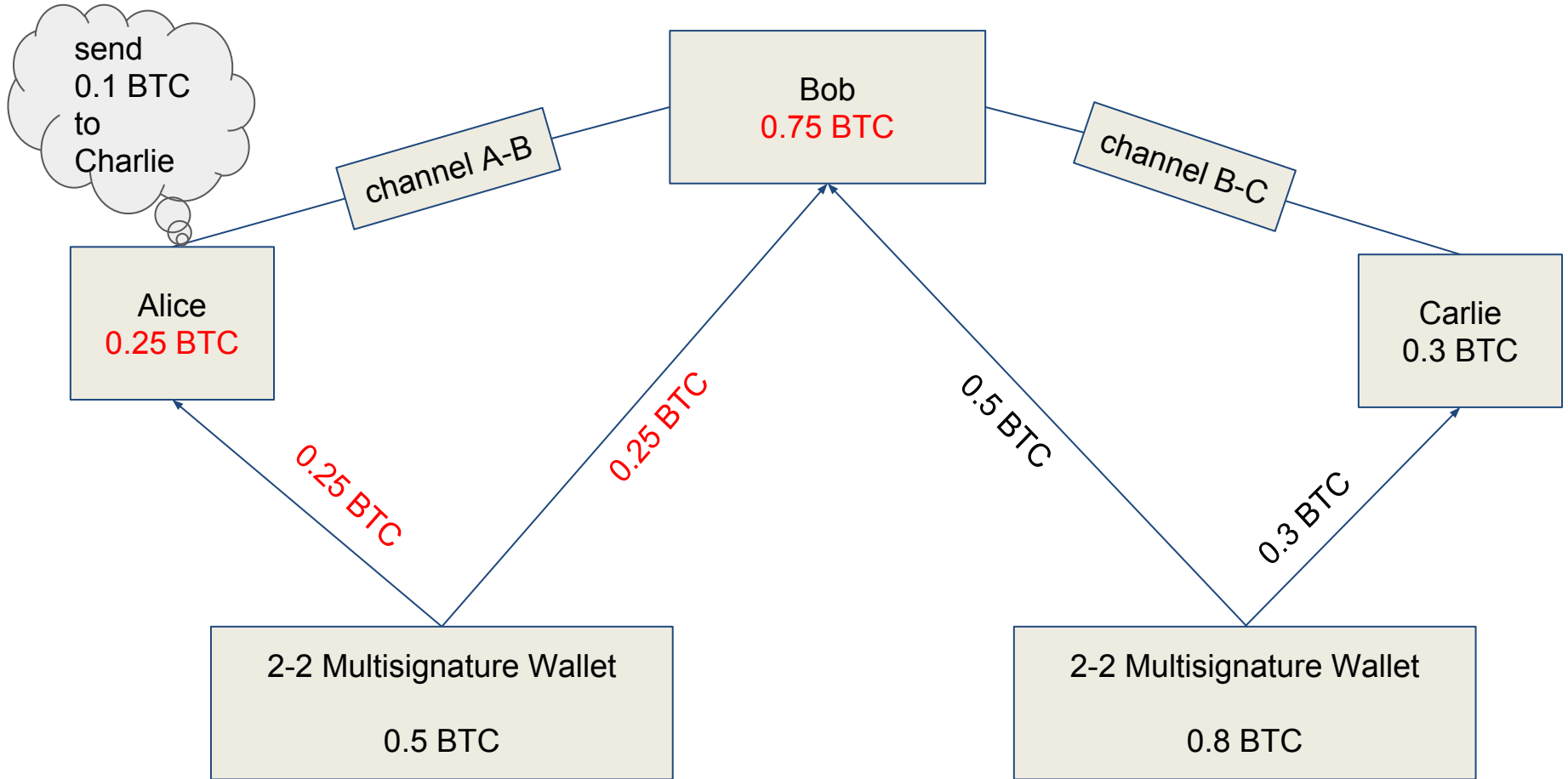
A trusting (Lightning) Network of payment channels



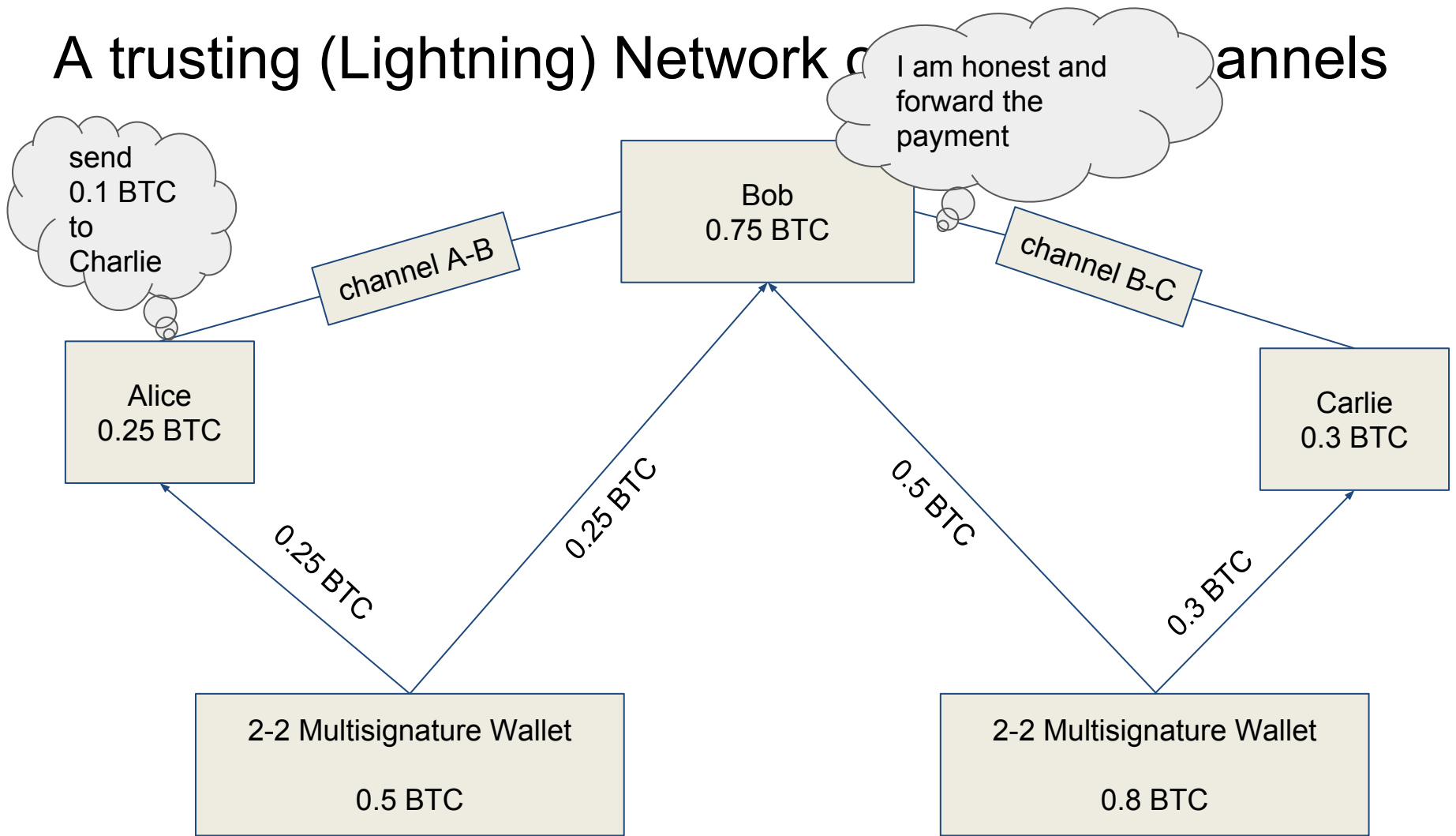
A trusting (Lightning) Network of payment channels



