Lightning Network Part I

scritto da Alberto De Luigi | 26 Maggio 2016
Go to part II >

Blockchain data size is expanding too much and too fast

To reach the same volume of Visa transactions, the Bitcoin network would require 50 terabyte per year. In this regime there is no possibility for bitcoin to substitute fiat money.

Each single user (full node) should download too many data, unless it trusts some "authorities" which download the entire blockchain, intermediating and guaranteeing the information about which outputs are already spent.

The loss of indipendency of the user and the centralitazion of the system contradict the ideological premises the Bitcoin system is based on, and obliterate the main benefits compared to other money systems.

The lightning network is an «extension» of the Payment Channel

BitcoinJ software implemented the «Payment channel»: a secure system of «off chain» transactions (not broadcasted to the blockchain) which can reduce the size of data loaded on the blockchain and the costs (miners' commissions).

The «Payment channel» is a channel between two users who needs to make recurrent payment between each other (like the bill payer and the phone company). Only a final transaction will be loaded on the blockchain at the end of the relationship (after n. payments). It's a valid tool for the stipulation of contracts.

The system is already working but does not solve the problem

of scalability. A valid extension of this system is the Lightning Network.

How Payment Channel Works:

that is, how Bob and Alice can trade goods and services in exchange of bitcoin in a secure way, without to broadcast each transaction on the blockchain

The multisignature «funding» transaction

Suppose Alice has 10 bitcoin and she and Bob are willing to create a channel payment.

Alice create a multisignature transaction we call F (funding transaction), which transfer 10 bitcoin from Alice wallet as input into an «address» as output that is conrolled by both Alice and Bob.

From a technical point of view, the signature script moving the bitcoin from F presents two public key hash, one corresponding to the pair of keys public/private of Bob, the other Alice’s pair of public/private

If F would be broadcasted to the blockchain, neither Bob nor Alice could independently move the bitcoin transferred in F as a new input for another transaction, since both private keys are necessary to move that output. For this reason, Alice does not sign the transaction to send her own bitcoin in F before Bob guarantees he won’t «blackmail» her keeping the money locked in F.

Since by hypothesis both Alice and Bob are willing to create the channel payment, Bob gives her this guarantee. In fact a new transaction is created, called R, where R means Redeem or Refund transaction (the transaction allowing Alice to redeem her funds). R uses 10 bitcoin present in F as input e and sends them back to Alice’s wallet.

It’s possible to transfer the 10 bitcoin in F even if F is not already broadcasted to the blockchain. In fact, to create the transaction R connected to F, it’s enough to know the hash of F, broadcasted to Bob without let him know other details about F. Bob signs the signature script which is constituted by the public key hash of R communicated by Alice, signing with the private key associated to an unspent output: normally, this output is in the blockchain, in the case of F it is located in the server where F is registered

The guarantee: a «refund» transaction

Thus Bob signs the signature script of the transaction R with the private key he owns unlocking the output of F to be used as input for R. Bob broadcasts R to Alice.

Alice owns the other private key for F, then she can sign at any time the funds transfer and can broadcast it on the blockchain, sending back the 10 bitcoin to her wallet.

Only after Bob signs R and broadcast it to Alice, she will sign F. In fact now Alice can move back to her wallet the output of F into R at any time. In this moment both Alice and Bob can broadcast F on the blockchain, but it is not necessary.

A “fund” (the multisignature F) has been created and it works as «base» for the channel payment transactions. In fact, all the payments in the channel use the bitcoin present in F.

The time constraint nLocktime

Alice can broadcast the transaction R at any time, but she will actually «hold» the output in R («hold» means she can use it as input in another transaction) only after a certain amount of blocks have been created.

In fact the transaction R is created specifying a parameter that doesn’t allow Alice to use the output before a certain date.

In the next graphic examples, if the channel is created Monday, Alice won't hold the bitcoin in R before Friday.

