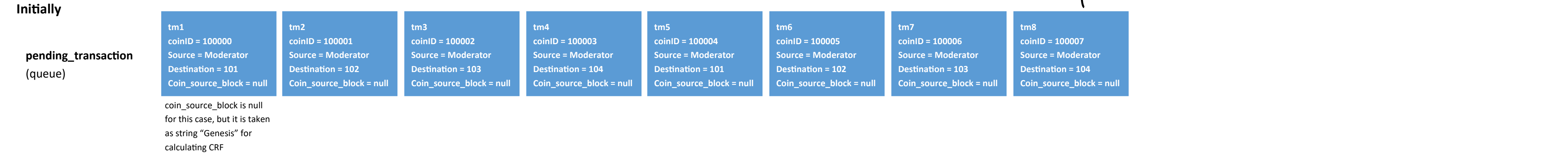


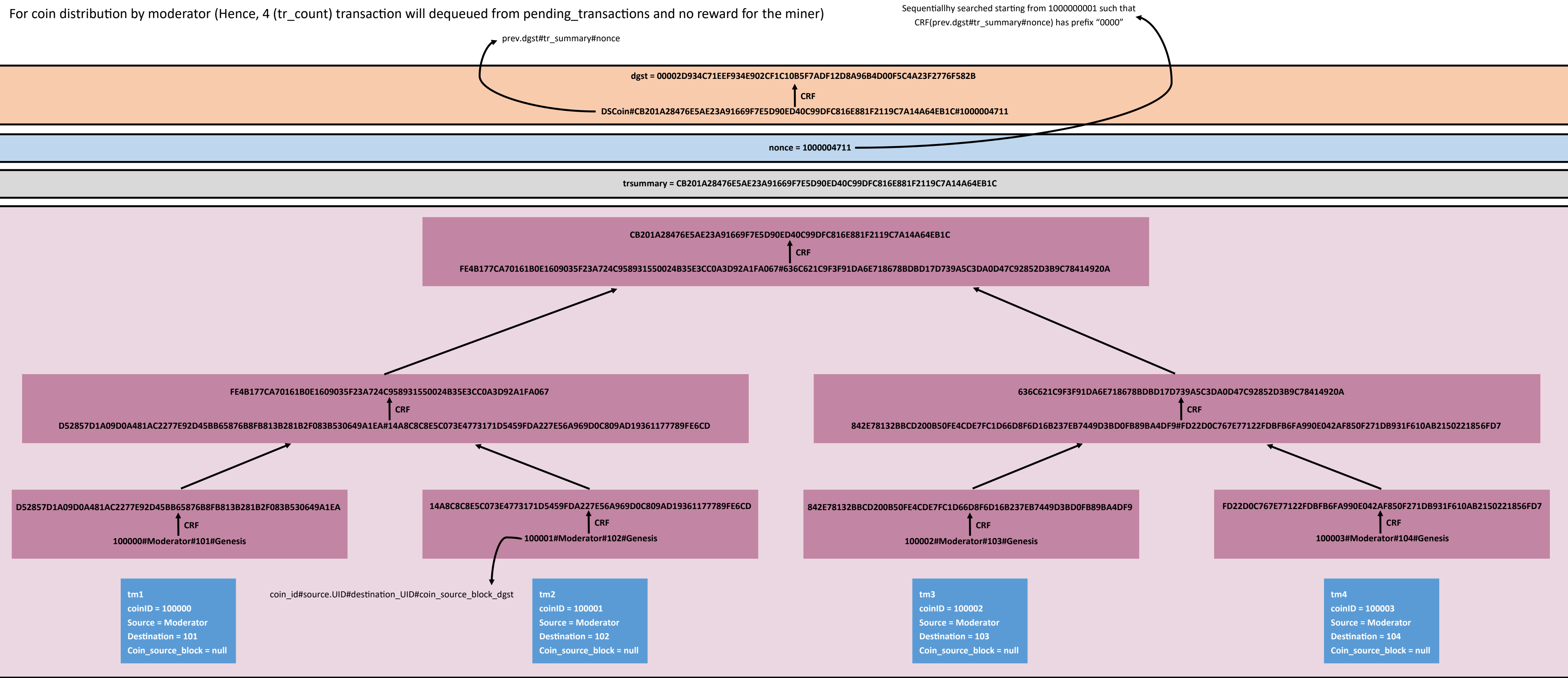
DSCoin_Honest Example:

