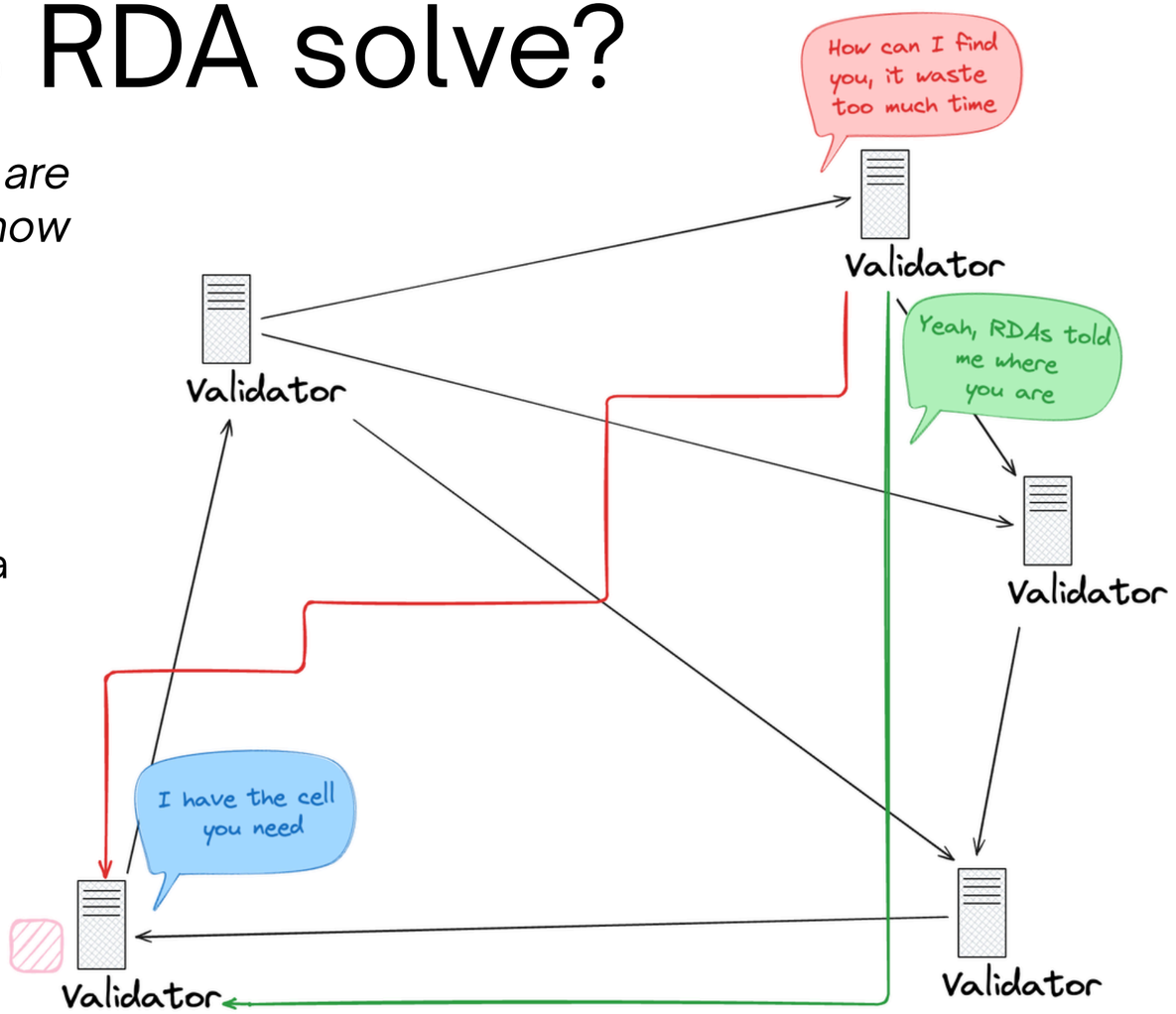
Robust Distributed Arrays

What problems does RDA solve?

“Current works on Data Availability Sampling are focusing on how to encode the data efficiency, not how to store and communicate efficiency and securely.”

RDAs defines and constructs an efficient and secure distributed storage networks in order to enable implementing DAS in real world. RDAs defines network protocols to help nodes answer DAS question - is that data available now?

- STORE: how to store a symbol
- GET: how to query a symbol
- JOIN: how to join network and sync needed symbols

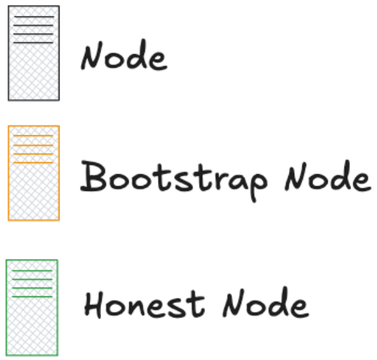


Network Topology of RDAs

“Using a Global Hash Function, a node can know its position on network, which looks like a matrix. A number of honest nodes are required for initialization, some of them are bootstrap nodes, which have to do more works than normal node.”

RDAs divides network into row subnets and column subnets (Viltalik used to refer to this solution):

- All bootstrap nodes must lie on each rows, others just have to lie on some rows.
- All nodes will also lie on a pre-assigned column.