The period of time between Monday and Friday is measured by means of the parameter **nLocktime**, namely the number of new blocks created, remembering each new block is created after 10 minutes average.

The Payment Channel transactions occur within this time span (Monday-Friday)

Note: the time constraint is only for the Refund transaction

(only for pedants, in case skip directly to the pictures)

In the next pictures is shown a payment channel where each transaction (TX 1, 2 and 3) presents a nLocktime and the relative output can be used only after a certain date (respectively Thursday, Wednesday and Tuesday).

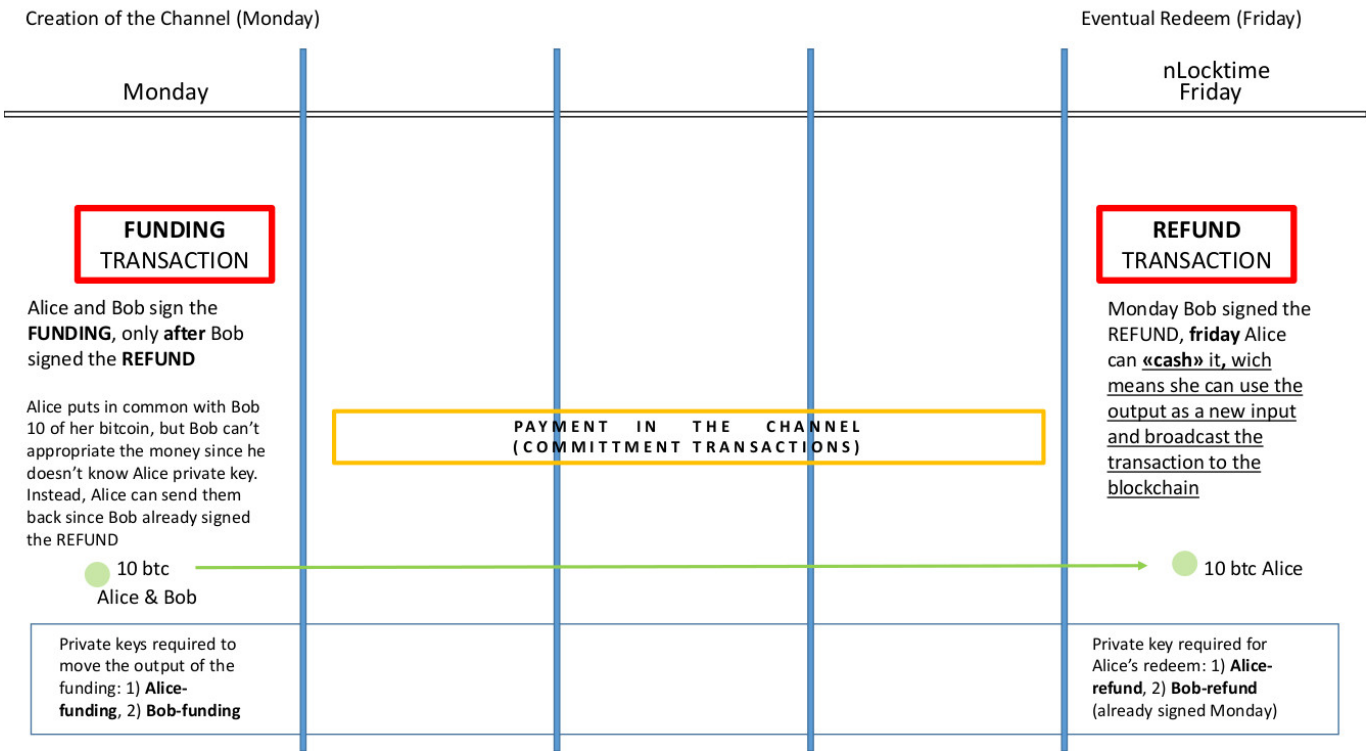
Actually, in the payment channel implemented in BitcoinJ, only the Refund tx has a specified time contraint set by the nLocktime, since the transactions TX are saved on the server, which deletes the old transactions, replacing them with the new one. Both Bob and Alice coudl ask at any time the server to broadcast the last transaction and take the money, closing the channel, but they can't broadcast the transaction in autonomy (otherwise they could broadcast i.e. TX1 even if TX2 already occurred).