- Let us assume that, there are 4 members in our DSCoin example with UID’s 101. 102, 103, and 104 respectively.
- Let the tr_count is set to be 4 in this example.
- Initially, the moderator distributes 2 coins to each member. And hence, a total of 2 Transaction Blocks are created in order to distribute the coins to the user. These 2 transaction blocks are created by the moderator and hence, no rewards are allotted for mining these transaction blocks namely, tb1 and tb2.
- Say, tm1, tm2, tm8 are the transactions for distributing the initial coins to the members.
- Please go through the transaction block, pending_transactions queue, and block coin carefully to understand the procedure being followed in DSCoin.

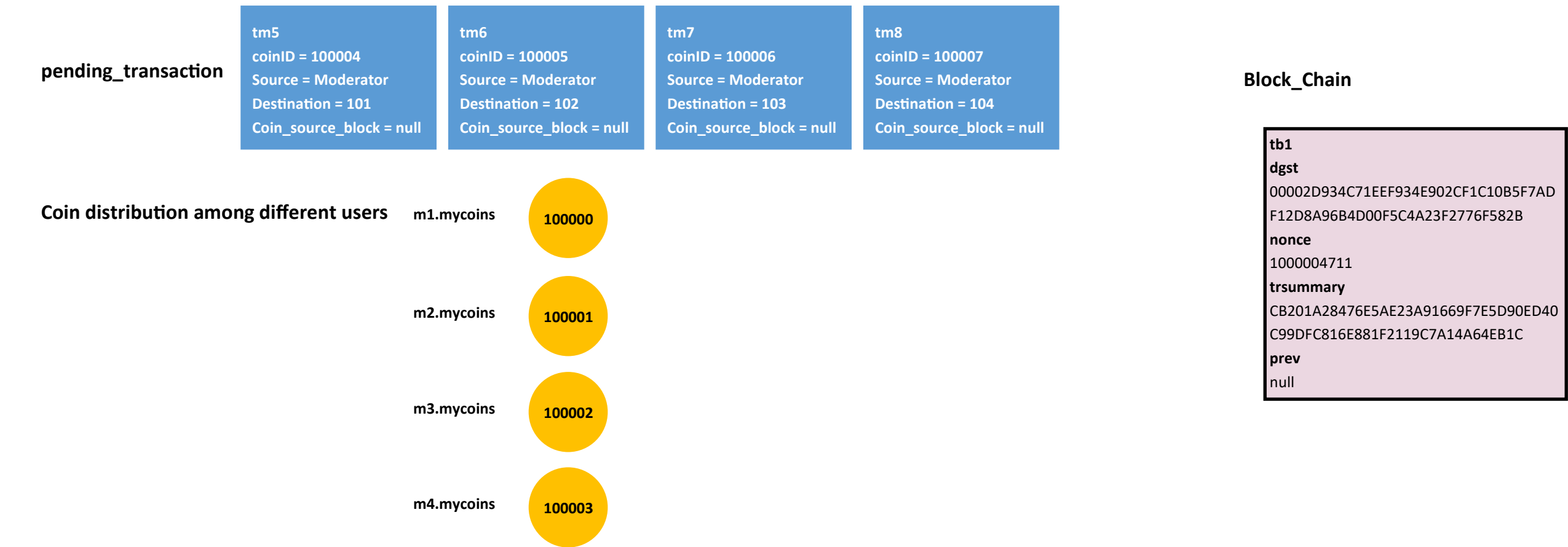


TransactionBlock 1 (tb1):

For coin distribution by moderator (Hence, 4 (tr_count) transaction will dequeued from pending_transactions and no reward for the miner)