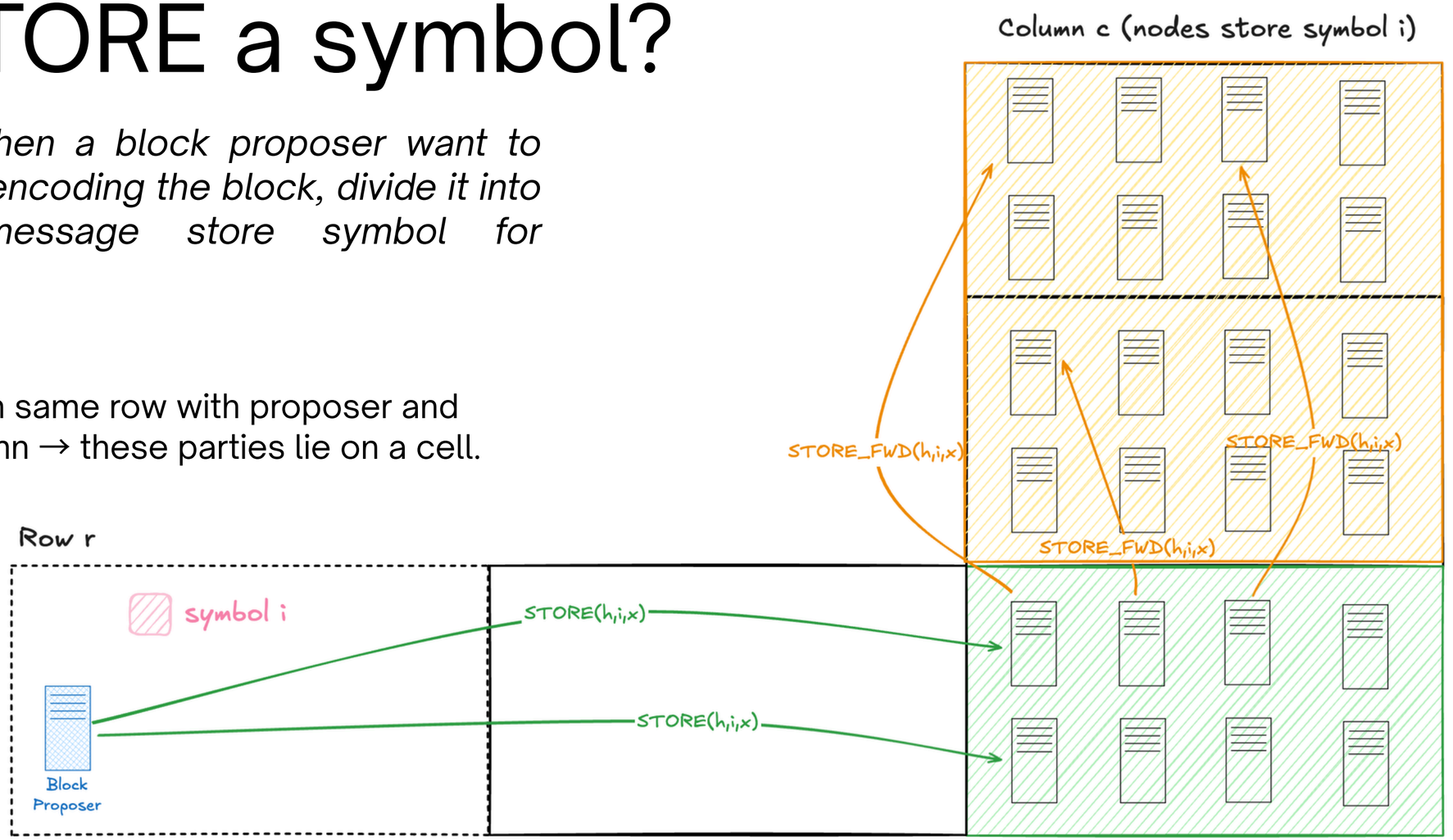


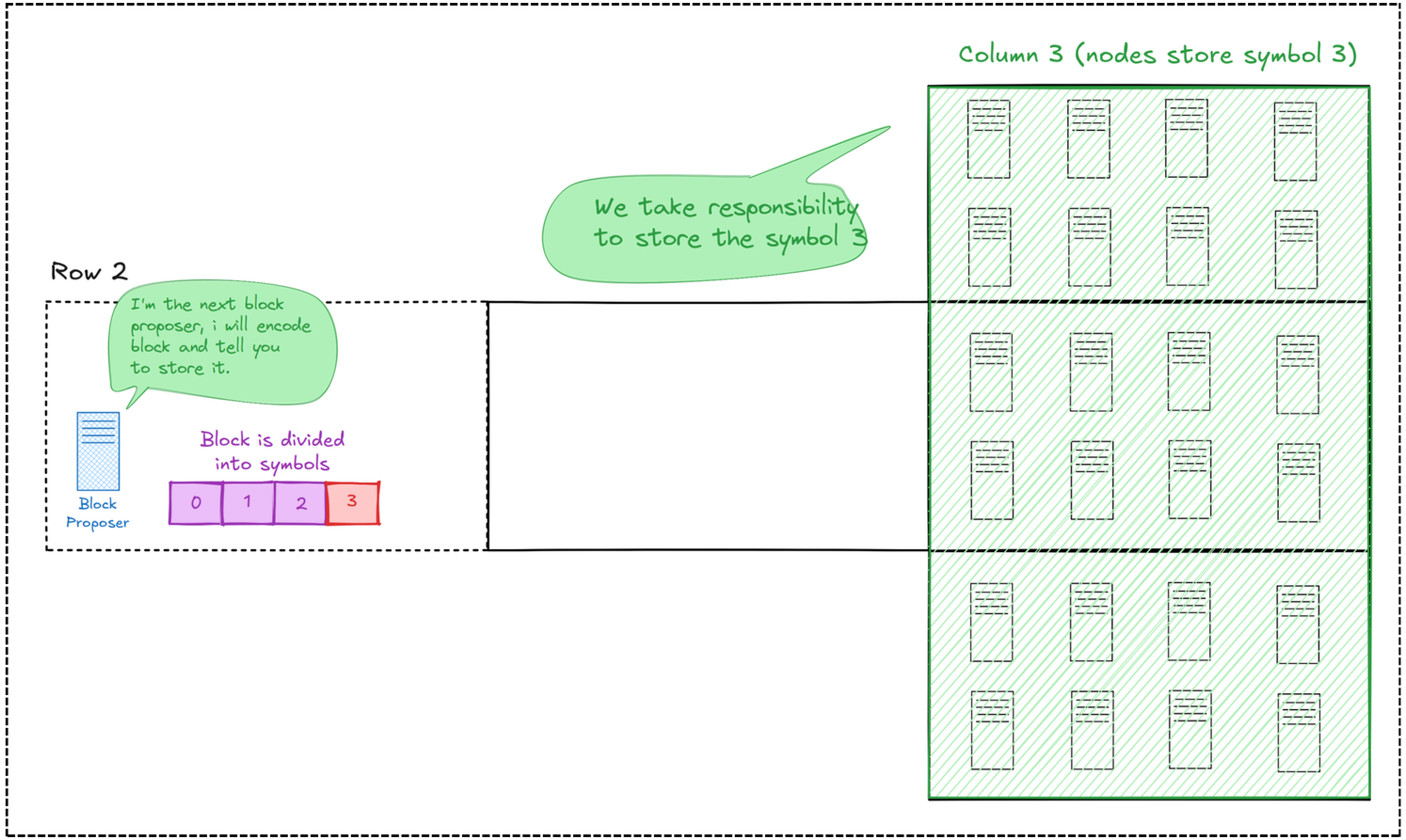
How to STORE a symbol?

“Store action occurs when a block proposer want to propose a new block, it encoding the block, divide it into chunks and send message store symbol for corresponding nodes.”

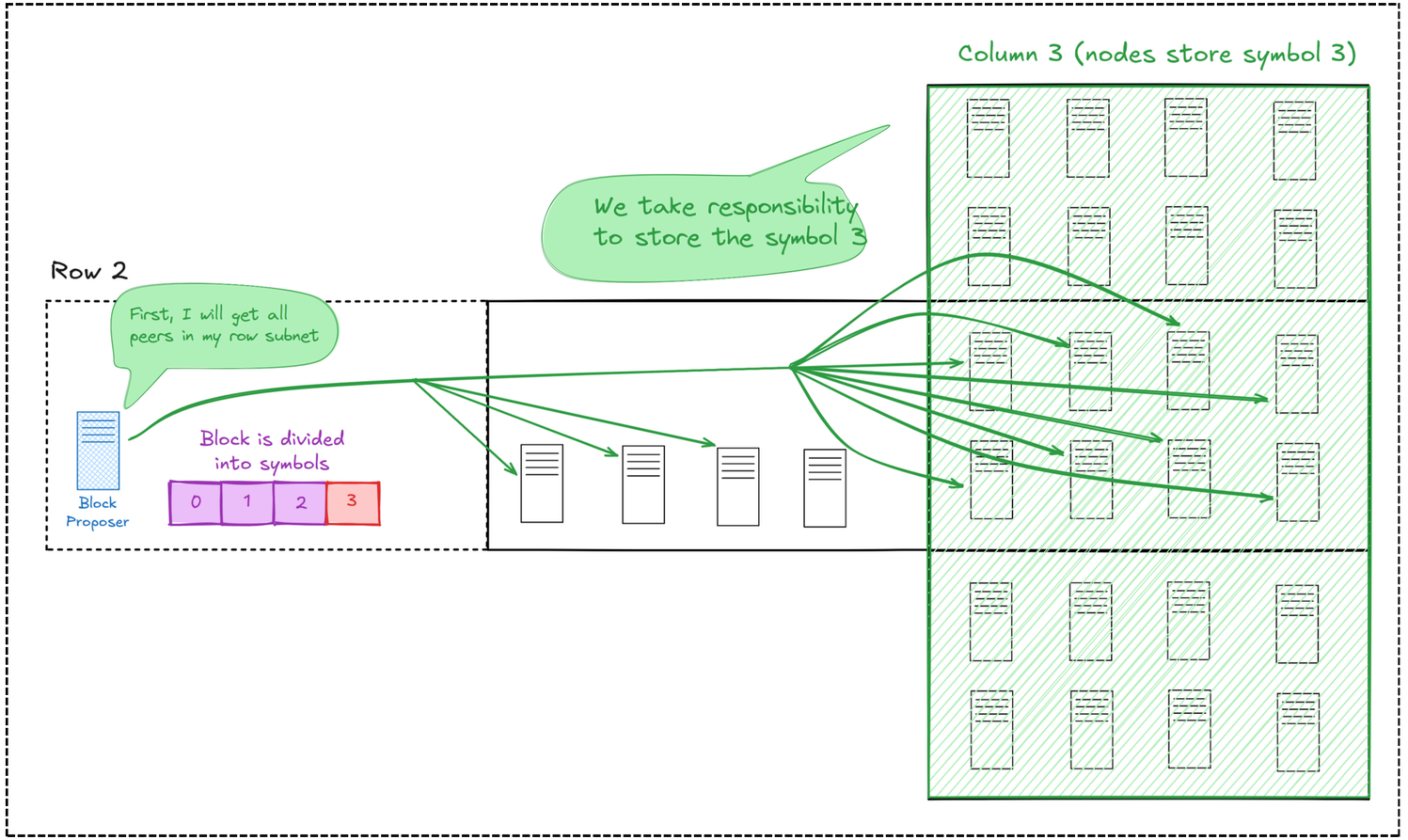
Store protocol:

- Find parties which lie on same row with proposer and also on the target column → these parties lie on a cell.
- Send msg store to them.
- These parties will forward store msg to all nodes on the same col subnet.
- At last, we have a col subnet store that symbol.