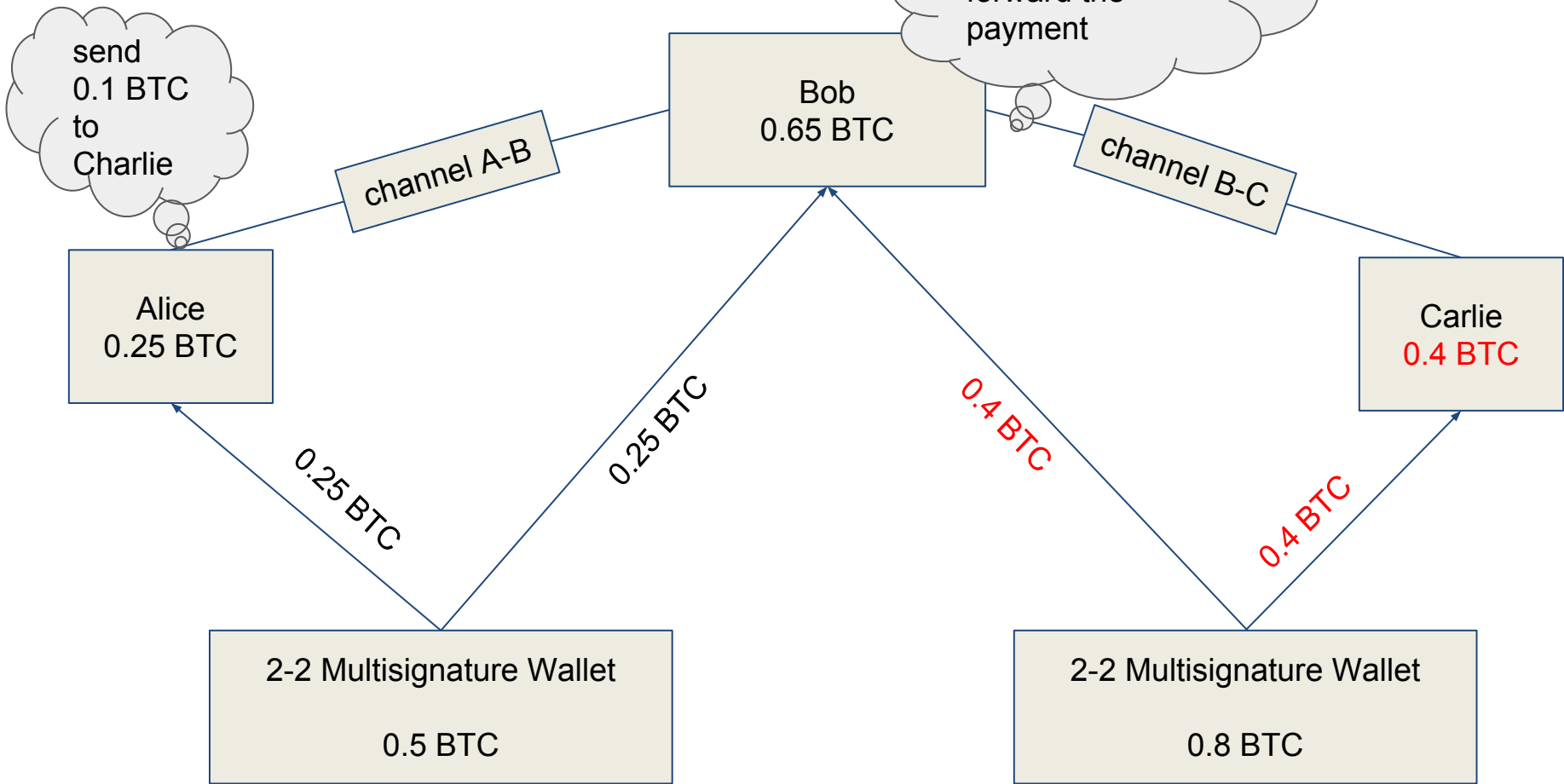
A trusting (Lightning) Network of payment channels



A trusting (Lightning) Network of Channels



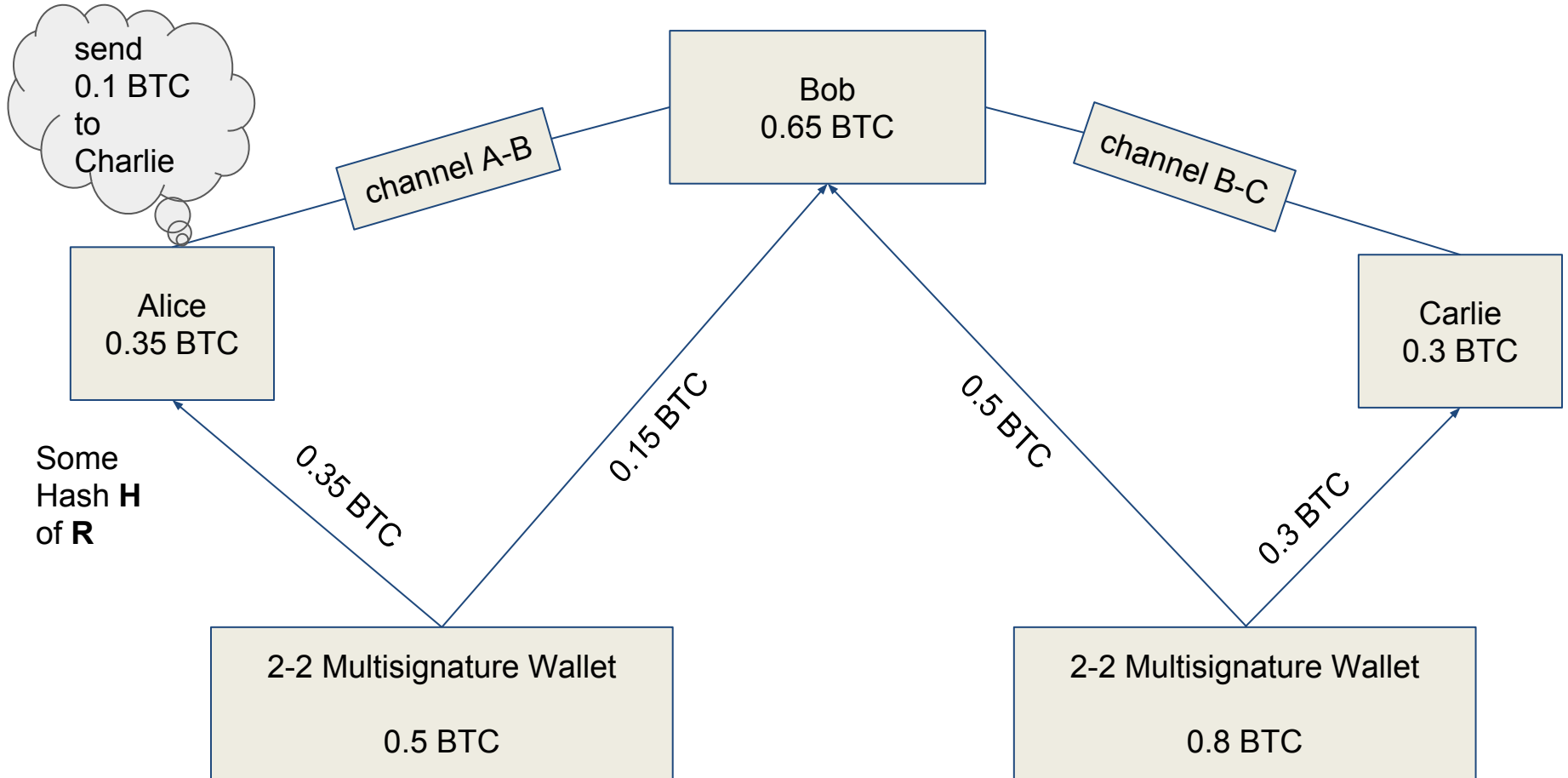
A trusting (Lightning) Network of Channels



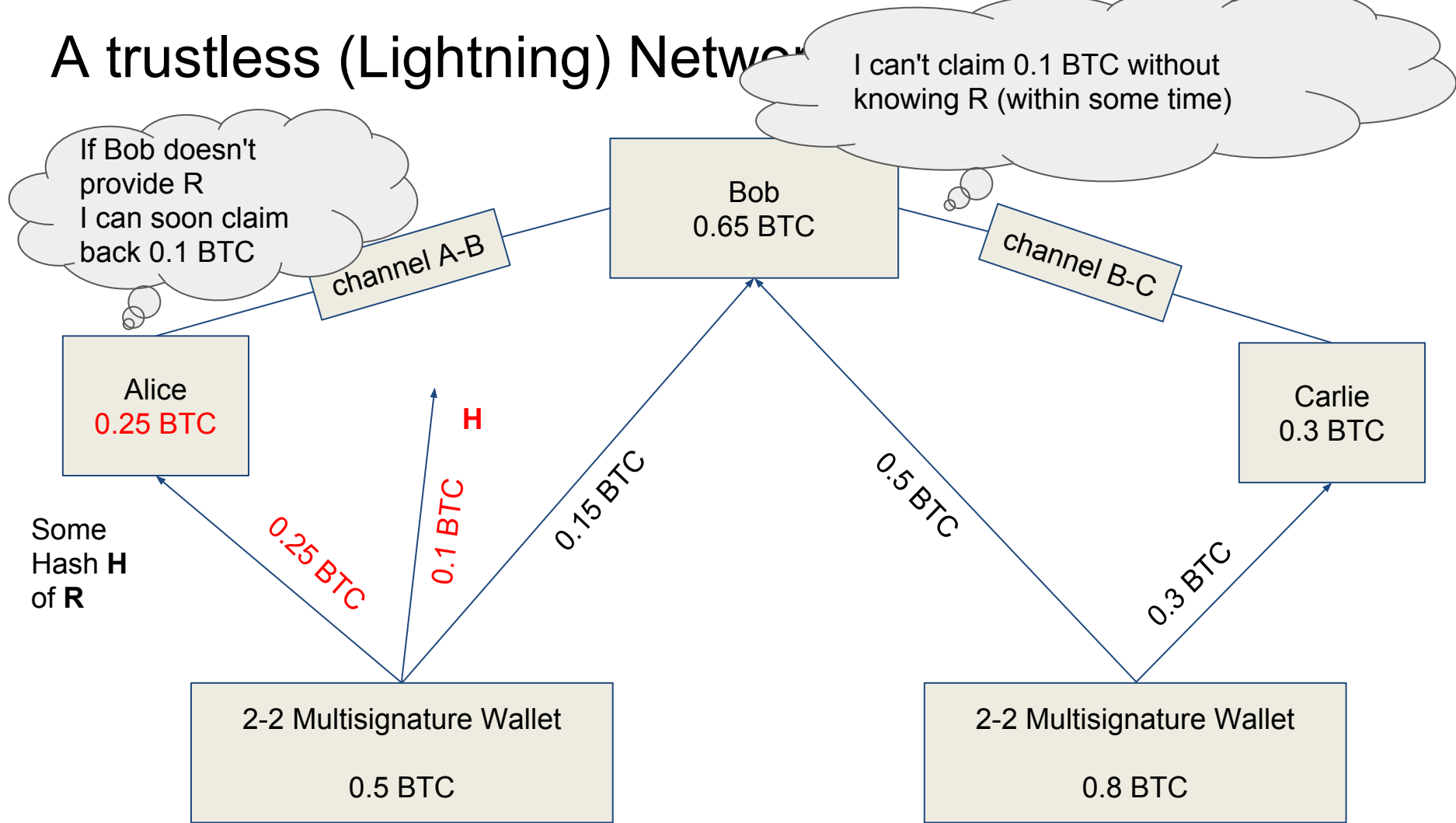
Alice needed to trust Bob to forward the payment