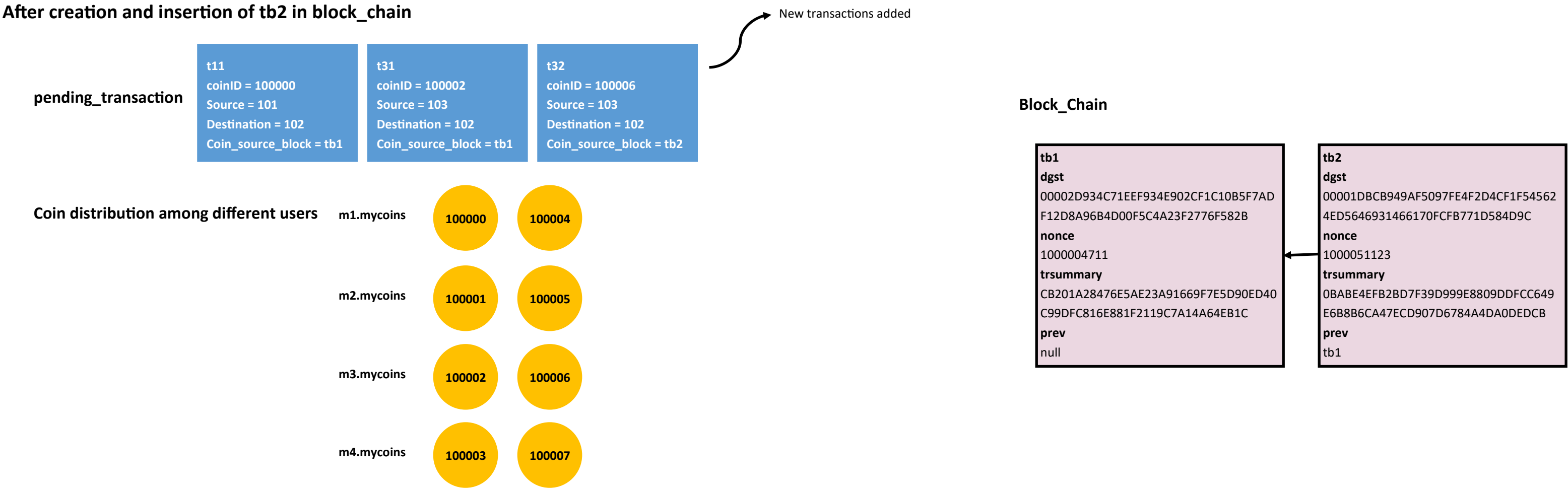
After creation and insertion of tb1 in block_chain



TransactionBlock 2 (tb2):
For coin distribution by moderator (Hence, 4 (tr_count) transaction will dequeued from pending_transactions and no reward for the miner)