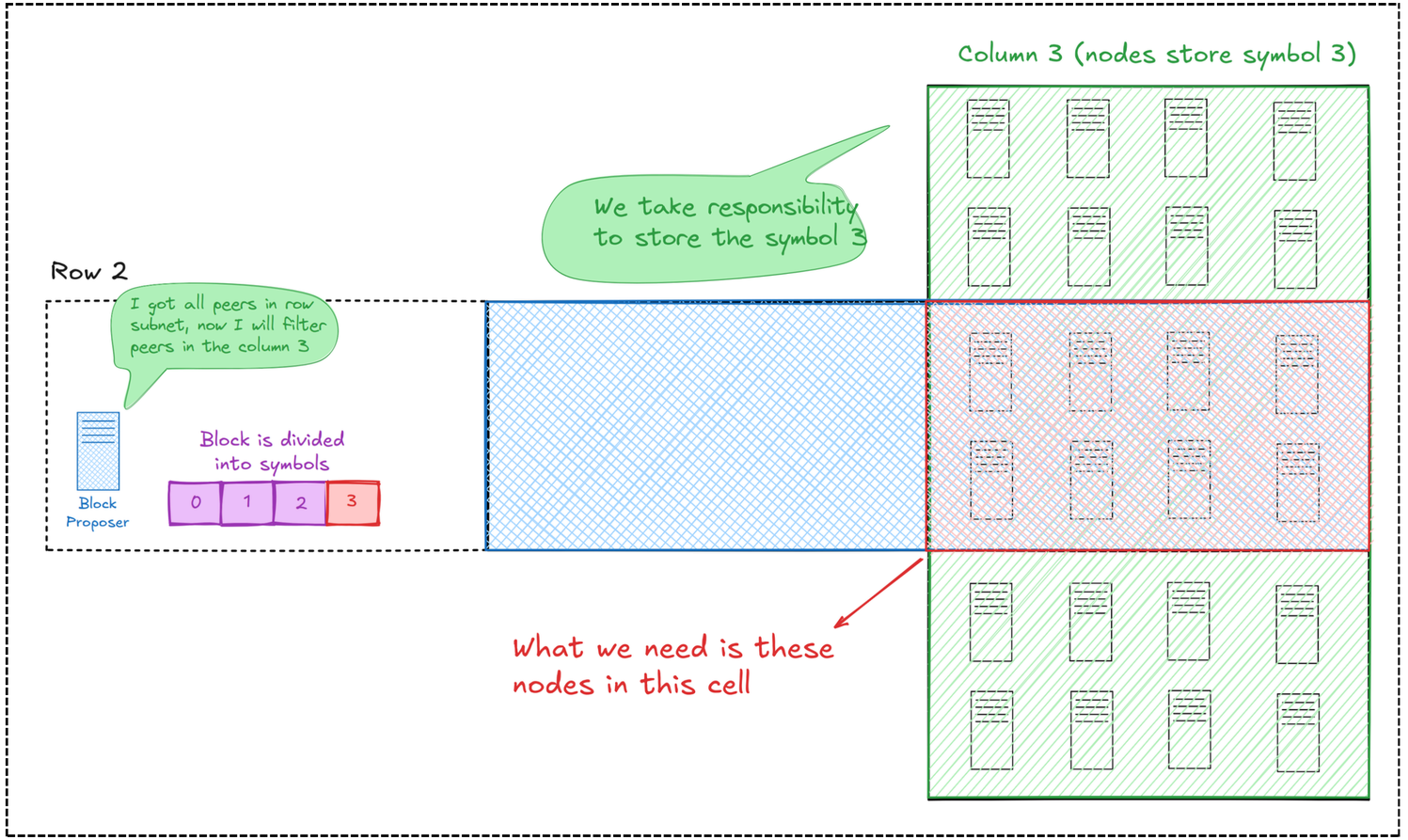




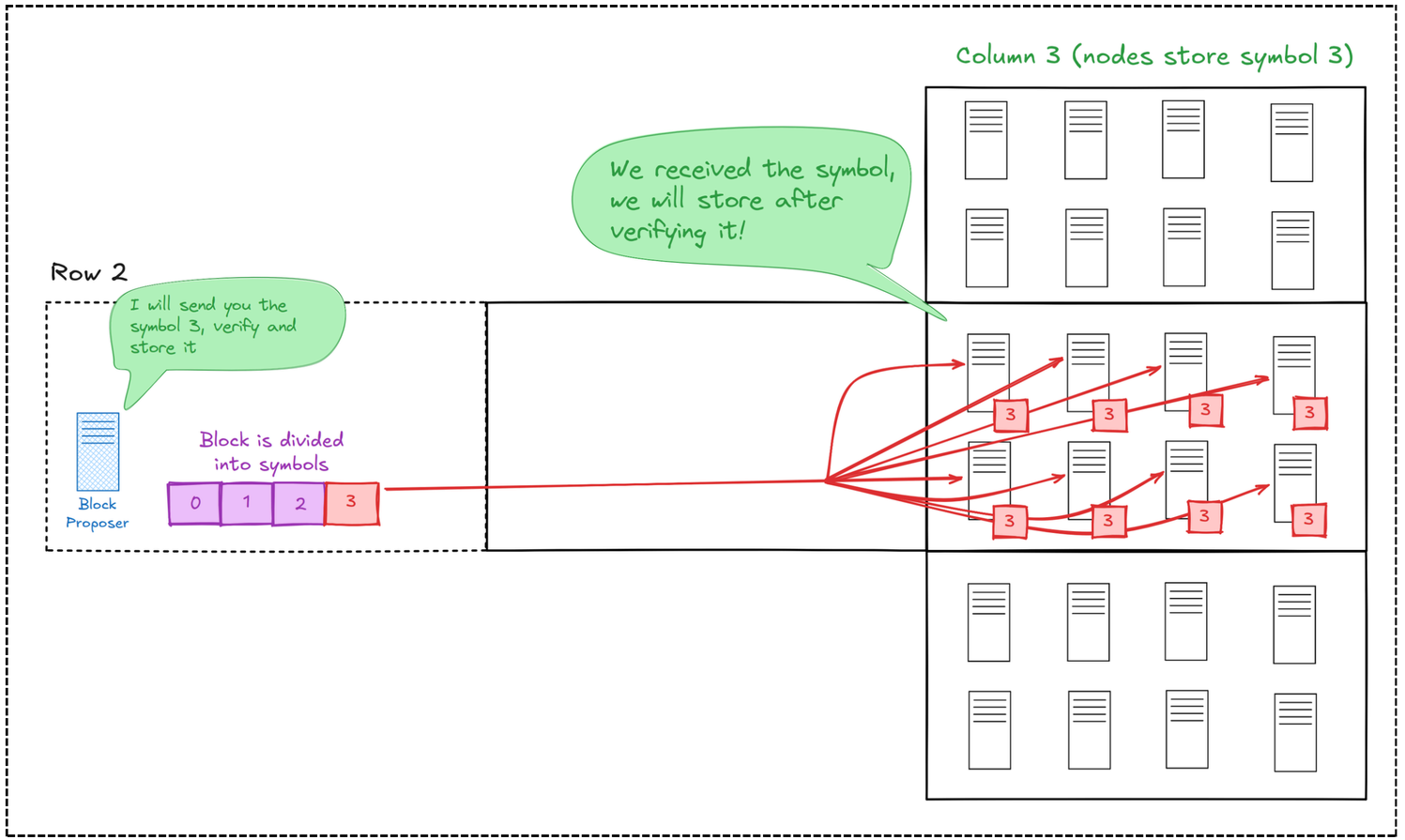
STORE protocol in detail - 1



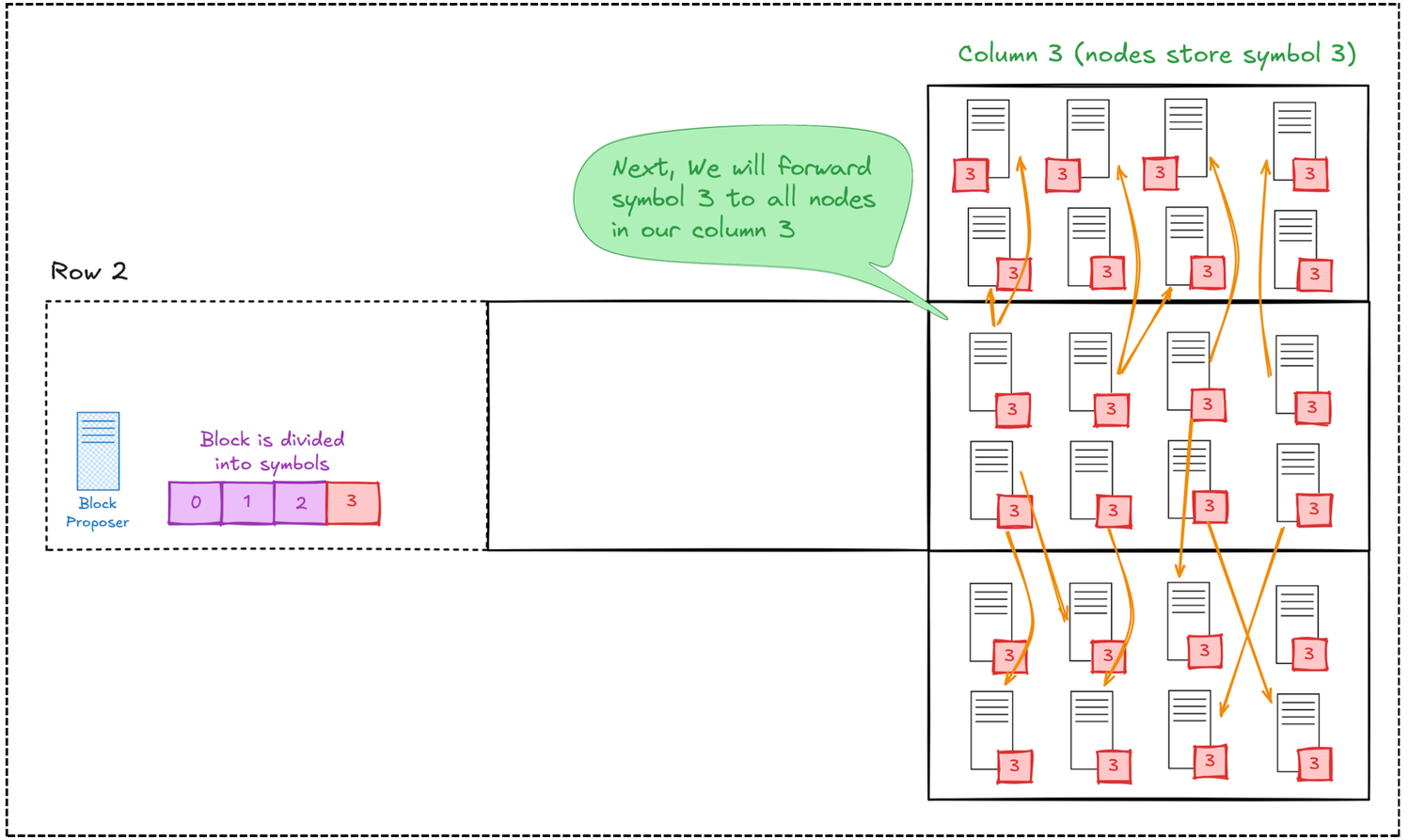
STORE protocol in detail - 2



STORE protocol in detail - 3



STORE protocol in detail - 4



STORE protocol in detail - 5

How to GET a symbol?