- Both payment channels needed to negotiate new commitment transactions for both sides
- Could we change the output of the CTXs so that routing works trustless?
- Idea (aka Hashed Time Locked Contract - HTLC):
 - Add another conditional output to the Commitment Transactions
 - It can only be spent by the recipient
 - within a certain time frame
 - if the recipient can provide the preimage of some hash
 - After the timeframe the sender can reclaim the funds
- For brevity I refer to the lightning network paper to look up
 - the Script for the sigScript
 - the Breach Remedy Transactions that are attached to this output
 - You can trust me (: It is similar to what we have seen before

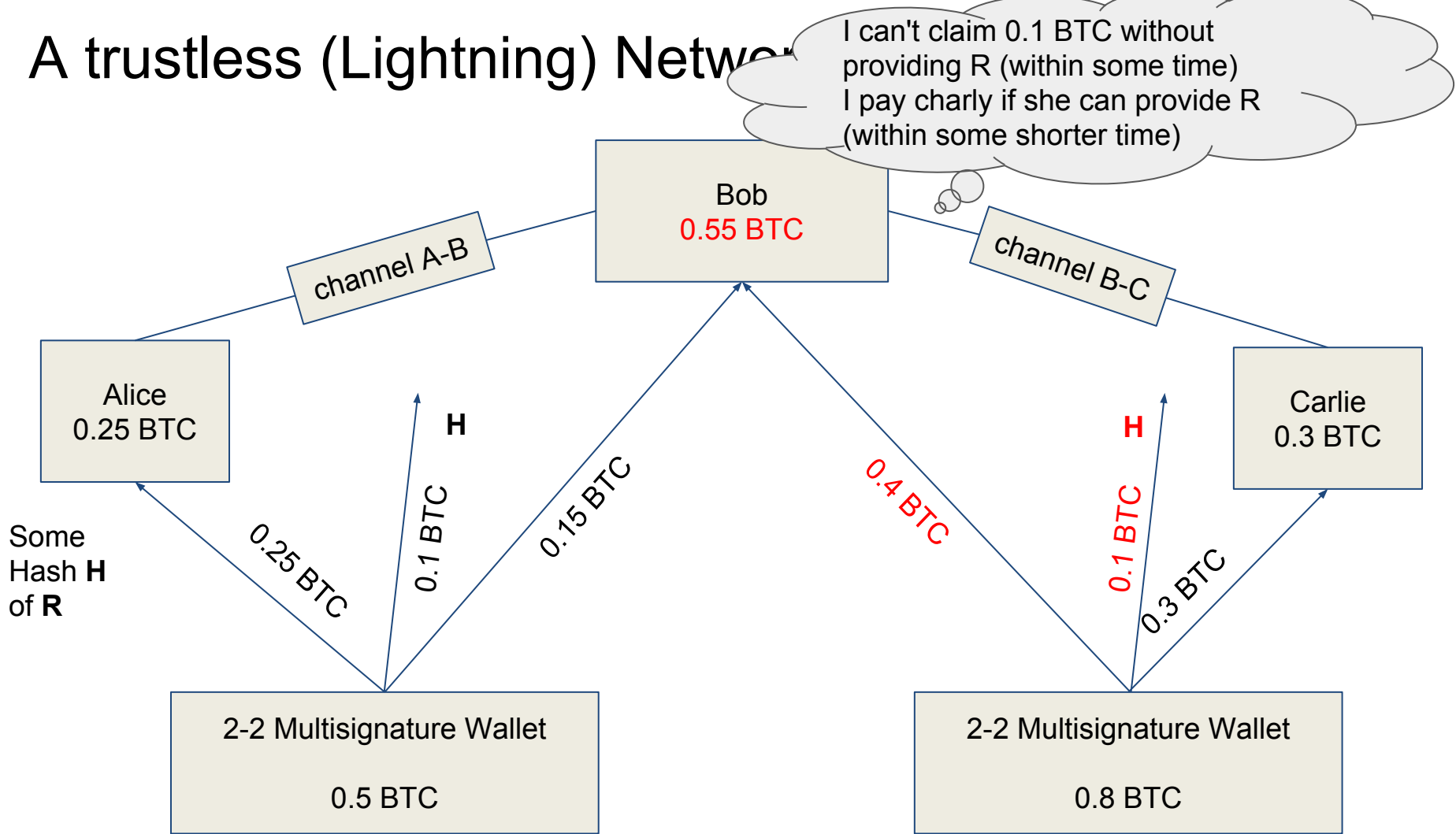
A trustless (Lightning) Network of payment channels



