Specification about GPC

Zhichun Lu

Part 1: The detailed message exchange during making payments and channel establishment.

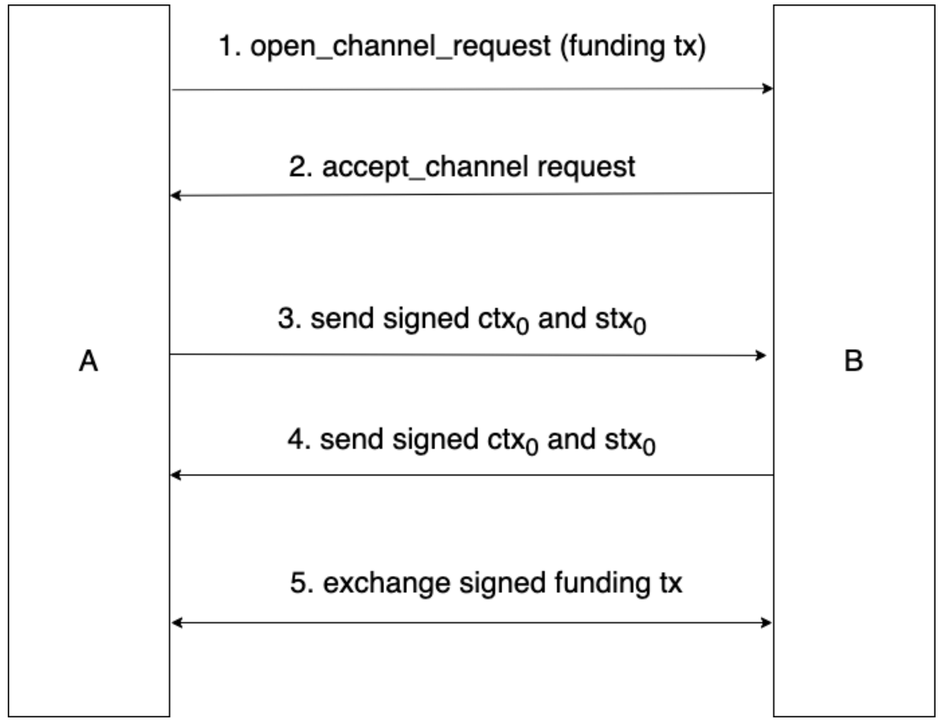
Part 2: The Pseudocode of lock script.

Part 3: The modul.

Zhichun Lu

Payment channel establishment

1. A creates a unsigned funding tx and sends it to B.
2. B checks whether the funding tx is correct as they had negotiated before. If it is correct, just send the “accept” message.
3. A sends signed ctx_0 and stx_0.
4. B check ctx_0 and stx_0, and sends signed ctx_0 and stx_0.
5. Both A and B sign the funding tx and send to each other.