The system is not perfect because it requires intermediation: if the server is destroyed, Bob broadcast F, Alice can redeem her money broadcasting R, whose copy she keeps in local on her PC; however no TX transaction is recoverable, even if Bob has

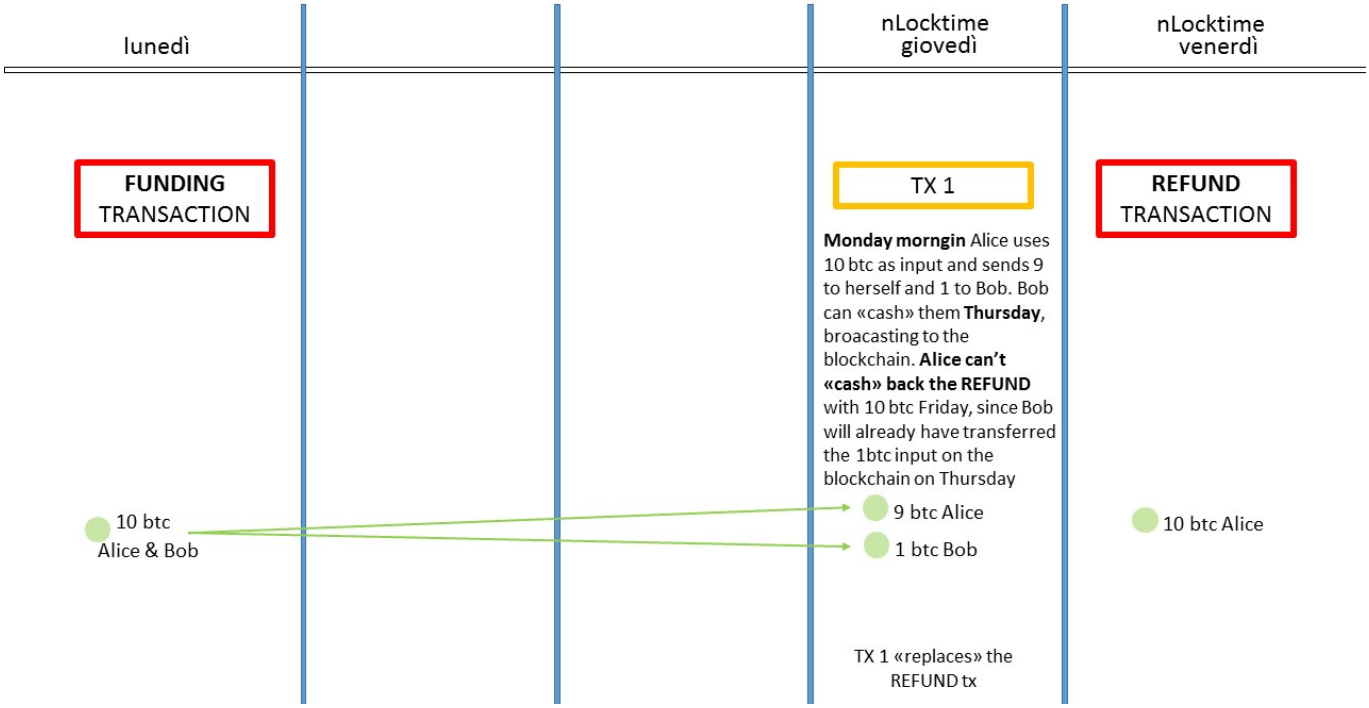
already delivered the smartphone to Alice. She could then «steal» the smartphone to Bob if something happens to the intermediary BitcoinJ.