“Get action is triggered when an arbitrary node wishes to perform Data Availability Sampling. Specifically, the Get action attempts to retrieve a symbol from the corresponding column and verifies its validity to determine the success of the DAS.”

Get protocol:

- Find parties which lie on same row with proposer and also on the target column

→ these parties lie on a cell.

- Send msg get to

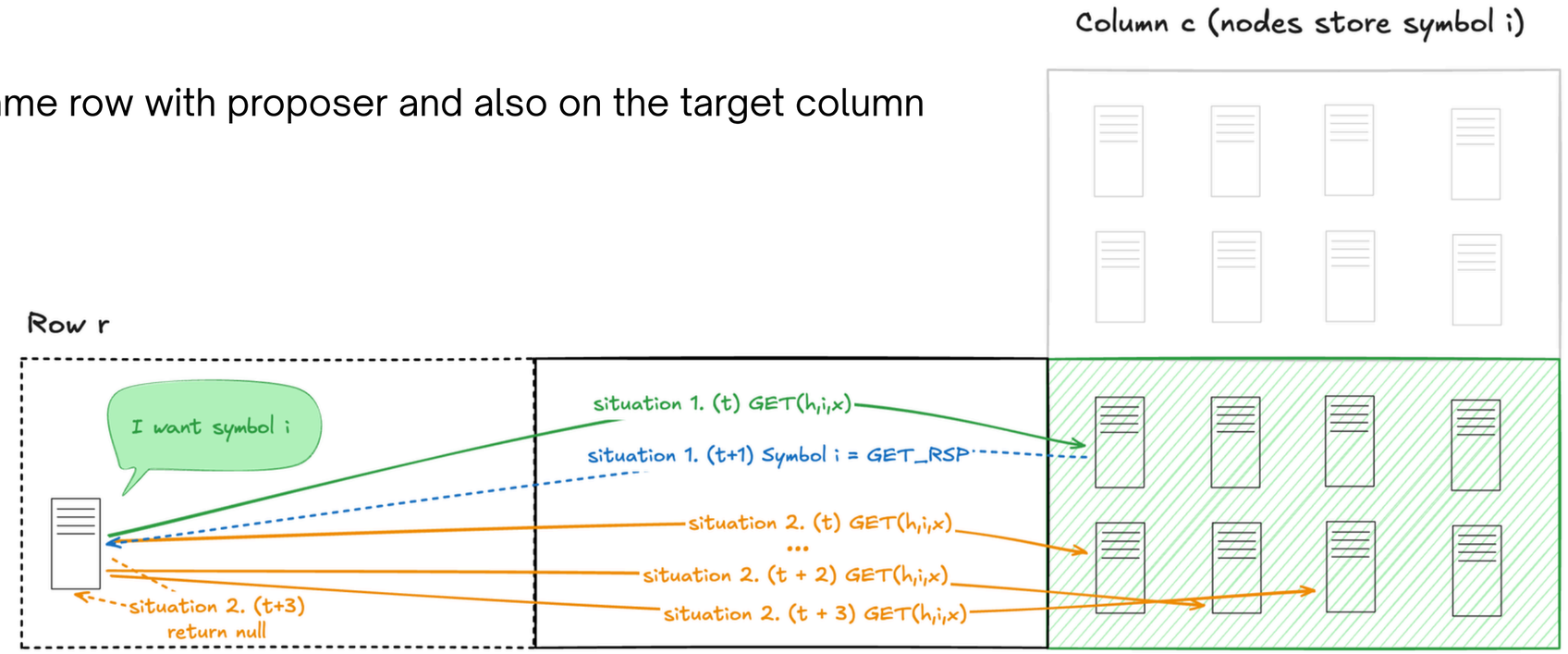
them.

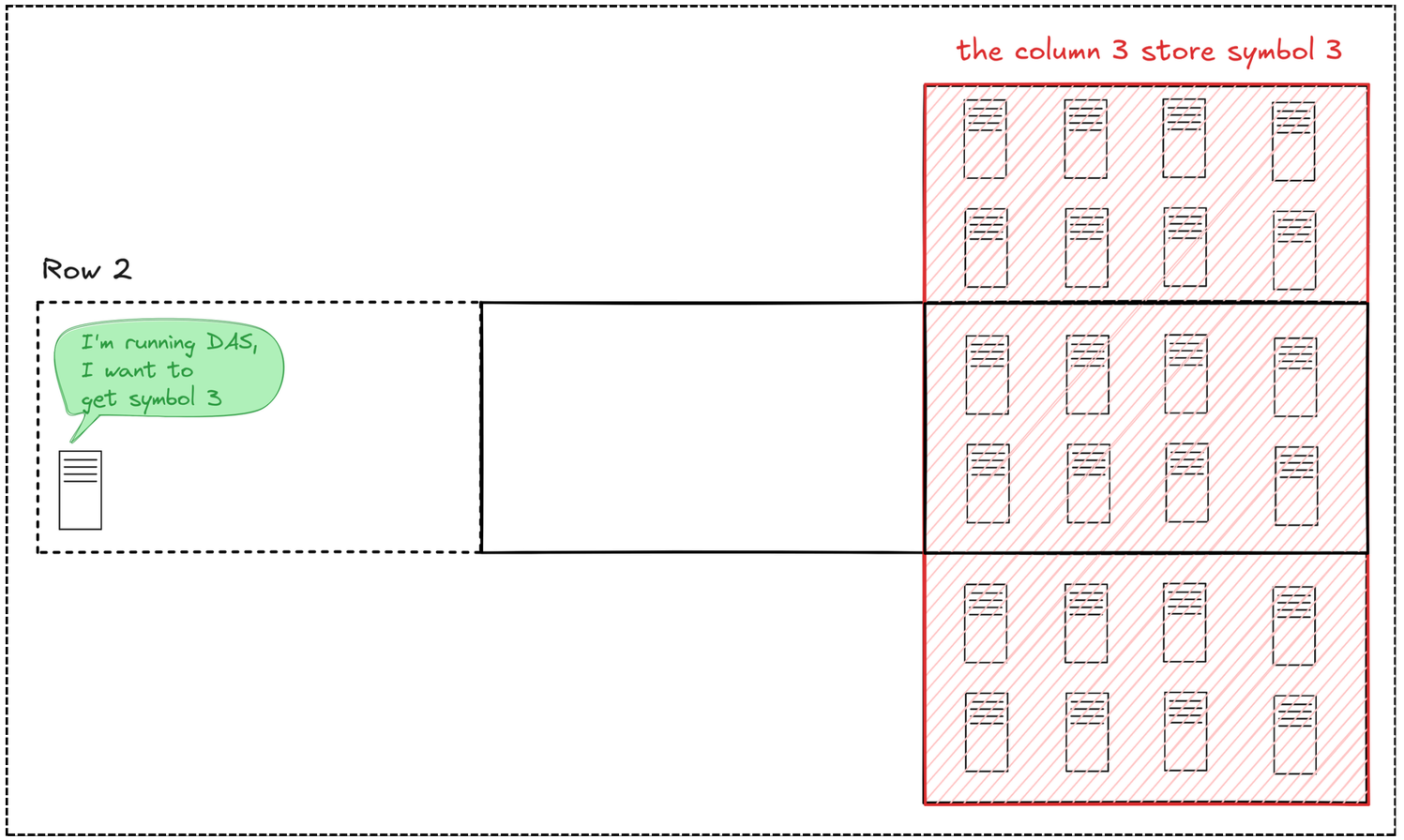
- If before timeout limit,

enough nodes send the
get response (that symbol),
the DAS would be success.