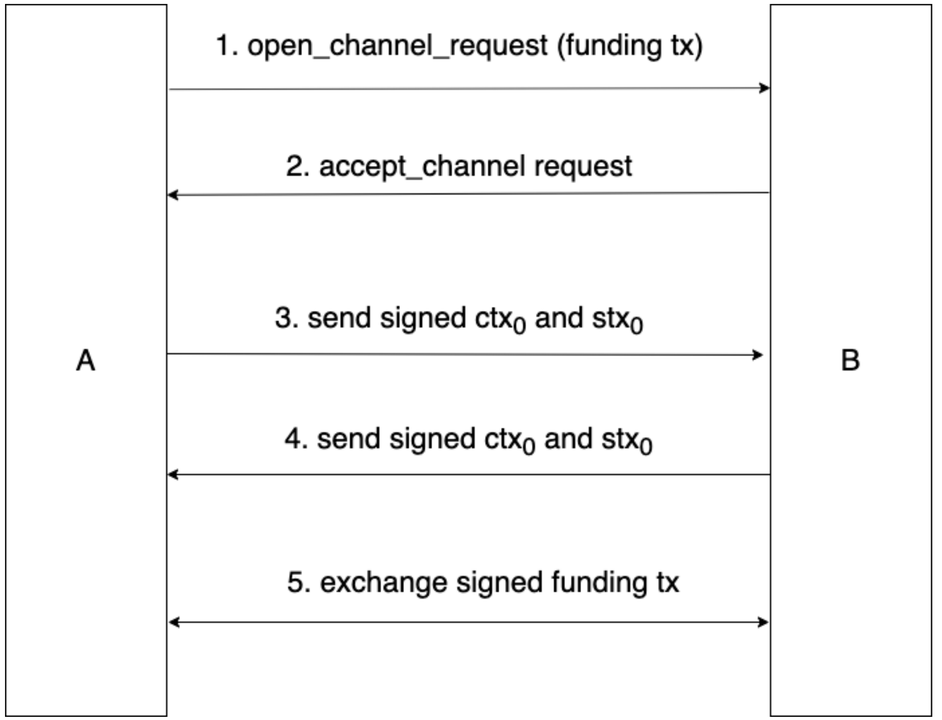


Zhichun Lu

Payment channel establishment

- As we can see, the funding tx is unsigned before the step 5. So if the node does not follow the first four steps, it will only cause the channel to fail, but will not cause any economical loss.

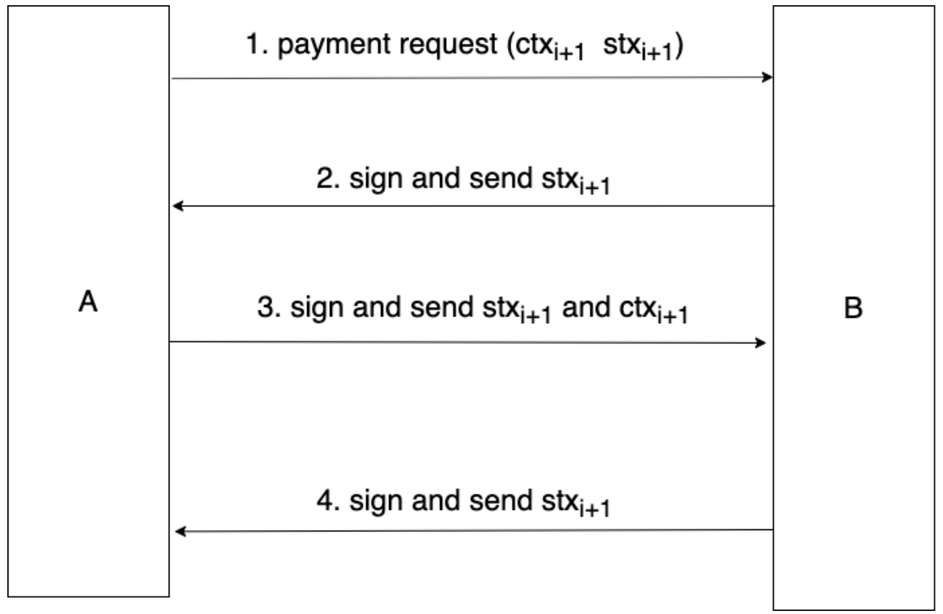
- In step 5, I summarize by exchanging signatures, because I think it is harmless for either party to send first. For example, If A sends his signature first, B attack his signature to it and broadcasts. A can get his money back by using refund tx. And vice versa.



Zhichun Lu

Making payment

Let's assume that the nounce of the latest commitment that both parties now have is i. And they're going to do a payment, and nounce is i+1.