It’s more useful to think of each transaction occurring in the channel with a time constraint as showed in the graphs. This doesn’t require intermediation and keep us close to the idea of Lightning Network.

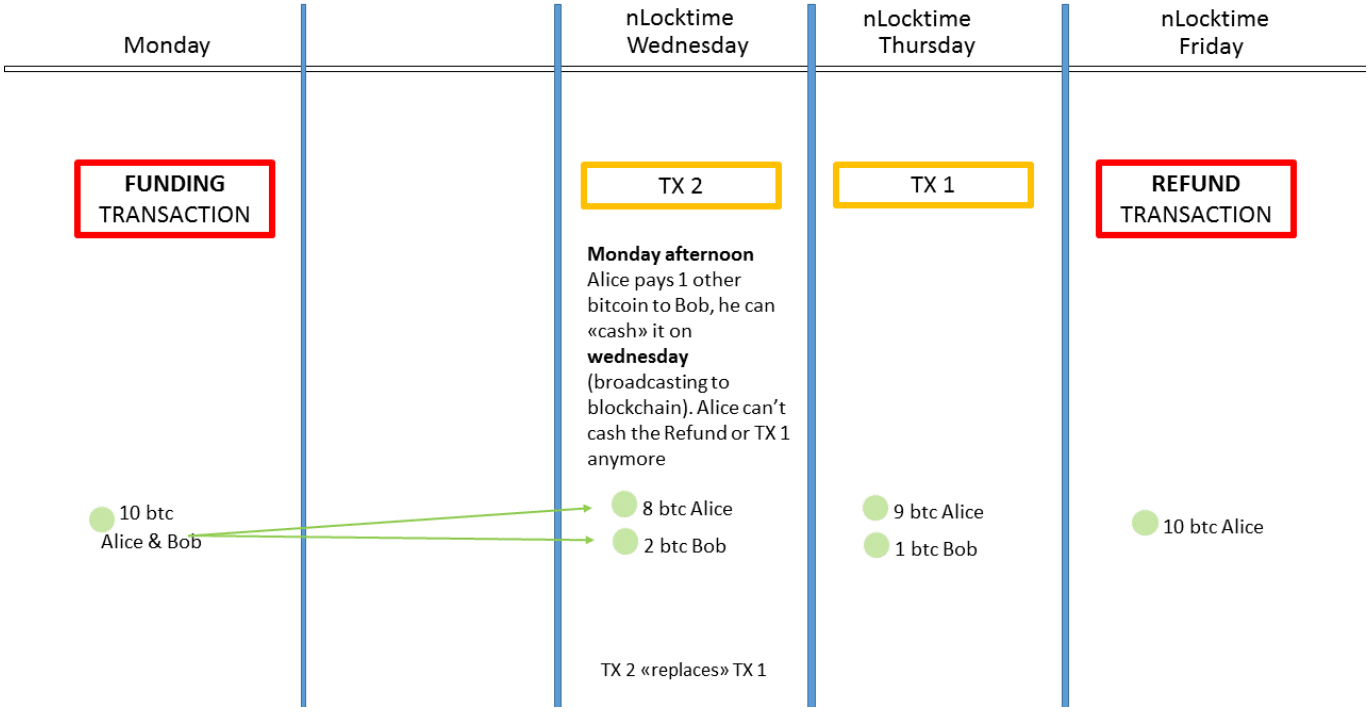
Note also that the term «commitment transaction» is taken from Lightning Network terminology.