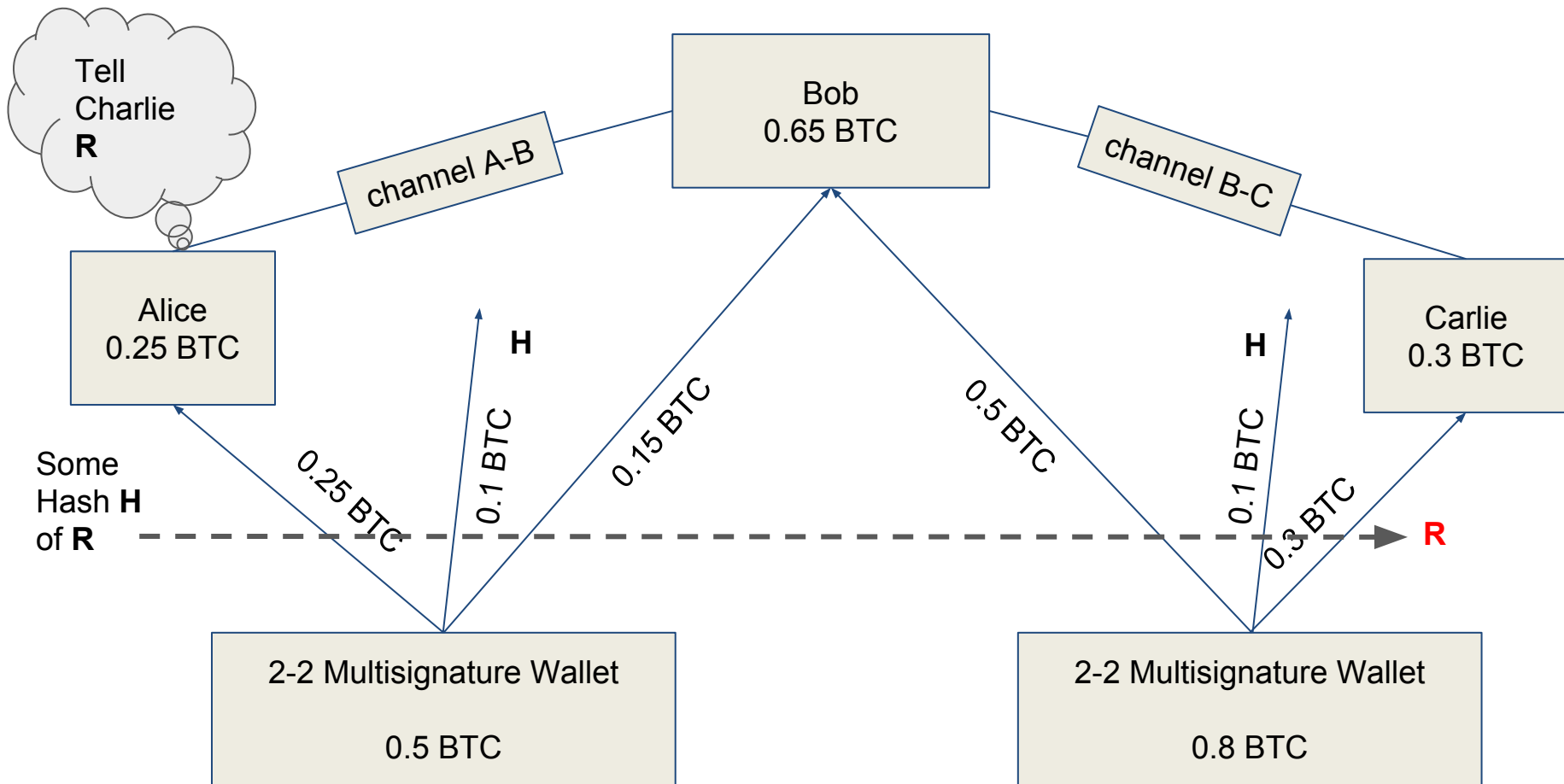
A trustless (Lightning) Network



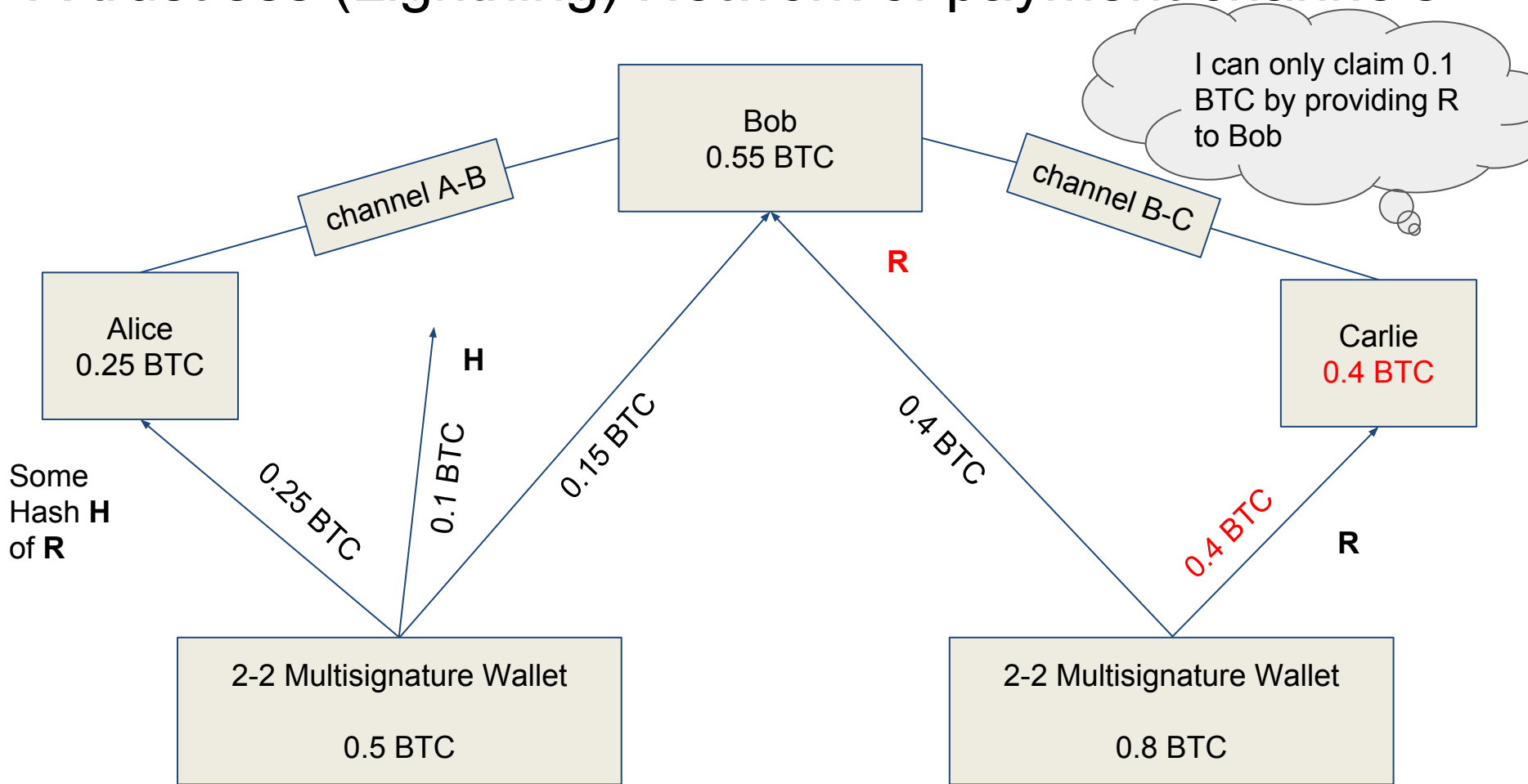
A trustless (Lightning) Network



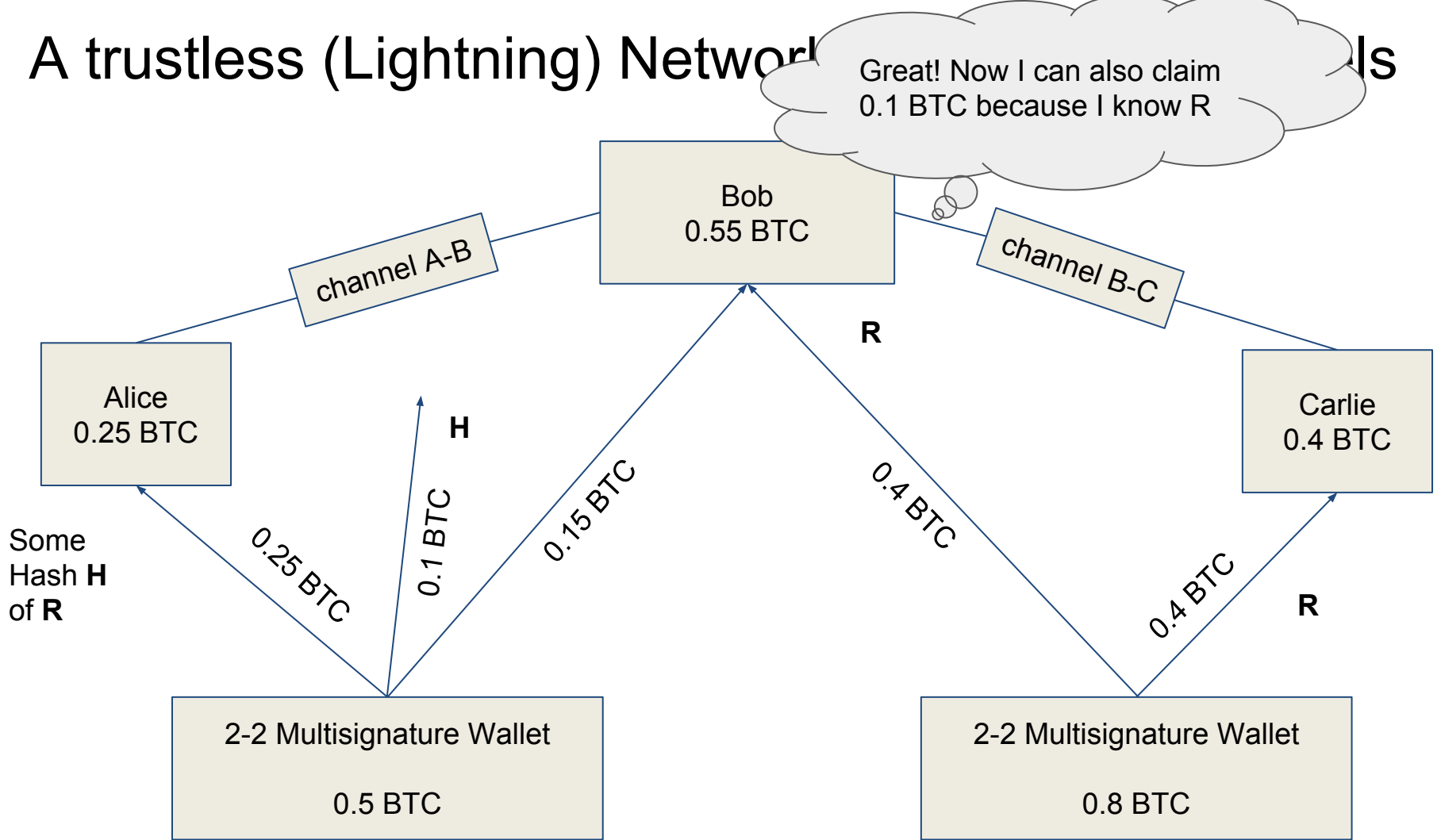
A trustless (Lightning) Network of payment channels



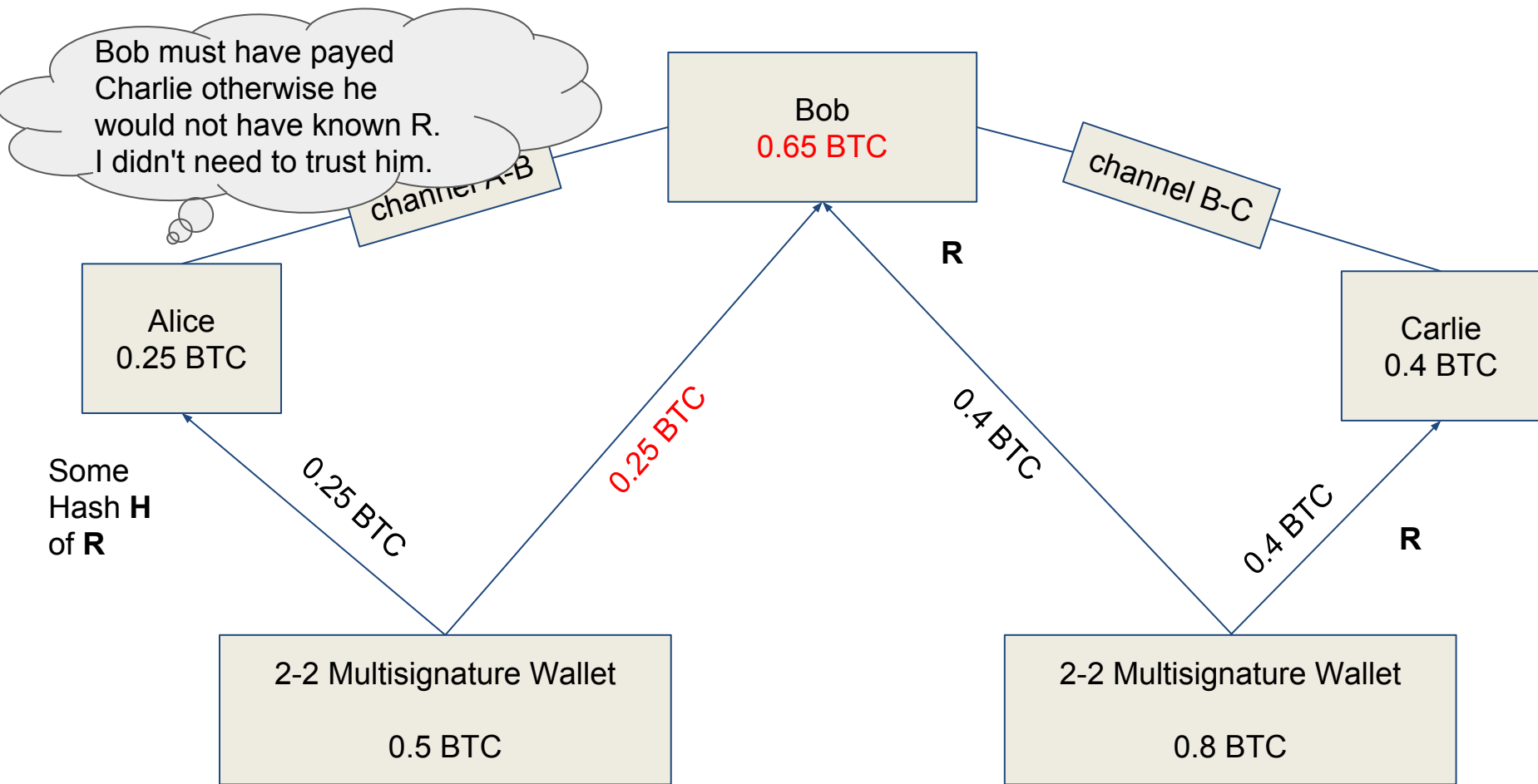
A trustless (Lightning) Network of payment channels



A trustless (Lightning) Network



A trustless (Lightning) Network of payment channels





TO THE
MOOOON!