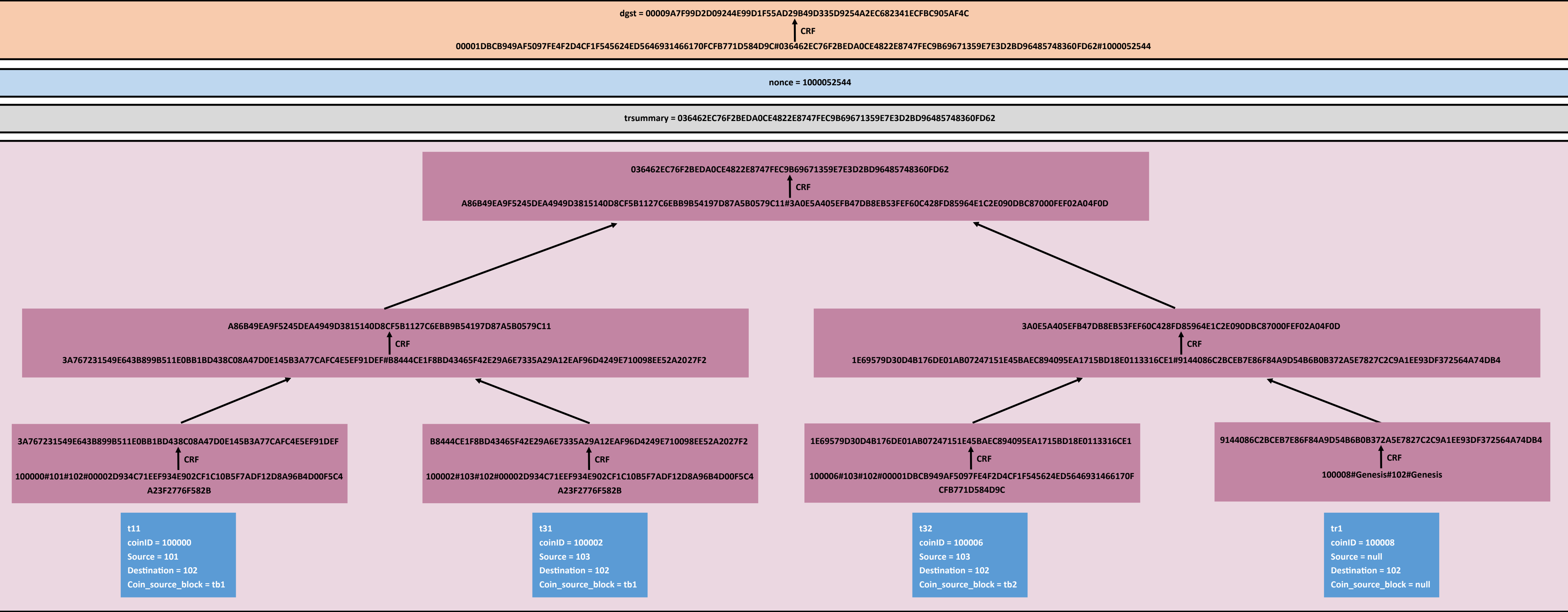


After creation and insertion of tb2 in block_chain



TransactionBlock 3 (tb3): (Mined by m2 (UID: 102))

Mined by a member, hence 3 (tr_count – 1) number of transactions are dequeued from the pending_tranascions and a transaction for reward for miner is added to the block



After creation and insertion of tb3 in block_chain

Observe that member 101 has already initiated a transaction t11 where it is sending the coin “100000” to 102, hence it can’t spend the same coin again. This kind of transaction is termed as double_spending.
(Search the block chain for finding and ignoring doble spending transactions.)

pending_transaction

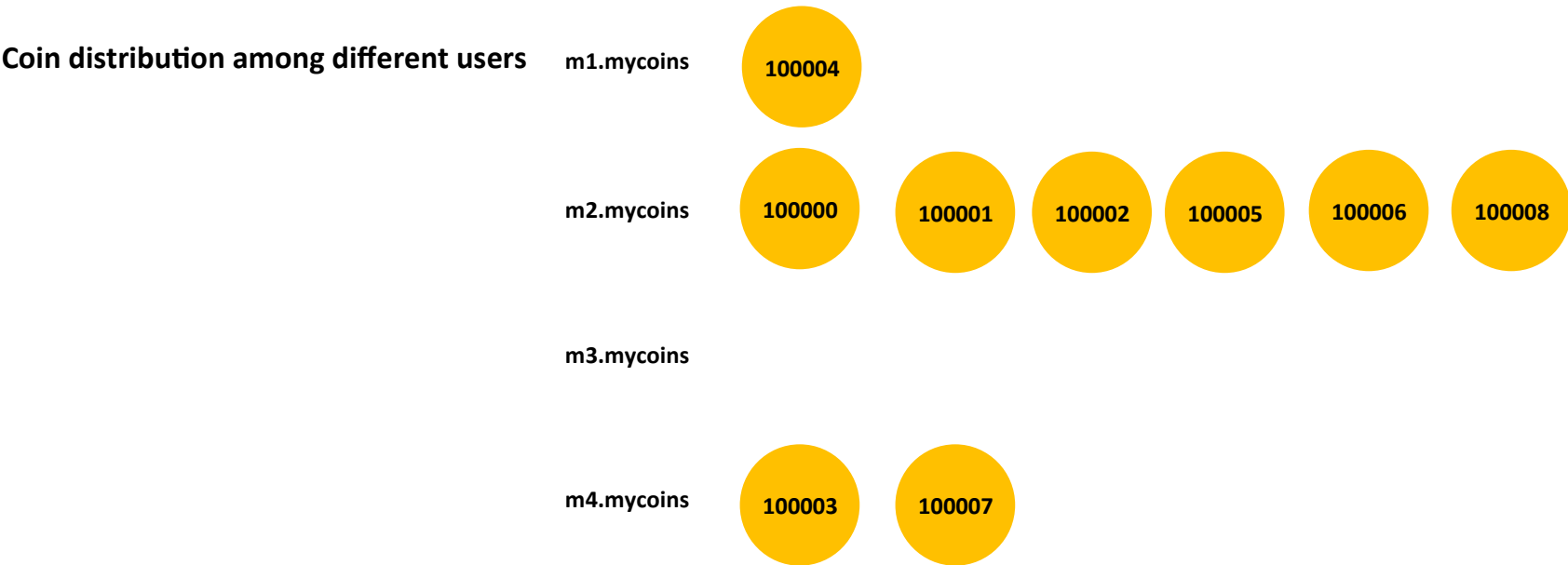
t21
coinID = 100000
Source = 102
Destination = 101
Coin_source_block = tb3