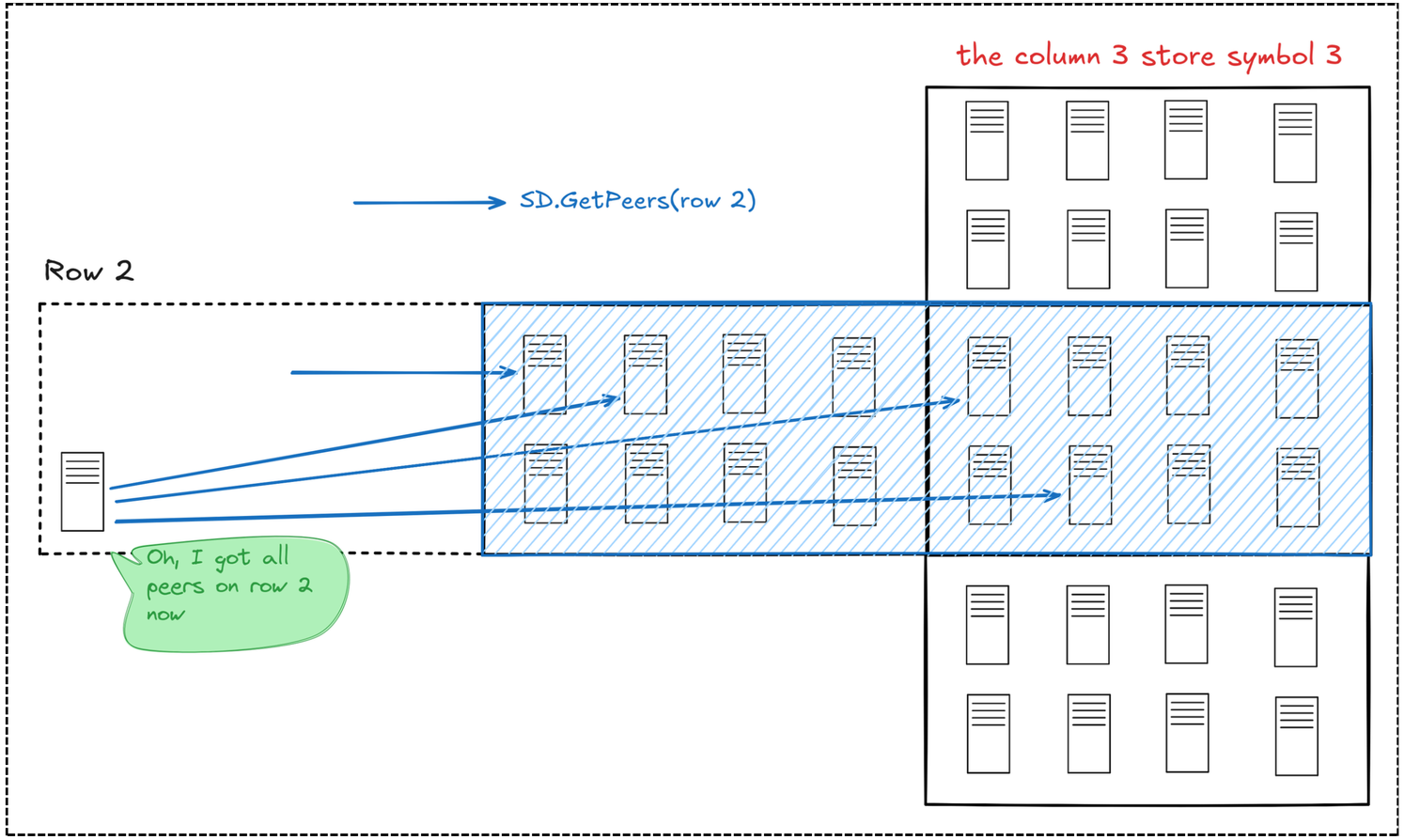
- Otherwise, the DAS

would be failed.
There is at least one honest
node in each cell at a time T.