To the Mooooooon...

- We created a trustless bidirectional payment channel
- Via HTLCs we can create a trustless network of payment channels
- This stuff is all specified in the lightning rfc aka BOLT 1.0
 - Basics Of Lightning Technologies
 - <https://github.com/lightningnetwork/lightning-rfc>
- 3 independent implementations
 - C-lightning (Blockstream)
 - LND (Lightning Labs)
 - Eclair (Acinq)
- Already running on Mainnet

However: We are not done yet.

Problems with the Lightning Network (BOLT 1.0)

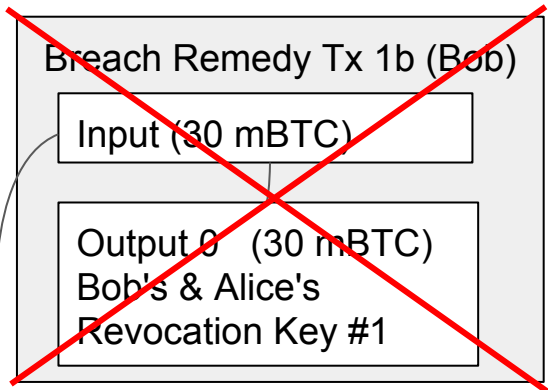
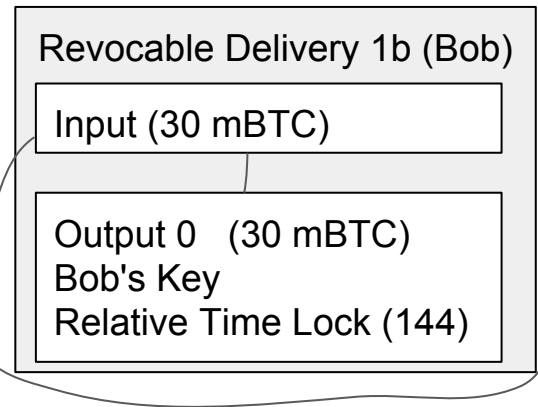
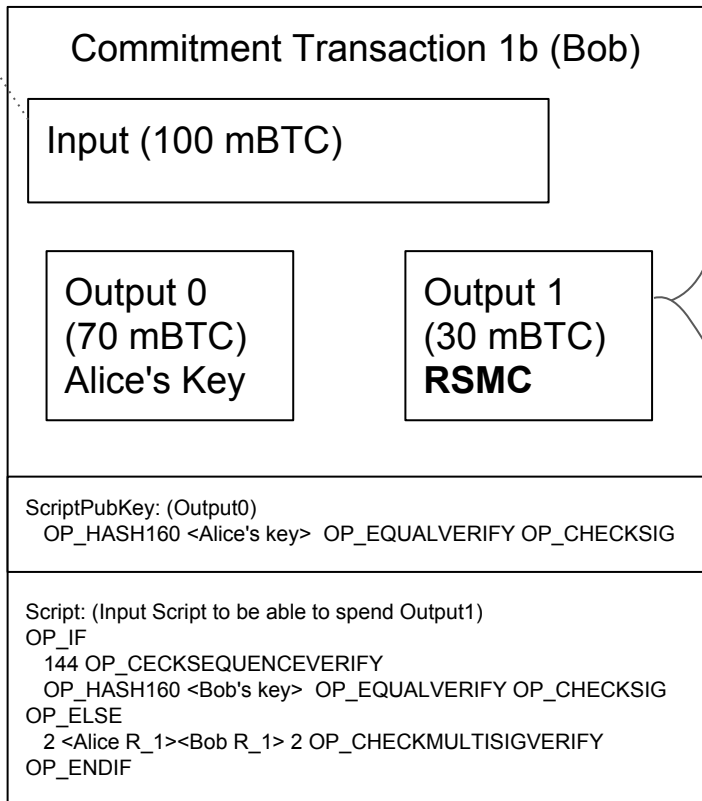
- What happens if a node goes offline?
 - Watch Tower
- Topology of the network partially unknown (in particular channel balances)
 - Routing protocols for creating routing tables and exchange local information
 - Autopilot feature / Network flow theory
- Dried out channels or routes with channels that have too little capacity
 - Atomic multipath routing
 - Splicing
- Funds are being locked up in payment channels
 - Splicing
- Funds are somewhat in a hot wallet for obvious reasons
 - Not aware of people working on this (my idea:)
 - Outputs could only go to a previously defined address from some cold wallet
 - Can only be changed if both sides agree to this (semitrust)

A High level outlook to BOLT 1.1



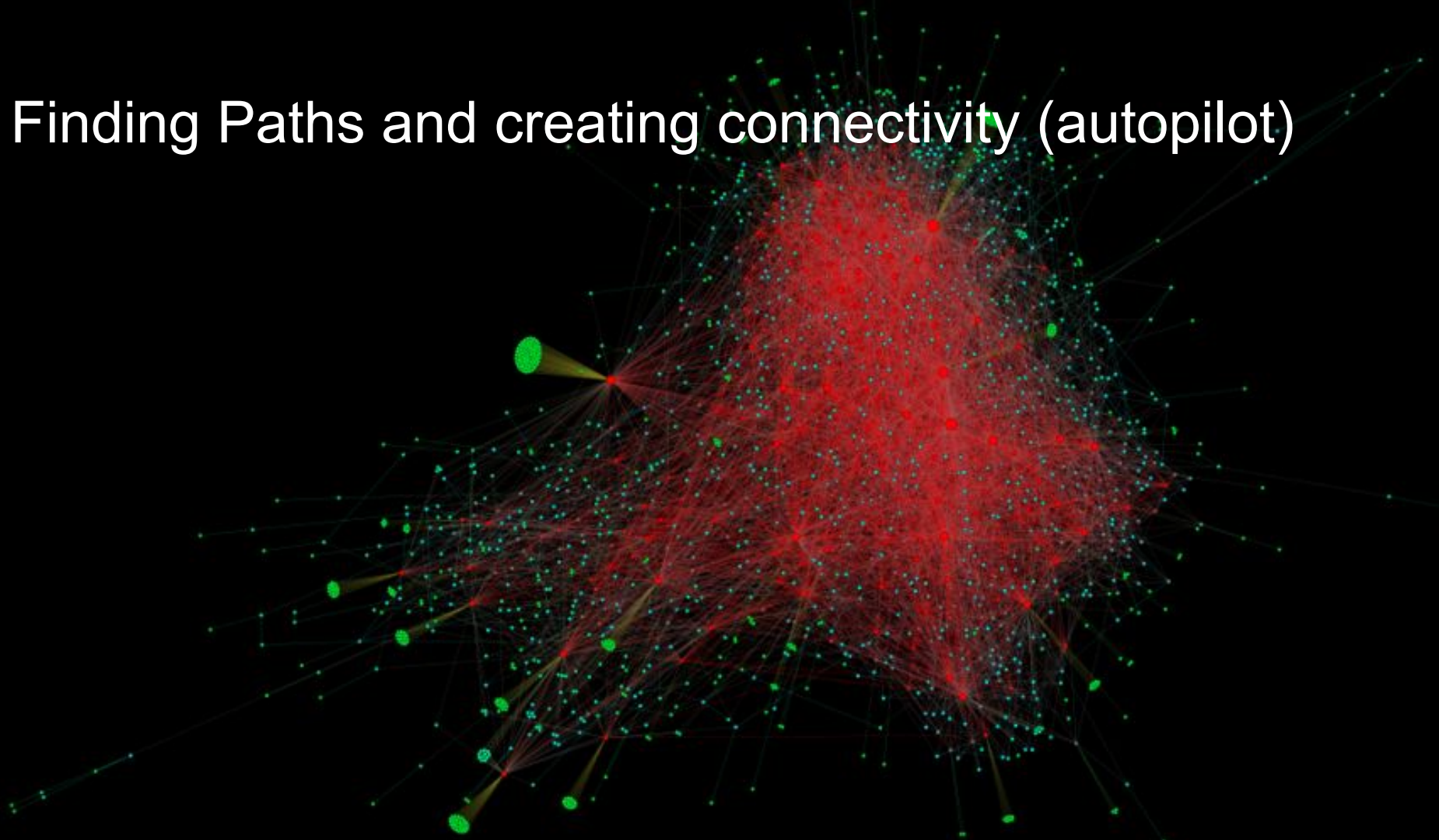
After 144 Blocks Bob can redeem "his" 10 mBTC. What if Alice was not paying attention?