t12
coinID = 100000
Source = 101
Destination = 103
Coin_source_block = tb1

t23
coinID = 100001
Source = 102
Destination = 101
Coin_source_block = tb1

t41
coinID = 100003
Source = 104
Destination = 101
Coin_source_block = tb1

New transactions added



Block_Chain

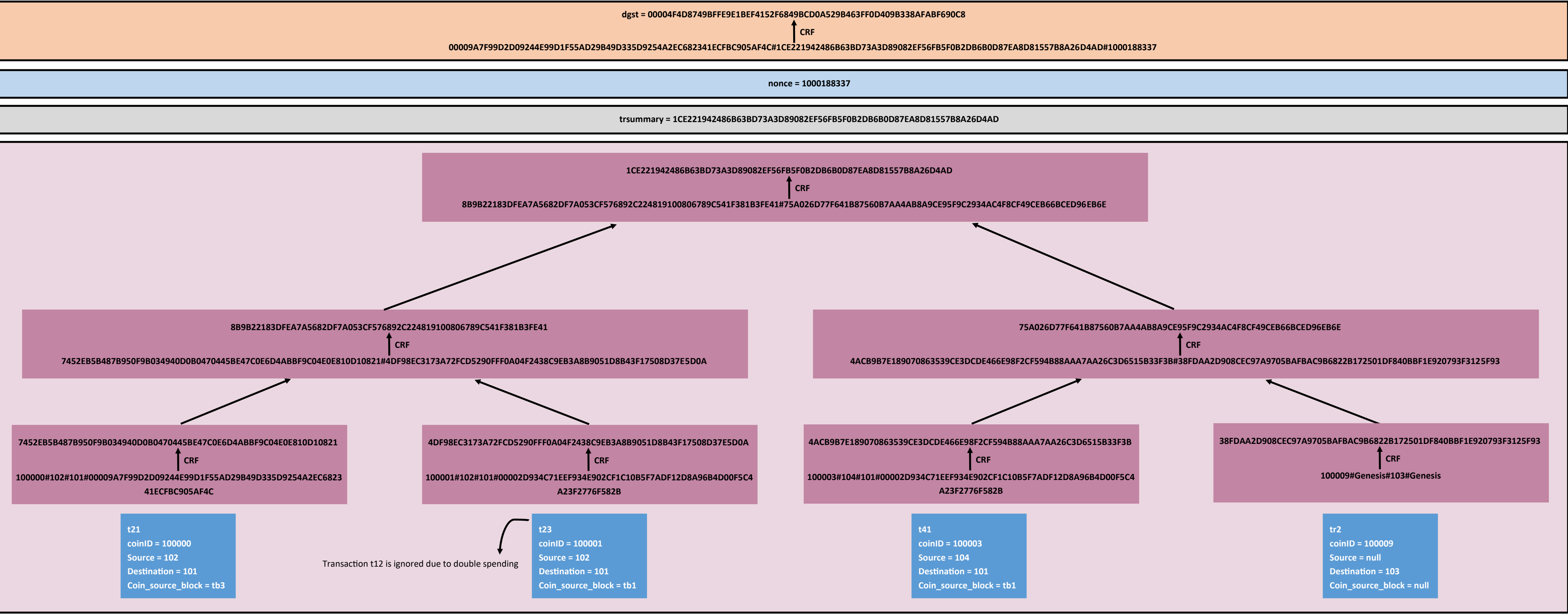
tb1
dgst
00002D934C71EEF934E902CF1C10B5F7AD
F12D8A96B4D00F5C4A23F2776F582B
nonce
1000004711
trsummary
CB201A28476E5AE23A91669F7E5D90ED40
C99DFC816E881F2119C7A14A64EB1C
prev
null