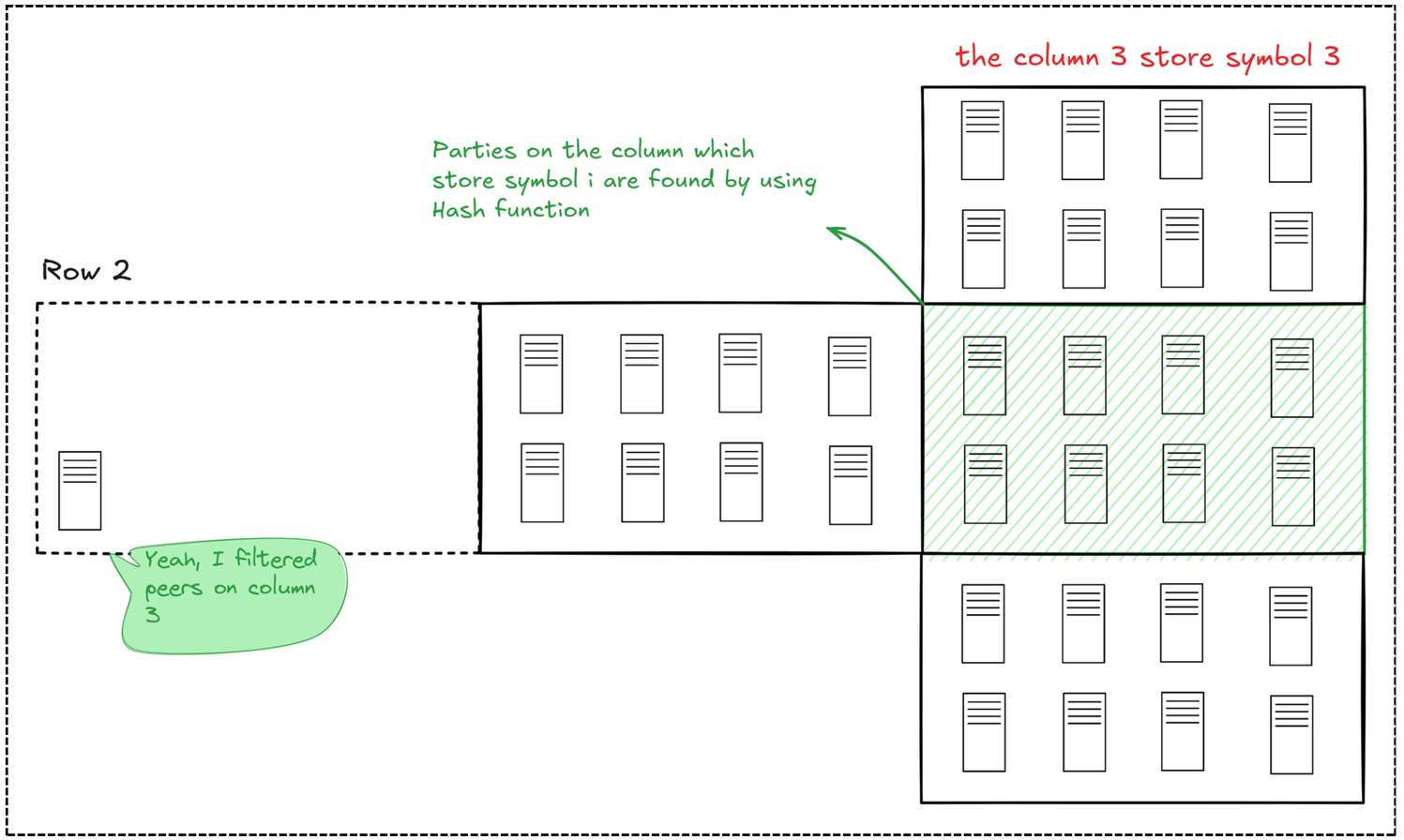




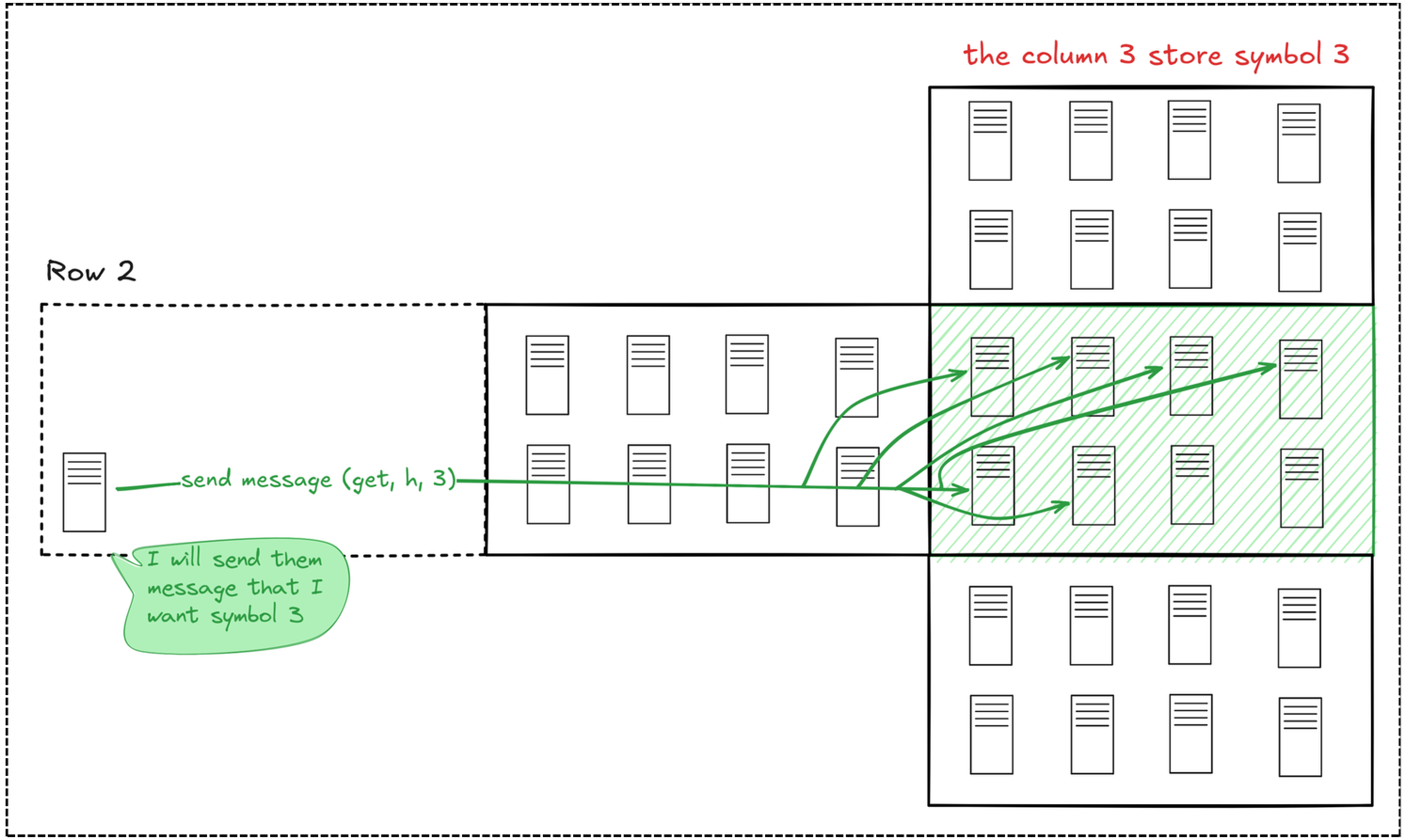
GET protocol in detail - 1



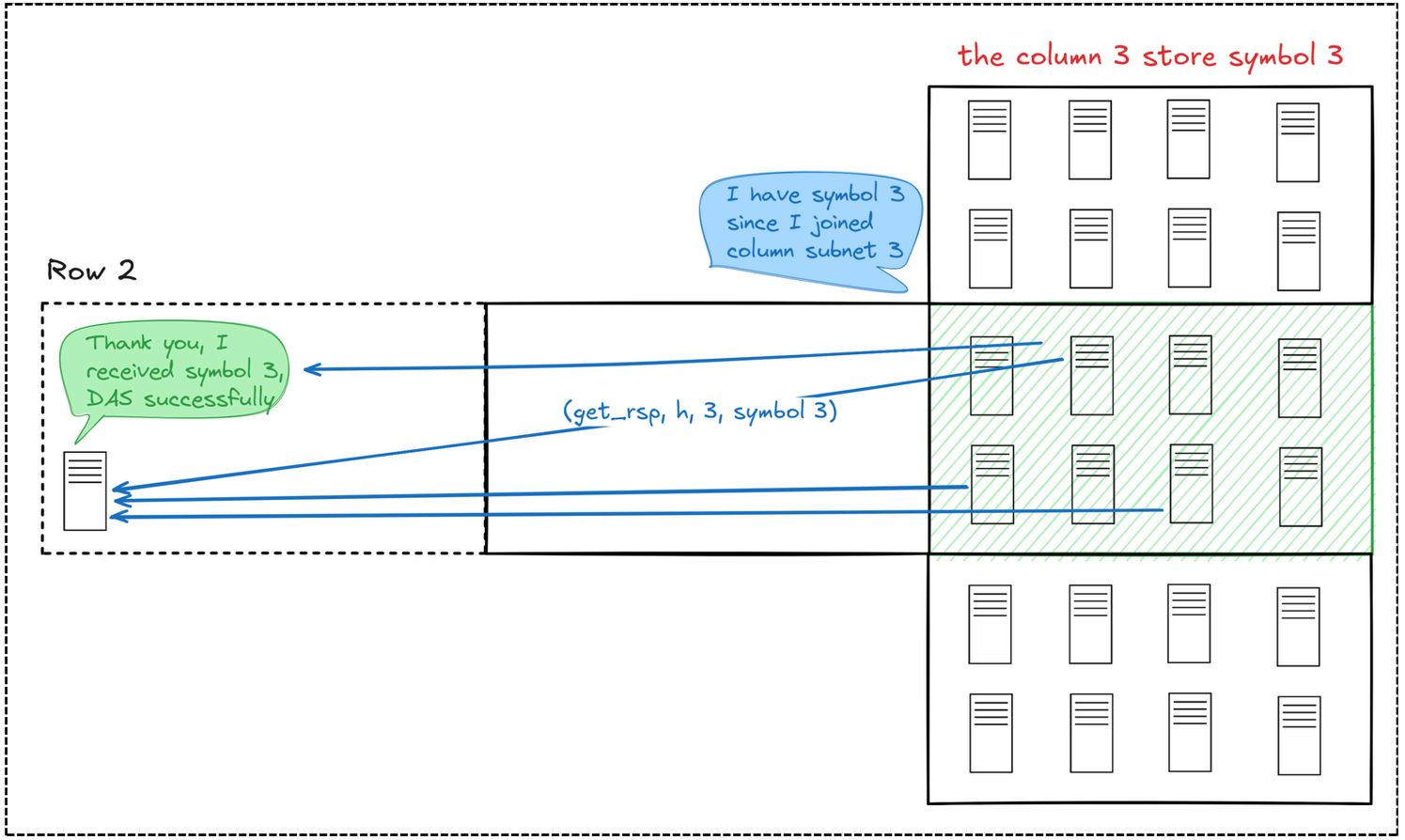
GET protocol in detail - 2



GET protocol in detail - 3

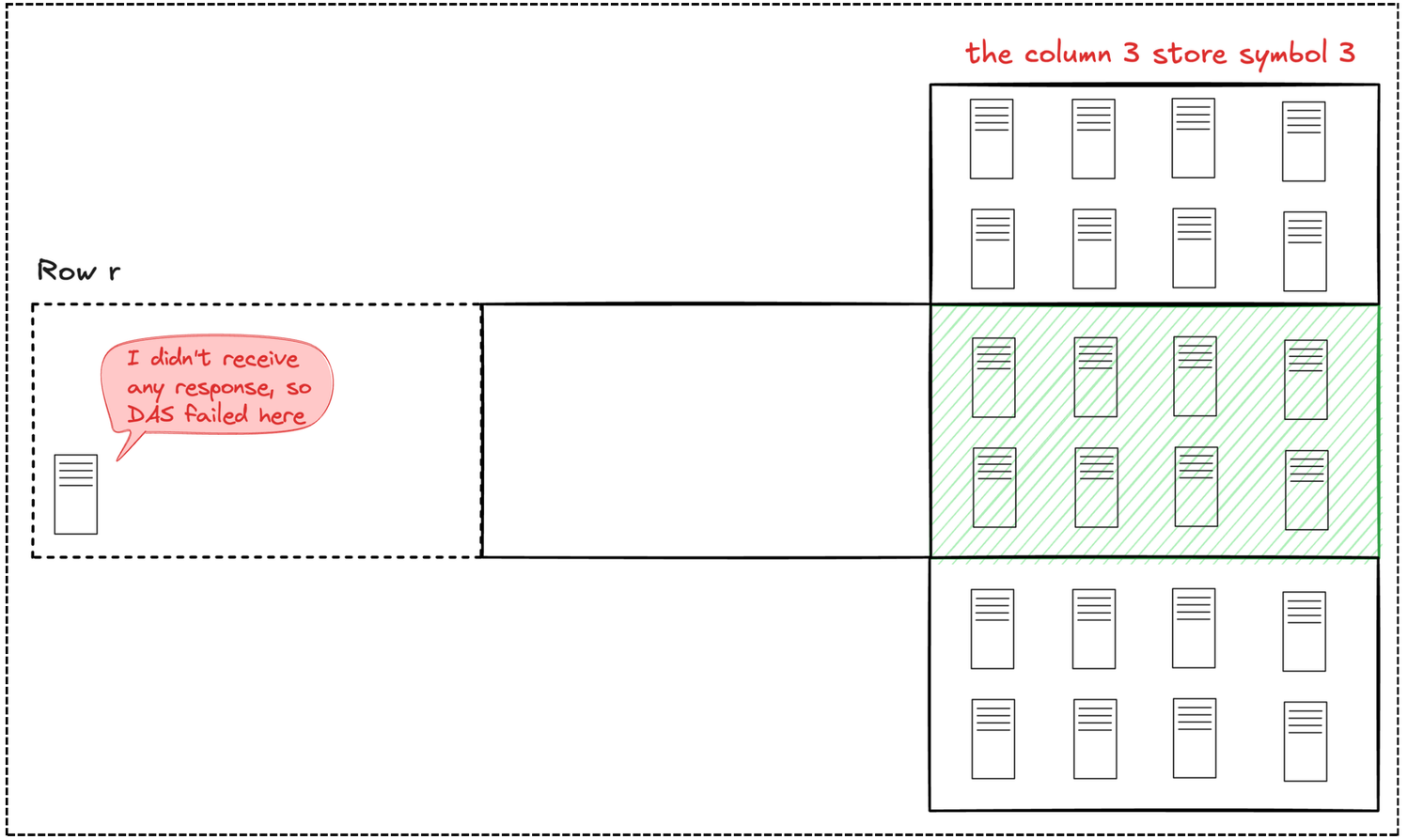


GET protocol in detail - 4



GET protocol in detail - 5

Success case



GET protocol in detail - 6

Failed case when honest node does not response on time

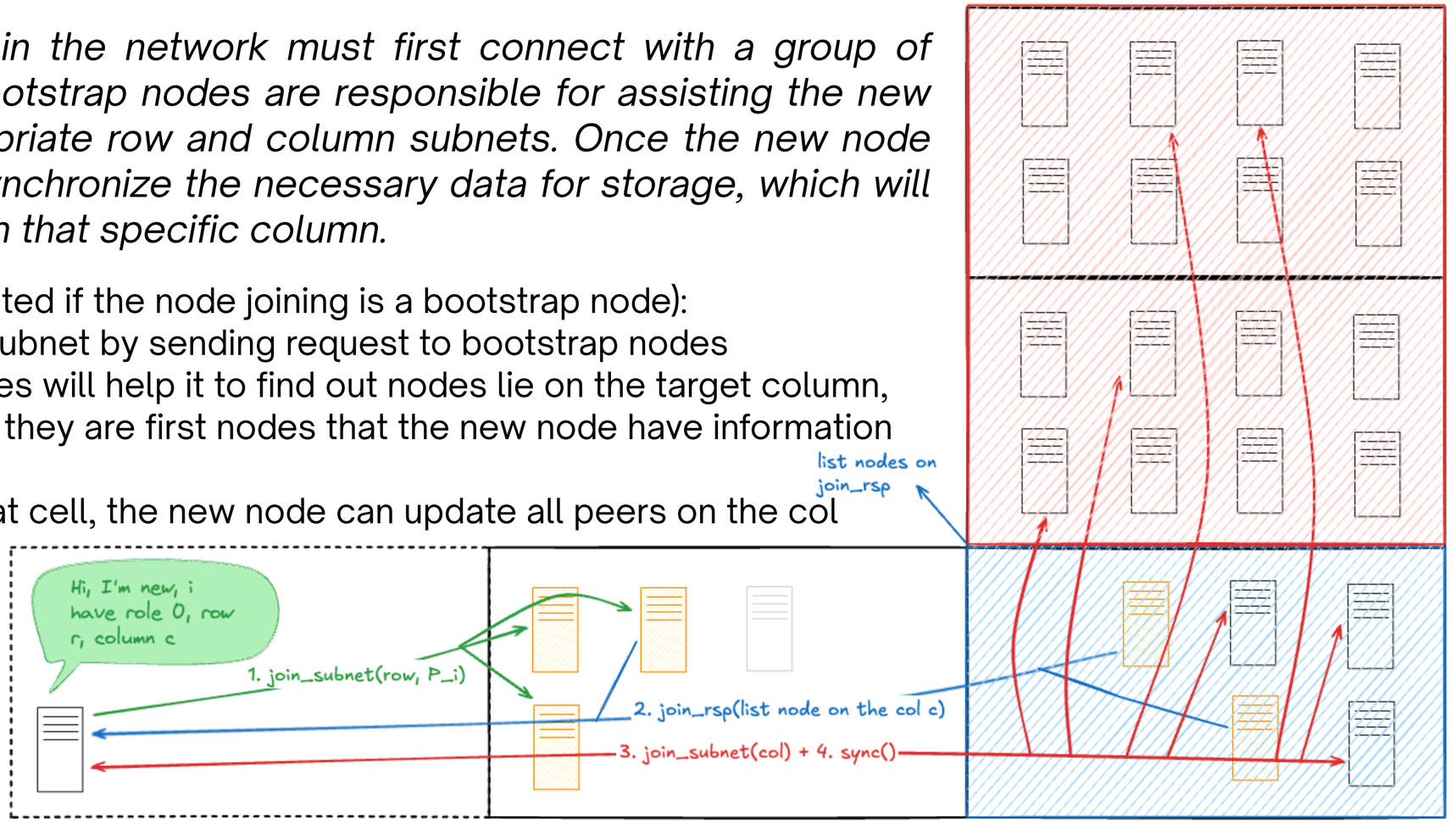
How to solve ?

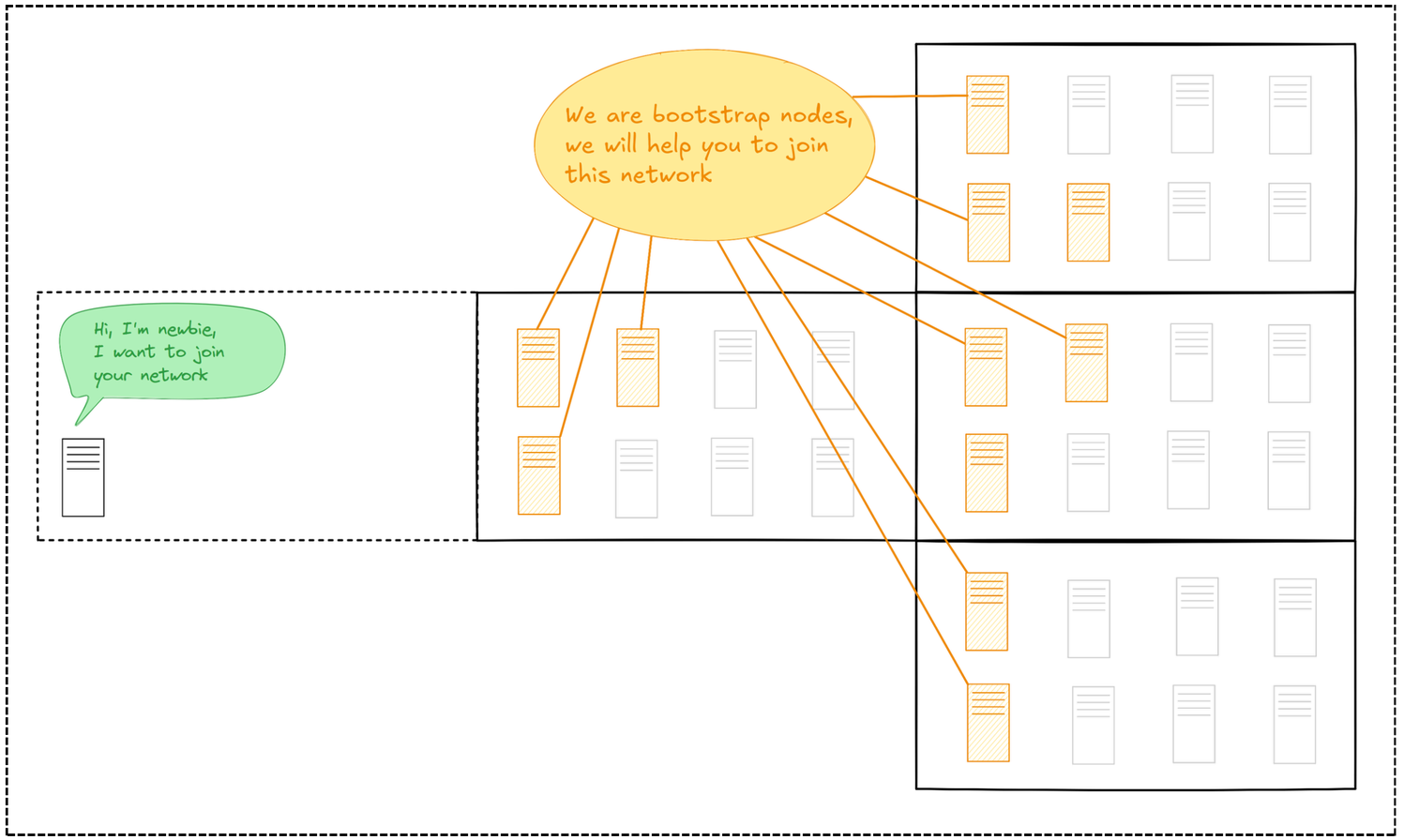