Reference



- Third party (watch tower) keeps track
- Watch tower will be incentivised by getting a small fee as an additional Output in Breach Remedy Transaction
- Prototype implementation by lightning labs complete

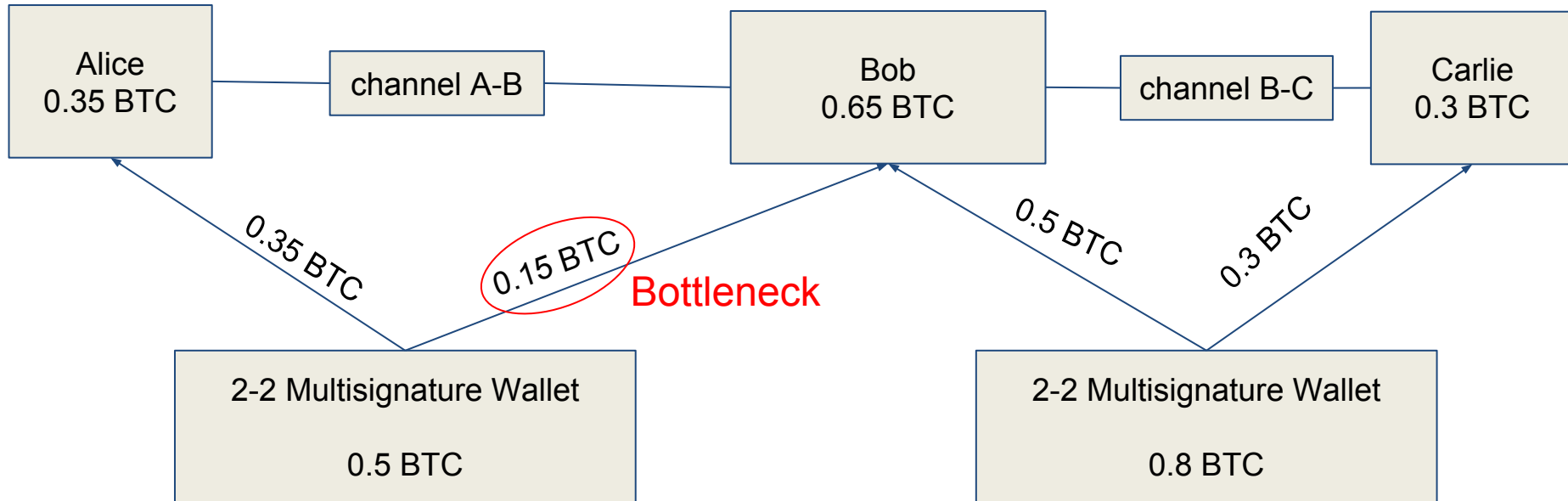
Finding Paths and creating connectivity (autopilot)



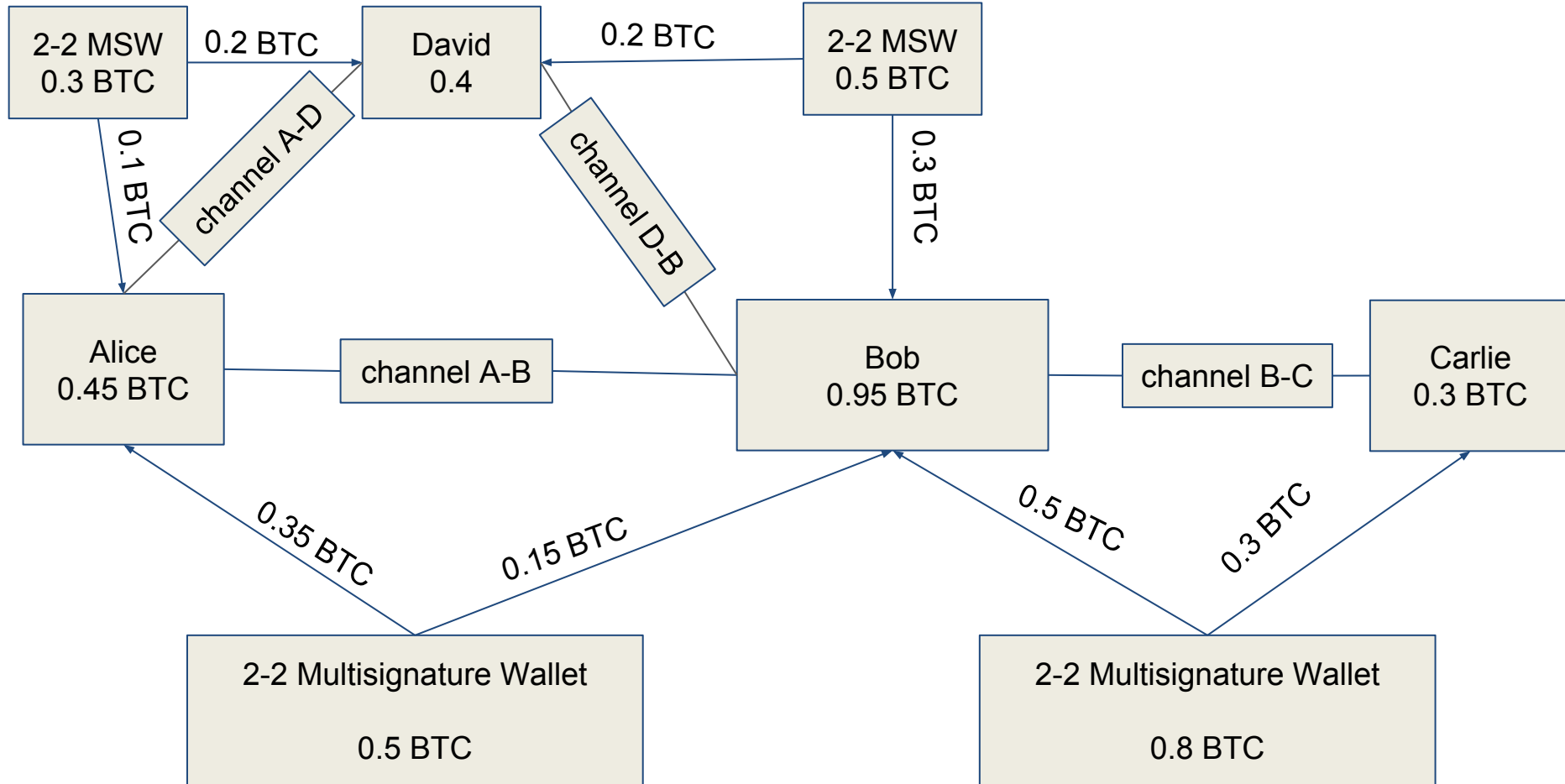
A Lightning Network is only useful if payment requests can successfully be routed

- Directed Graph
 - The capacity graph is undirected
 - However we are interested in balances (which is the possibility to forward payments)
- Topology is only partially known
 - Weights (channel balance) unknown globally due to privacy concerns
 - Weights will dynamically change while routing takes place
- Connectivity concerns can be tackled with the autopilot
 - Avoid super nodes (currently it rather creates super nodes)
 - Avoid central nodes (which when removed decrease the amount of possible paths)
 - Ongoing discussion on [my blogarticle](#).
- Max Flow defines how much money one node can pay to others
 - could be computed if full topology information was public
 - Not helpful without **atomic multipath routing** (next slides)
- Other solutions exist via Splicing or Virtual Payment Channels

Charlie can't her send 0.3 BTC to Alice



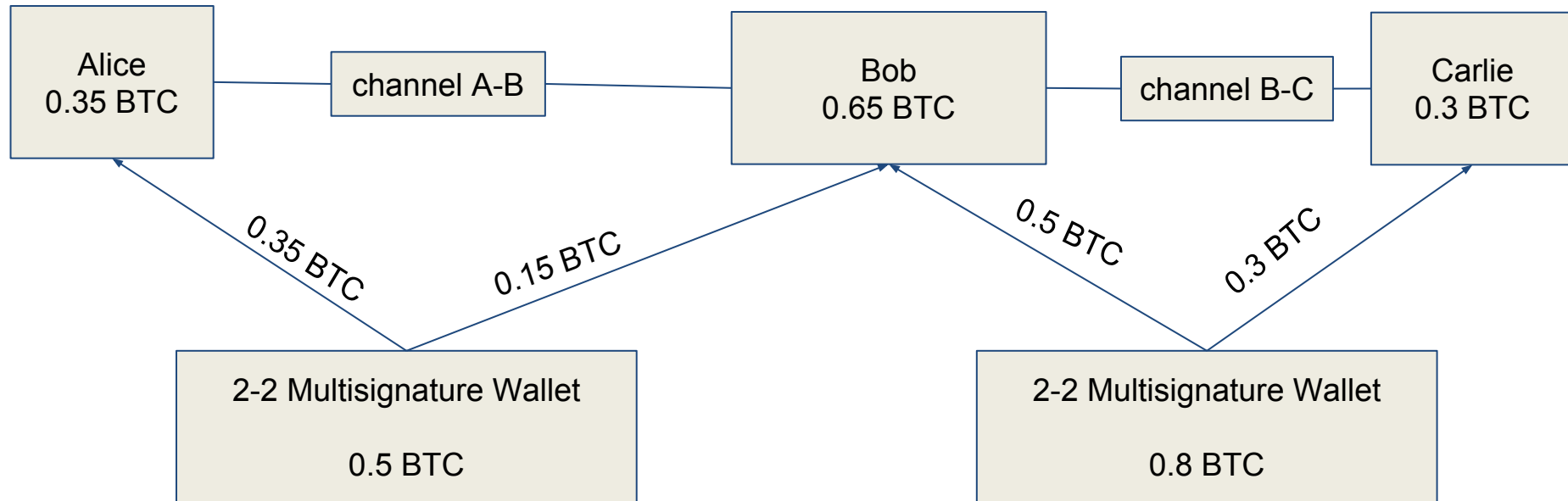
Atomic multi path routing (if several paths exist)