tb2
dgst
00001DBC8949AF5097FE4F2D4CF1F54562
4ED5646931466170FCFB771D584D9C
nonce
1000051123
trsummary
0BABE4EFB2BD7F39D999E8809DDFC649
E6B8B6CA47ECD907D6784A4DA0DEDCB
prev
tb1

tb3
dgst
00009A7F99D2D09244E99D1F55AD29B49
D335D9254A2EC682341ECFBC905AF4C
nonce
1000052544
trsummary
036462EC76F2BEDA0CE4822E8747FEC9B6
9671359E7E3D2BD96485748360FD62
prev
tb2

TransactionBlock 4 (tb4): (Mined by m3 (UID: 103))

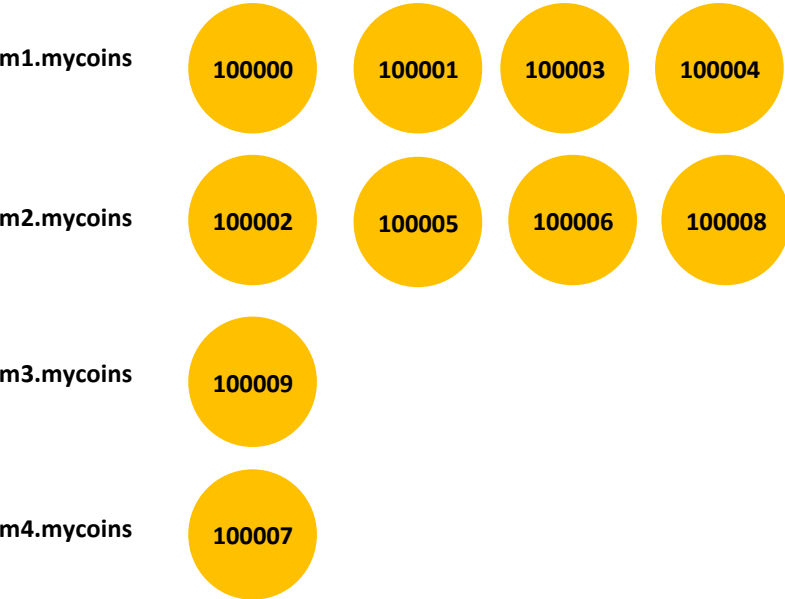
Mined by a member, hence 3 (tr_count – 1) number of transactions are dequeued from the pending_tranascions and a transaction for reward for miner is added to the block



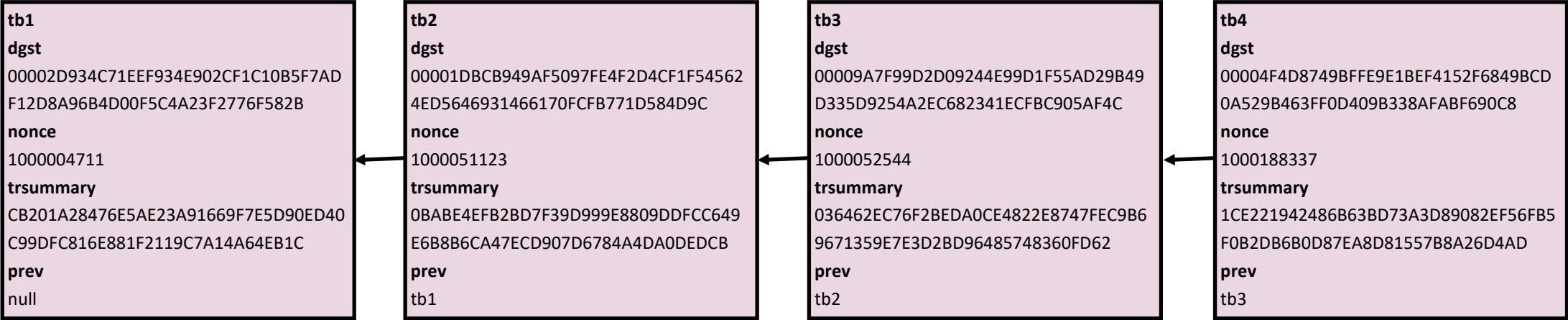
After creation and insertion of tb4 in block_chain

pending_transaction (empty queue)