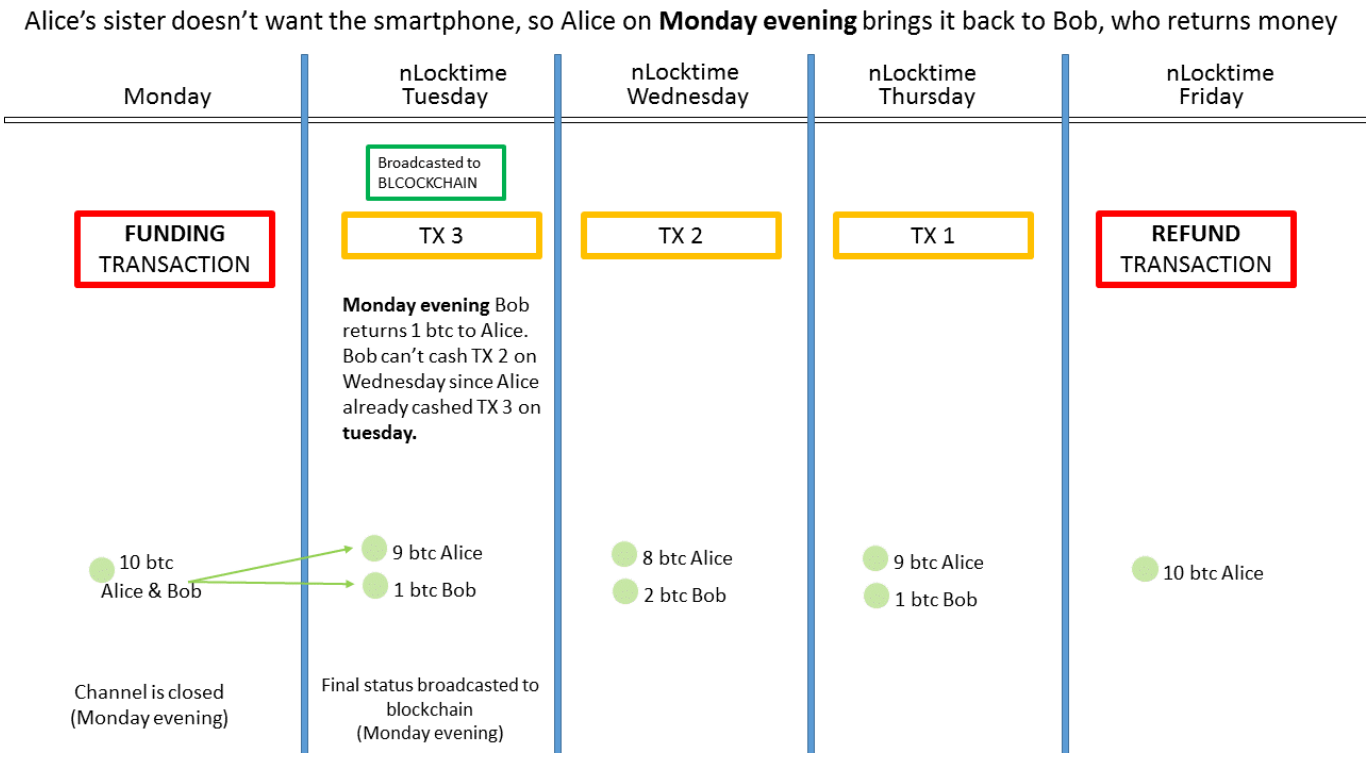


Monday morning Alice goes to Bob's shop and buys a smartphone paying 1 btc



Monday afternoon Alice comes back to Bob, she likes the smartphone and buys a new one for her sister





Payment channel vs. Lightning Network

1. In the Payment channel the channel can be opened only between two users with continuous relationship, thus for each couple of users it is necessary to broadcast to the blockchain at least 1 transaction
2. The channel can't last beyond the prearranged date (number of blocks created) fixed by the Refund transaction. It is thus necessary to create a new channel each time, increasing blockchain size
3. It is necessary the intermediation of BitcoinJ (or other intermediaries), since TX 1, 2 and 3 have no time constraint, contrary to what we saw in the graphs

vs

1. In the Lightning network the channel is open indefinetly.
2. It is not necessary the intermediation of a BitcoinJ server (or others)
3. The channel is «replaced» by a network of users (potentially, every bitcoin user)

Go to Part II >