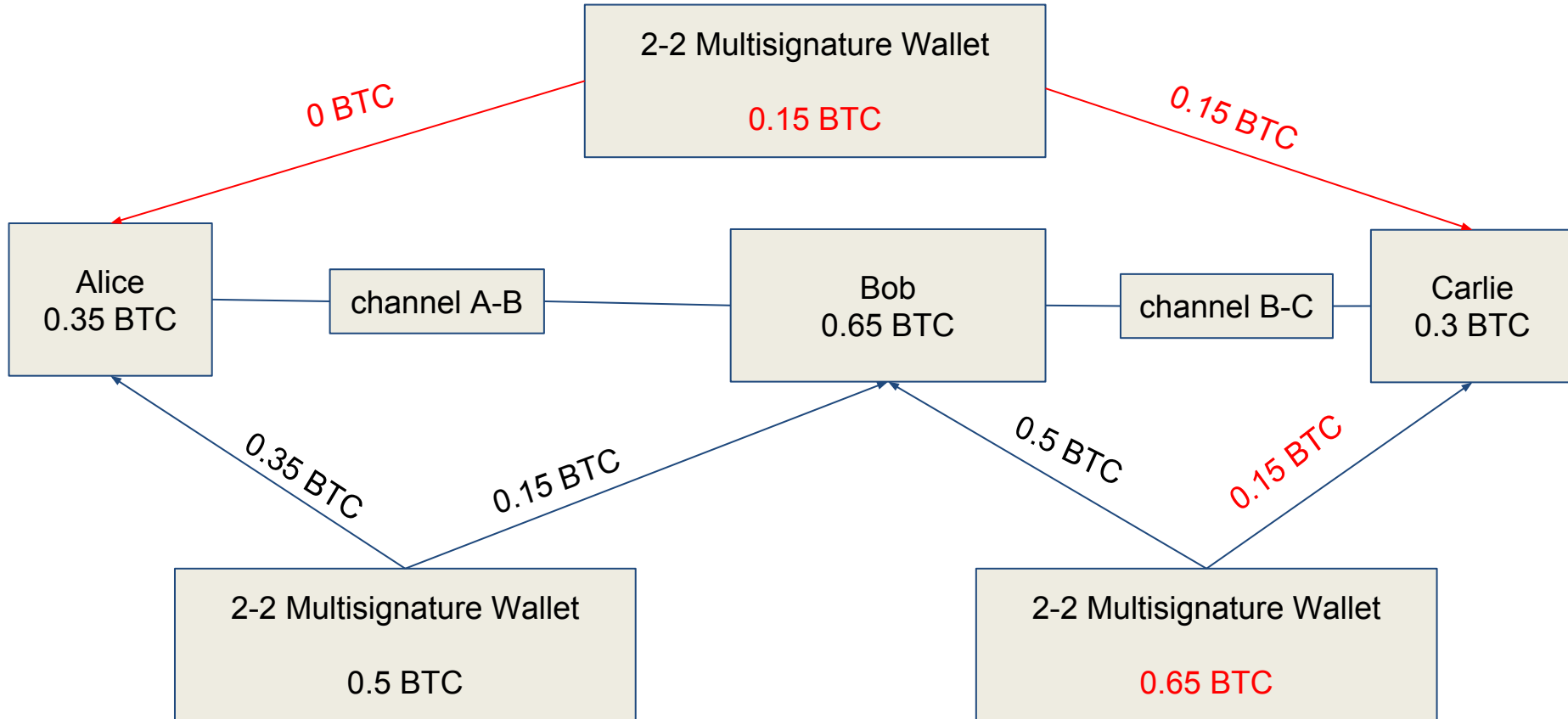
Splicing: How to update the capacity of a channel?

- 2 Goals:
 - Use (at most) 1 blockchain transaction (in the collaborative case)
 - Be non-blocking: Channel should continue to work while splicing tx is not yet confirmed
- Splicing gives a lot of flexibility and helps for the adoption of the Lightning network
 - Sending funds via bitcoin yields 1 blockchain transaction
 - Now you can open a big payment channel with all your funds
 - If the LN doesn't find a route you can splice out funds to create a new payment channel
 - Best case splice out tx and splice in tx happen within one transaction
 - Bitcoin wallets will secretly use the lightning network for transactions.
 - The user doesn't even have to know it!
 - If Atomic Multipath Routing fails Splicing kicks in
 - Speed of transactions will just be faster (and cheaper) as on chain transactions

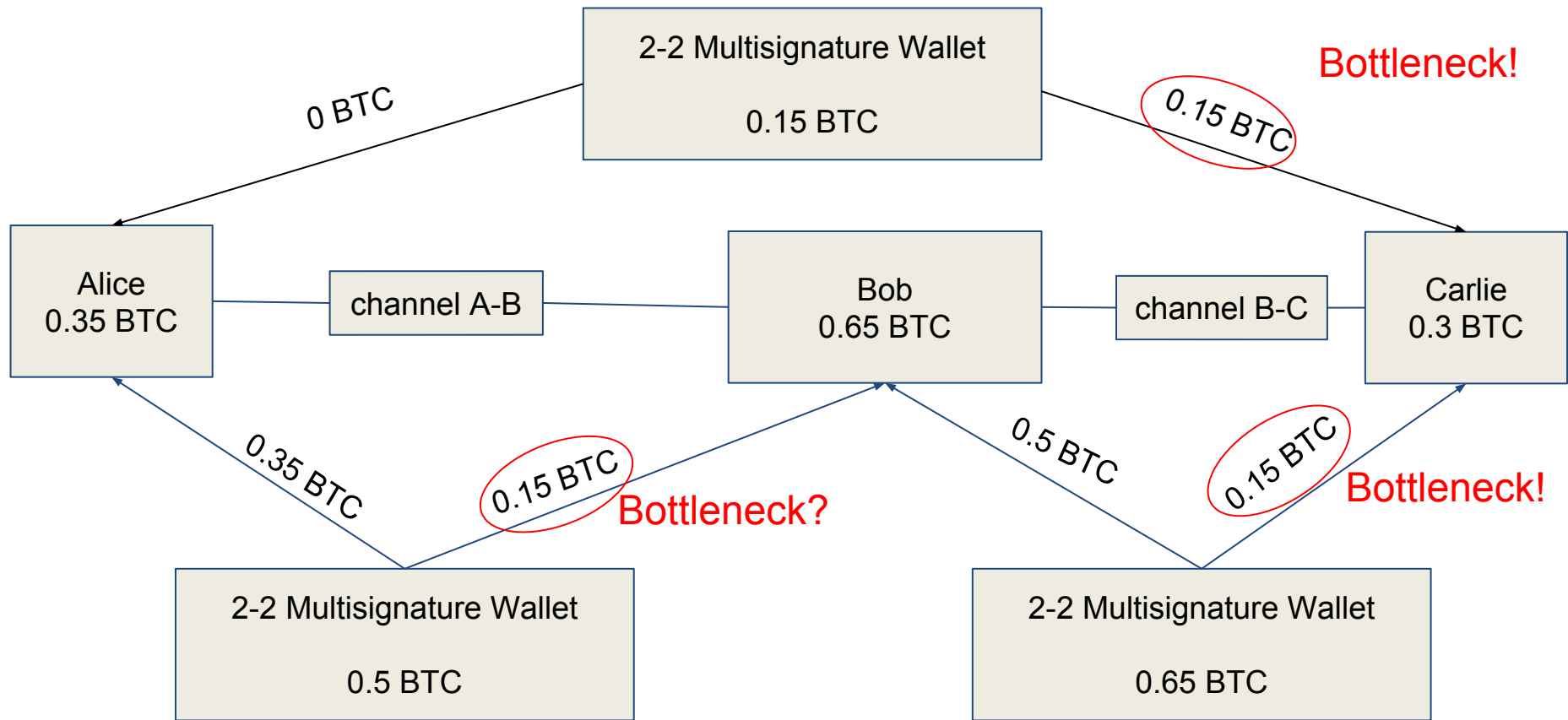
If no other path exists and Charlie has no onchain funds to create another channel she can splice



Move capacity from one channel to another (new) one



We still need Atomic multipath routing in this example



Proposal: A non blocking strategy for splicing out

- Create a new funding tx with 2 outputs which spends the former Funding Tx
 - Output 1 splice amount and recipient address
 - Can be funding for a new payment channel or a splice in tx for an existing channel
 - Output 2 multisig wallet that is currently being used
 - Risk for address reuse?
 - Just create new 2-2 multisig wallet
 - Start updating new commitment txs even if funding tx is not yet mined
 - By protocol contract don't update the commitment tx of old channel
 - or update them accordingly (not necessary)
- No risk for double spending the initial funding tx
 - Can only be done with an old commitment transaction
 - Penalty based revocation system will prevent or reward this
 - Or by consent and mutual agreement (eg revert splicing before confirmation)
 - Recipient of the spliced out tx should still wait for block confirmations to be on the safe side