Coin distribution among different users



Block_Chain



finalizeCoinsend Example

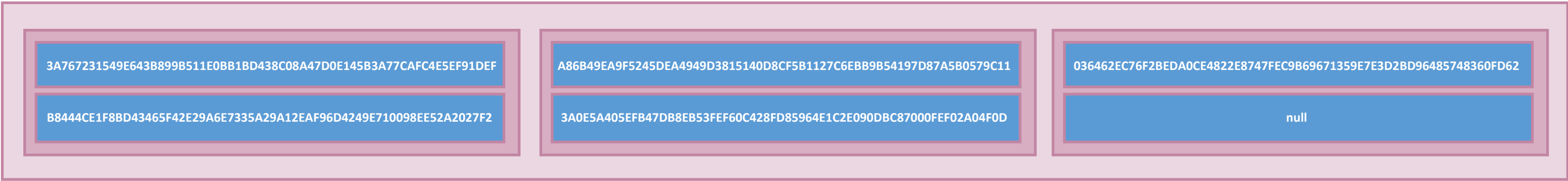
finalizeCoinSend(t31, DSobj)

- We first need to find the TransactionBlock containing t31, we first sequentially search tb4.tr_array (as tb4 is the last_block of the block chain), since t31 is not present in tb4, using prev pointer we goto tb3, and sequentially search for t31, and since it is present in tb3, we will now take tb3 and do the following.
- We have to compute the sibling coupled path to the root for t31 in tb3 Merkle tree to prove membership of transaction (i.e. proof that transaction t31 is a member of TransactionBlock tb3)

MerkleTree of tb3 (with sibling cupled path to root highlighted for t31):

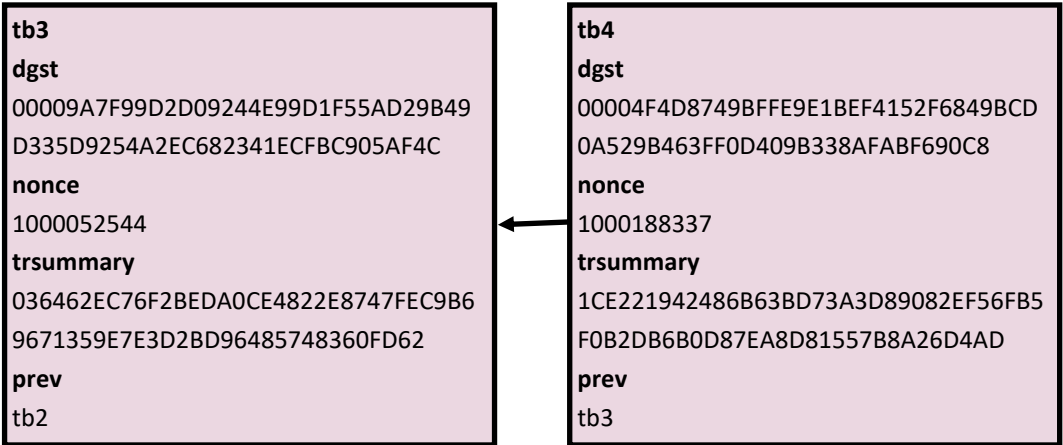


Sibling coupled path to root for t31: (to be outputted)



- Now, we have to compute a list of pair of strings of k+2 pairs, where k is the number of transaction blocks after the transaction block tB is present in. i.e. in our case after tb3, there is 1 block, therefore a list of 3 pair of strings is to be computed where each pair of string looks like: (t_i.dgst, t_i.previous.dgst + “#” + t_i.trsummary + “#” + t_i.nonce) and first block looks like (tb.previous.dgst, null).

Block Chain with and after tb3:



Therefore, the List of Pair of Strings will look like:
(to be outputted)