How does a new node join network?

A new node seeking to join the network must first connect with a group of bootstrap nodes. These bootstrap nodes are responsible for assisting the new node in entering the appropriate row and column subnets. Once the new node successfully joins, it can synchronize the necessary data for storage, which will correspond to the symbol in that specific column.

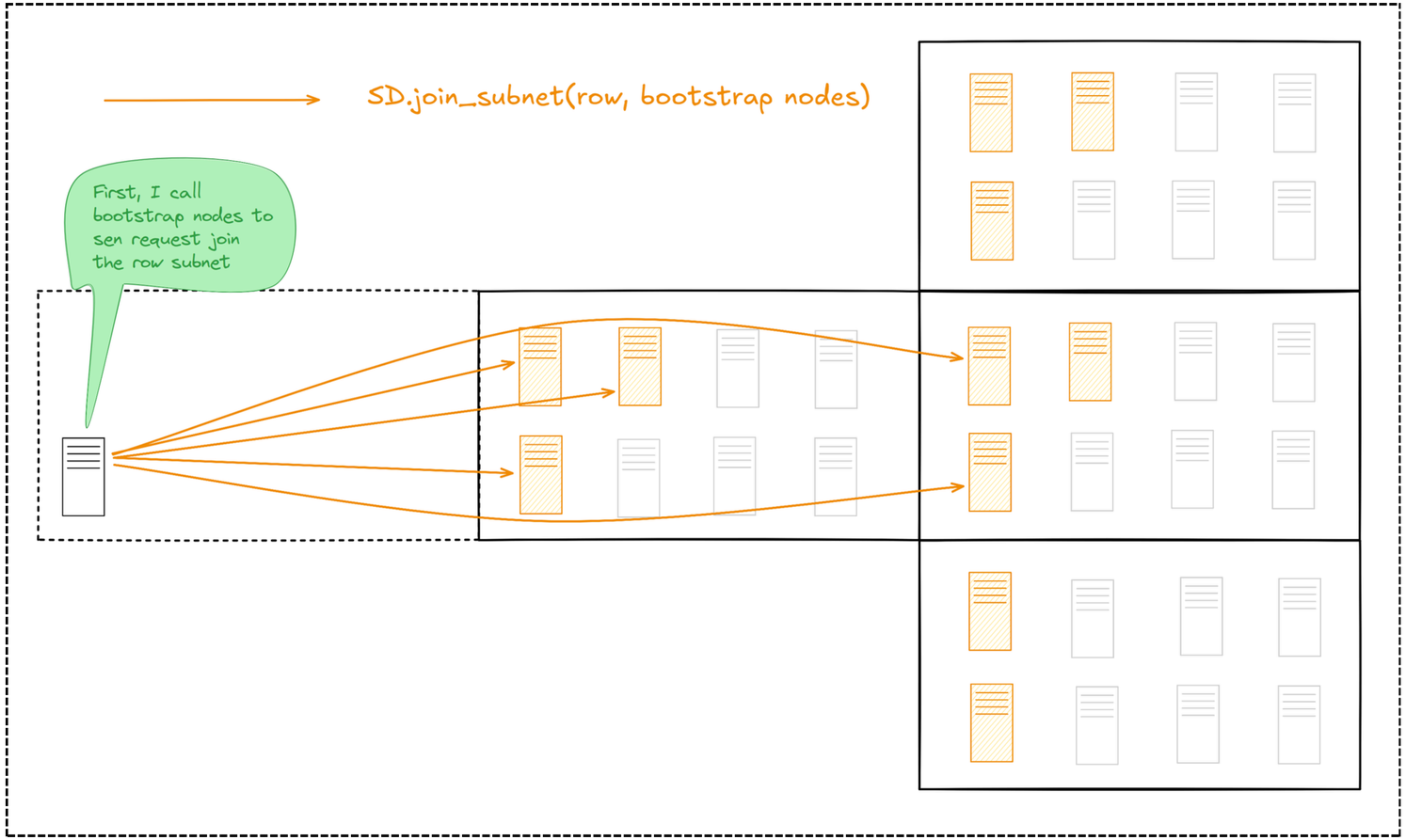
Join protocol (A little complicated if the node joining is a bootstrap node):