Proposal for A simple strategy for async splicing in

- Just create a second payment channel on the same multisignature wallet
 - Can't use as long as funding tx is not confirmed
 - Just continue using first channel
- Merge channels When 2nd funding tx is confirmed
 - Update channel state (new pair of commitment transactions)
 - CTXs will spend output of both funding transactions
 - Invalidate both old pairs of commitment transactions by revealing the revocation keys
- Disclaimer: Splicing not tested yet. I might have overseen something
 - Could be nasty to implement handling several active states
 - However I don't see what speaks against this approach

Virtual Payment Channels speed up channel creation

- Channel operates on trust instead of blockchain
 - Opportunity to have less funds on a hot wallet
 - Risk for channel partners
 - Alice can effectively spend Bobs funds and vice versa
- No Risk for the Network as HTLCs and trustless routing is not affected
- Applications
 - Central nodes and hubs
 - Regular node might exceed hardware capacities
 - Distributing node will be difficult which channels connect to which node.
 - Balance and routing difficulties might arise
 - Run several nodes with virtual payment channels among them
 - Mining pools (and other entities serving payouts for multiple customer)
 - Fund inbound virtual payment channels with (funds on the side of the customer)
 - Fund outbound regular payment channels with central lightning nodes
 - Give customers access to spend their virtual funds

Eltoo - invalidating channels with less overhead

- Extend the Protocol to allow for a new type of channels
- No need to store all old revocation keys
- Ascribing blame not necessary
 - Symmetric information about the channel
- Multiparty channels become much easier
 - Especially if Schnorr-Signatures become part of the Softfork which enables eltoo
- Implementation will carry less overhead
 - Watch towers
 - HTLCs
 - Splicing
 - AMP Routing (haven't thought this through yet)
- Again: Read that Paper:
 - <https://blockstream.com/eltoo.pdf>
 - Summary at <https://www.rene-pickhardt.de/thoughts-about-eltoo-another-protocol-for-payment-channel-management-in-the-lightning-network/>

Atomic Swaps - decentralized exchanges

- Lightning can be built on top of any blockchain
- A node that has payment channels on top of different blockchains could offer to do an atomic swap
 - Acting as a decentralized exchange
- The lightning network becomes an overly network of blockchain application
 - As the internet became an overlay network for link layer protocols

Lightning Network Improvement Proposal - LIP

- Bitcoin has a process to suggest protocol improvements
 - Bitcoin Improvement Proposals (aka BIPs)
 - <https://github.com/bitcoin/bips>
- Core Lightning developers will meet in November to start crafting BOLT 1.1
 - Currently many ideas are out in the wild (as described here)
 - Having a clear process for LIPs would make the process for BOLT 1.1 more efficient
- Looking for support for this idea from the community
 - <https://github.com/lightningnetwork/lips> (does not exist yet)
 - Over the next weeks I will start collecting the ideas that are out at:
<https://github.com/renepickhardt/lips> (which certainly shall move to the above URL)

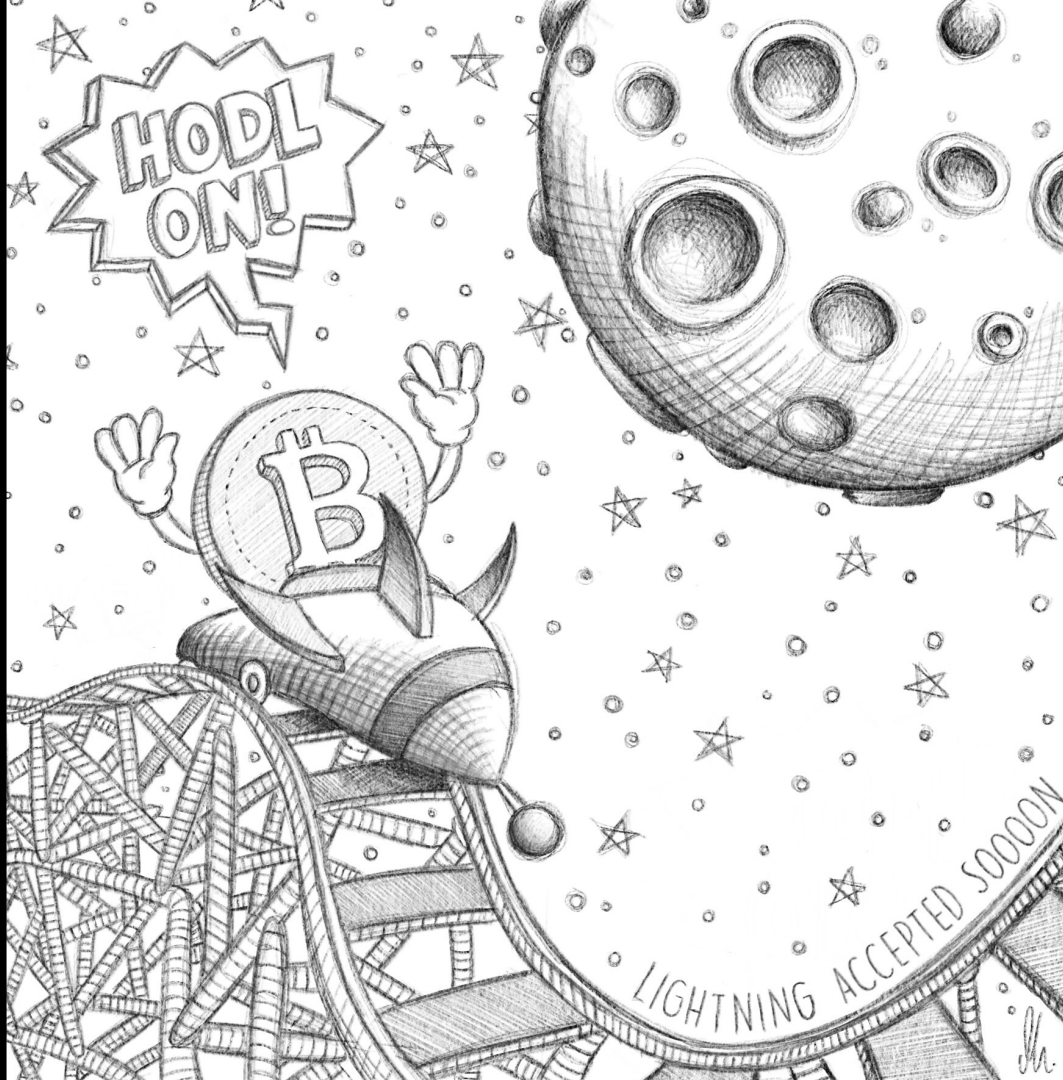
Join the financial revolution!

Become an open source developer / activist

- Many Open tasks
 - Lots of new features to be added for BOLT 1.1
 - Open issues on github
- Only few developers
 - Apparently companies rather focus on their own products instead of contributing to the technology that drives this industry
- I have just recently decided to really commit to lightning dev
 - I can tell you it is doable
- Very open and welcoming community
- Communicate with the community
 - Mailinglist
 - Slack
 - Blogs
 - Twitter / Reddit (?)

More resources about Lightning

- Lightning Network [white paper](#)
- Bitcoin [Whitepaper](#)
- Lightning - RFC ([BOLT - Basics of Lightning Technologies](#))
- Developer [Mailinglist](#)
- The [Bitcoin Wiki](#) has some high quality resources (e.g.):
 - [Transaction](#)
 - [Script](#)
 - [Lightning Network](#)
- [Wikipedia Article](#) - There is also the [German Version](#) (I am the main author)
- Implementations on Github
 - [c-lightning](#)
 - [Ind](#)
 - [eclair](#)



HODL ON!

LIGHTNING ACCEPTED SOON

(another) Shameless plug:
I can be hired :)

Thanks for your attention

- Slides are openly licensed. Fork them on Wikimedia Commons
- For Questions and Comments: <https://www.rene-pickhardt.de>
- Thanks to Fulmo Lightning
 - for sponsoring some of my activities
 - <https://www.fulmo.org>
 - #LightningHackday (next data September 1st in Berlin)
- Open a channel with my node (:



036f464b54416ea583dcfae3872d28516dbe85414ed838513b1c34fb3a4aee4e7a@144.76.235.20:9735

Copyright

This slide deck is licensed under [CC-BY-SA](#) License which means you have to give Attribution to the author in the Following way:

Attribution: Rene Pickhardt (<https://www.rene-pickhardt.de> & <https://commons.wikimedia.org/wiki/user:Renepick>)

Pictures used:

- <https://www.pexels.com/photo/action-adventure-challenge-climb-449609/> via <https://www.pexels.com/@martin-138963> (CC0)
- <https://en.bitcoin.it/wiki/File:Transaction.png> CC-BY-SA by Theymos via bitcoin wiki
- <https://en.bitcoin.it/wiki/File:TxBinaryMap.png> CC-0 by Etotheipi via Bitcoin Wiki
- <https://pixabay.com/en/girls-whispering-best-friends-young-914823/> CC-0 via Pixabay by <https://pixabay.com/en/users/Olichel-529835/>
- <https://pixabay.com/en/megaphone-loud-speaker-speaker-1480342/> CC-0 via Pixabay by <https://pixabay.com/en/users/terimakasih0-624267/>
- Hodl on Lightning Accepted soon CC-BY-SA Cryptospiration (Marietheres Viehler & Rene Pickhardt)
- To the Moon CC-BY-SA Cryptospiration (Marietheres Viehler & Rene Pickhardt)
- Photographs of the lightning Bolt CC-BY-SA (Rene Pickhardt)
- https://commons.wikimedia.org/wiki/File:Bitcoin%27s_Lightning_Network_Visualization.png CC BY-SA 4.0 By StopAndDecrypt via Wikimedia Commons (cropped by me)
- High Level Outlook to Bolt 1.1 by Martina Pickhardt