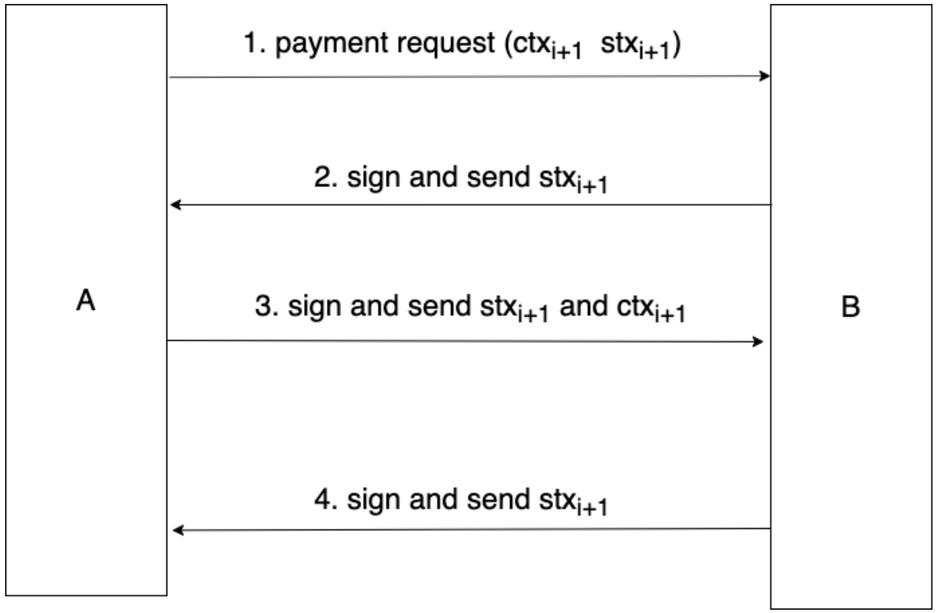


1. A creates unsigned closing and settlement tx and send them to B.
2. B checks whether the it is correct as they had negotiated before. He just send the signed stx_{i+1}.
3. A sends both the signed stx_{i+1} and ctx_{i+1}.
4. B checks the signature is correct and sends the signed stx_{i+1}

Zhichun Lu

Making payment

- The process is very similar to the funding & refunding tx, since stx_{i+1} takes the output in ctx_{i+1} as input. So even they sign stx_{i+1}, it can not be accepted by blockchain until ctx_{i+1} be signed and sent to chain.



Zhichun Lu

Closing channel

About closing channel, Jan seperate it into 3 situation. i). The good, ii). The bad and iii). The ugly. But I put the bad and the ugly together here, because they both only interact with blockchain.

- The good: In this case, we assume both party make an agreement about the final status about the payment channel. So they just exchange the signature about closing tx, the type flag in witness should be 0.

- The bad/ugly: In this case, the node only needs to interact with the blockchain, which can be summarise into 3 steps.

  1. Look for the latest transaction of the channel in the blockchain, which may be a funding transaction or a closing transaction, and create a tx, the input is the output in the latest tx's output. The output and witness are the latest ctx in local. Note that the signature about ctx is no-input-sig. So we can combine it with any input, and we should set the type flag in witness to 1.

  2. Check whether the tx is accepted by blockchain, if yes, goto step3. Otherwise, repeat step1.

  3. Send the stx corresponding the latest ctx to chain, and get the assets back. The type flag in this tx is 0.

Zhichun Lu

The lock script

In summary, the script should tackle with 4 situation.

- State == open and state == 0: It is “The good” case, in this case we just need to verify the signature is valid.
- State == open and state == 1: In this case we just need to verify the no-input-signature is valid, and the output’s nounce should be greater than input’s nounce.
- State == closing and state == 0: The same as situation 2.
- State ==closing and state == 1: It is the settlement case, we need to verify the signature is valid and the Timeout has passed.

Zhichun Lu

The lock script

Situation 1:

If state == open and state ==0: #bilateral closing

flag = (verify_sig(input.lock.arg.pkb_A, sig A)) and (Verify_sig(input.lock.arg.pbk_B, sig B))

If flag:
  return 0
else:
  return -1

Situation 2:

Elif state == open and state ==1:#unilateral closing

flag = (verify_no_input_sig(input.lock.arg.pbk_A, sig A)) (verify_no_input_sig(input.lock.arg.pbk_B, sig B)) and (input.lock.arg.nounce < output.lock.arg.nounce)

If flag:
  return 0
Else:
  return -1

Zhichun Lu

The lock script

Situation 3:

Elif state == closing and state == 0: # update tx

flag = (verify_no_input_sig(input.lock.arg.pbk_A, sig A)) and (verify_no_input_sig(input.lock.arg.pbk_B, sig B)) and (input.lock.arg.nounce < output.lock.arg.nounce)

If flag:
  return 0
Else:
  return -1

Situation 4:

Elif state == closing and state == 1# settle tx

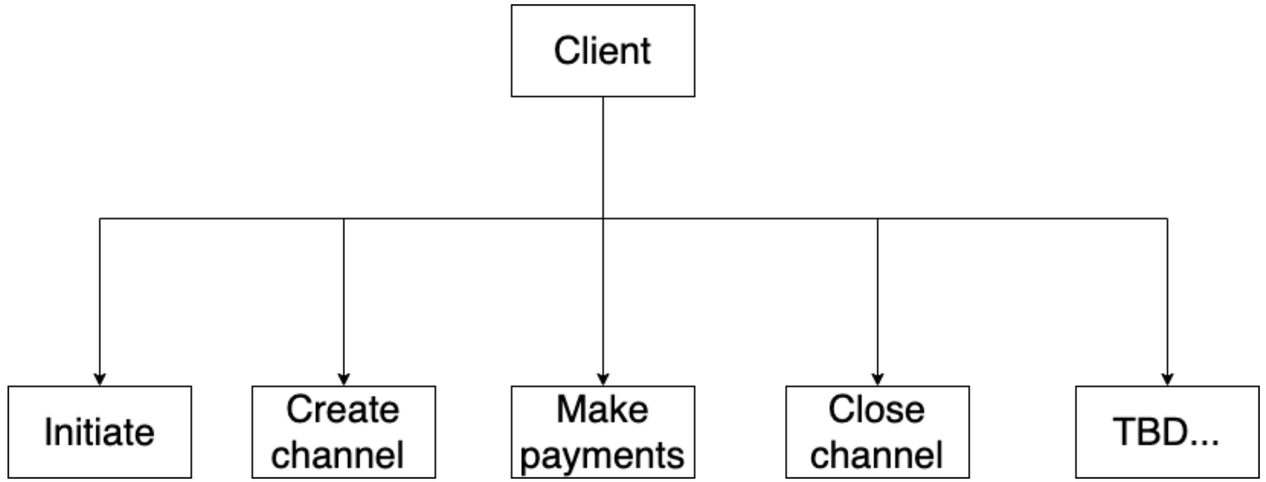
flag = (input.since == input.lock.arg.timeout) and (verify_sig(input.lock.arg.pkb_A, sig A)) and (Verify_sig(input.lock.arg.pbk_B, sig B))

If flag:
  return 0
Else:
  return -1

Specification

In summary, I think the client has at least four functions.

- Initiate (Relate the client to one specific CKB account).
- Create channel
- Make payment
- Closing closure
- Multi-hop payment (TBD)



And all other functions except initate will be classified as active and passive. For example, you can actively establish channels with others, or passively accept the request of others to establish channels.

Zhichun Lu

Specification-initiate

Description:For a command line client, we need to initialize it with the CKB public key and private key. Because the establishment and closure of channels need to interact with the blockchain. For example, in order to create a channel, the nodes need to know which live cells it have, and I think our client should provide that information to them.

Require:

1. The public key.
2. The private key.

Ensure:

submit transactions to the blockchain and get data from it.

Submodule:

1. A module to get the data in CKB. (ruby).
2. A module to send transaction to the blockchain. Note that it only about how to send the tx, which means that this module requires the user to provide the input, output, witness and other information of the tx. (ruby)

Zhichun Lu

Specification-channel establishment

More specifically, the establishment of channels should be divided into two cases. One is the initiator and the other is the receiver. At the same time, the client should establish a session pool. Currently I think the format of the session should be <cid, src_pub_key, trg_pub_key, status>

The initiator case

Description:For the initiator, the client can initiate a request to establish a payment channel to another node, and behave according to our interaction process described above. (I just want to list what I think is necessary here, but of course, we need to add more information when implementing it, just like the TCP specification.)

Require:

1. The input cells of funding tx.
2. The public key of another funder.
3. The IP address of another funder.

Ensure:

Finish the payment channel establishment (of course the establishment may fail).

Zhichun Lu

Specification-channel establishment

The submodule:
1. A module be used for network communicating. (python, tcp)
2.A module creates the transaction (it includes funding tx, closing tx, settlement tx) according to the args. (ruby)
3.A module to sign and check the signature (normal and no-input). (ruby or python)
4.A module to read stream data as other structure (json?) (python)
5.A module to verify that the input cells provided by another party is valid. In btc, it may ask other party, I am not sure how it work in CKB, if you have any idea, please tell me. (TBD)

The receiver case

Description: This module should be optional be on or off. When it is open, the client will report the request for establishing each channel, and the user can selectively respond. When closed, the user ignores all channel establishing requests.

Require:

1. The input cells.
2. The public key of another funder.
3. The IP address of another funder.

Ensure: The same as initiator case.

Submodule:The same as initiator case.

Zhichun Lu

Specification-closure

Closure falls into two categories, The good and the bad/ugly.

The good

Description: Both parties negotiate the finality state of the channel and submit it to CKB. This is similar to to process about establishing channel.

The bad/ugly

Description: The user submits the closing transaction and starts a contest to submit the latest "nounce". The person who submitted the latest nounce eventually won the game and used settlement tx to close the channel.

Require: The commitment with latest nounce.

Ensure: Closing the channel.

Submodule: The same as it section “Channel establishment”

Zhichun Lu