- Firstly, this node join row subnet by sending request to bootstrap nodes
- Next, these bootstrap nodes will help it to find out nodes lie on the target column, which are on the blue cell, they are first nodes that the new node have information about its column
- By asking the nodes on that cell, the new node can update all peers on the col
- At last, it request for syncing the symbol needed to store.

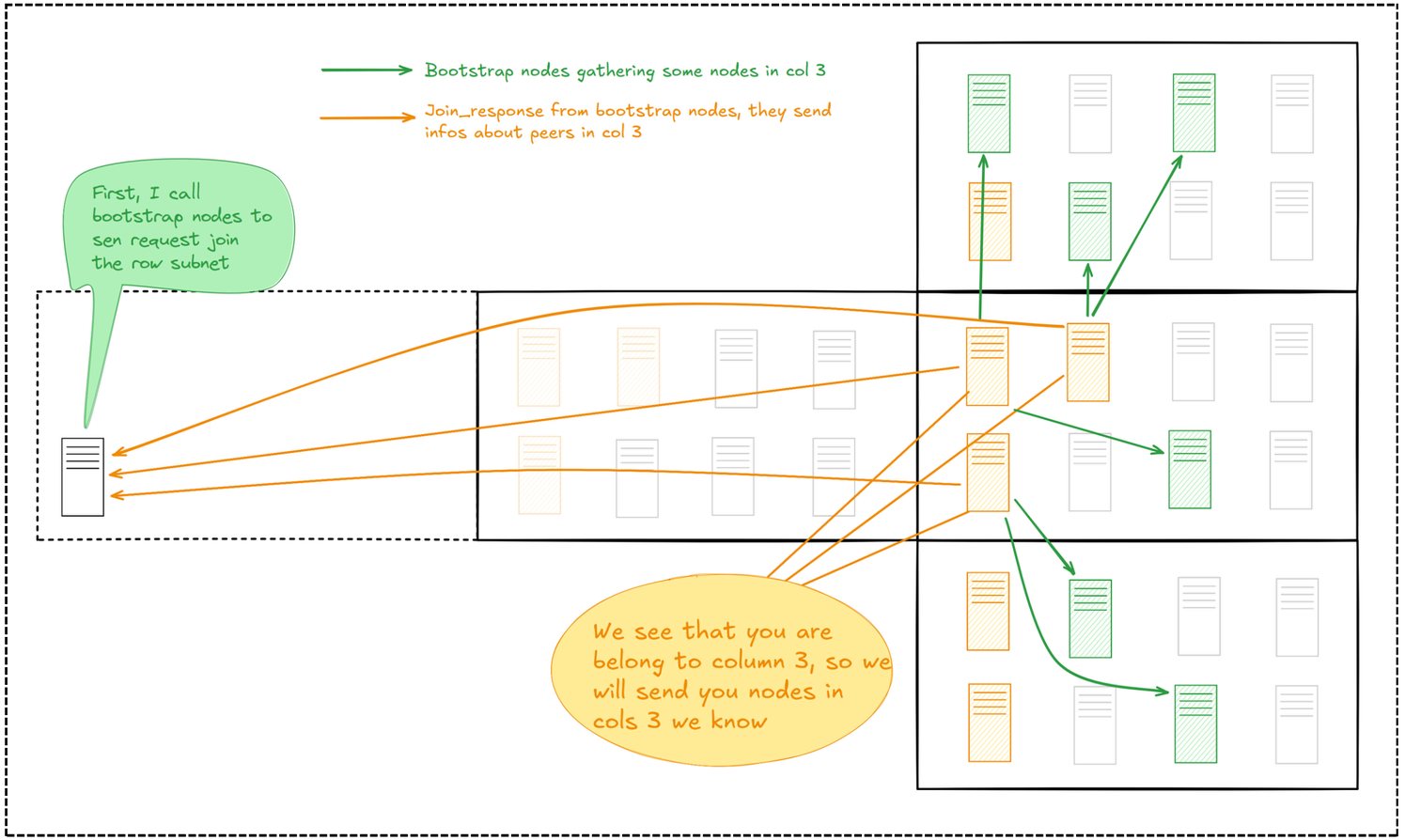




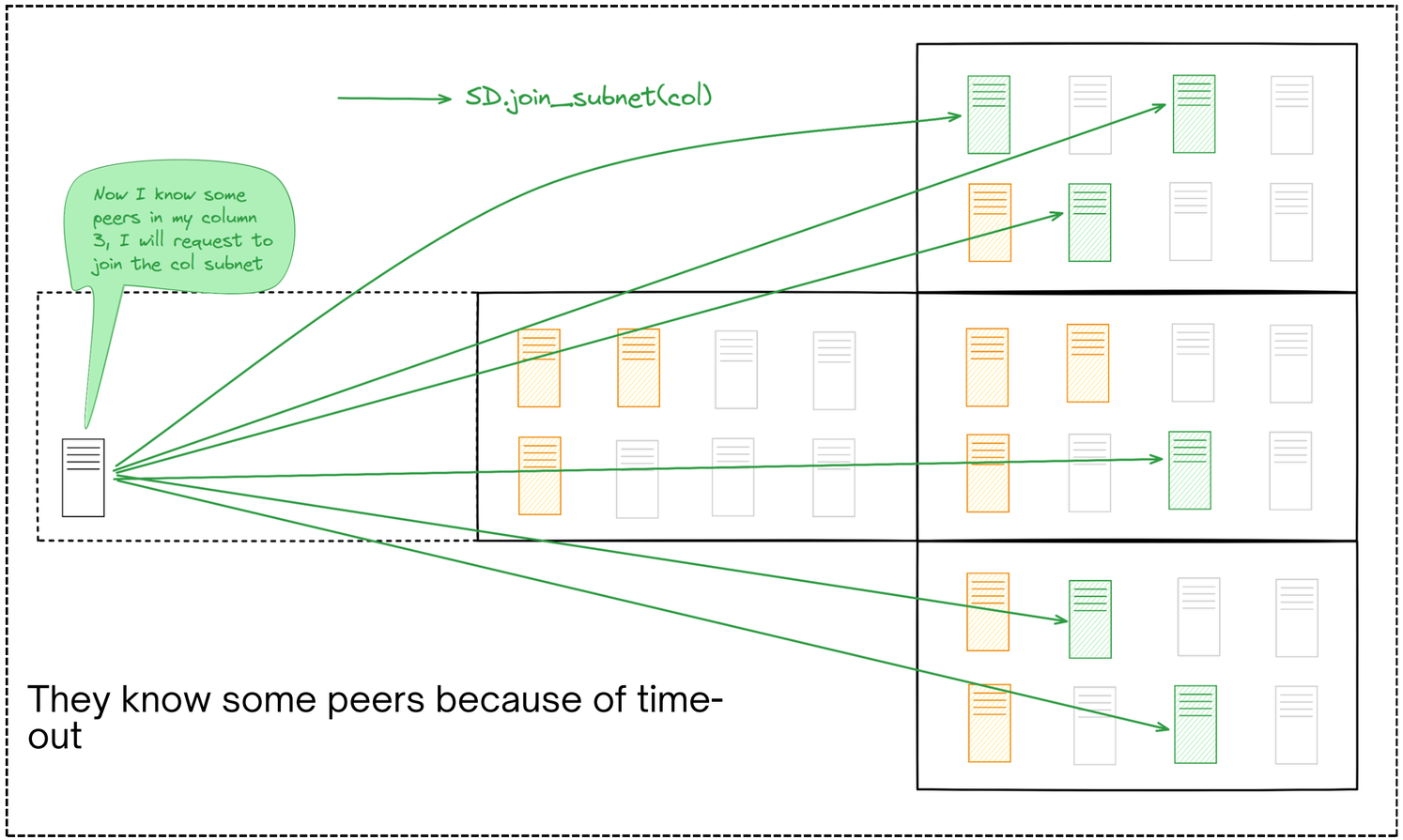
JOIN protocol in detail - 1



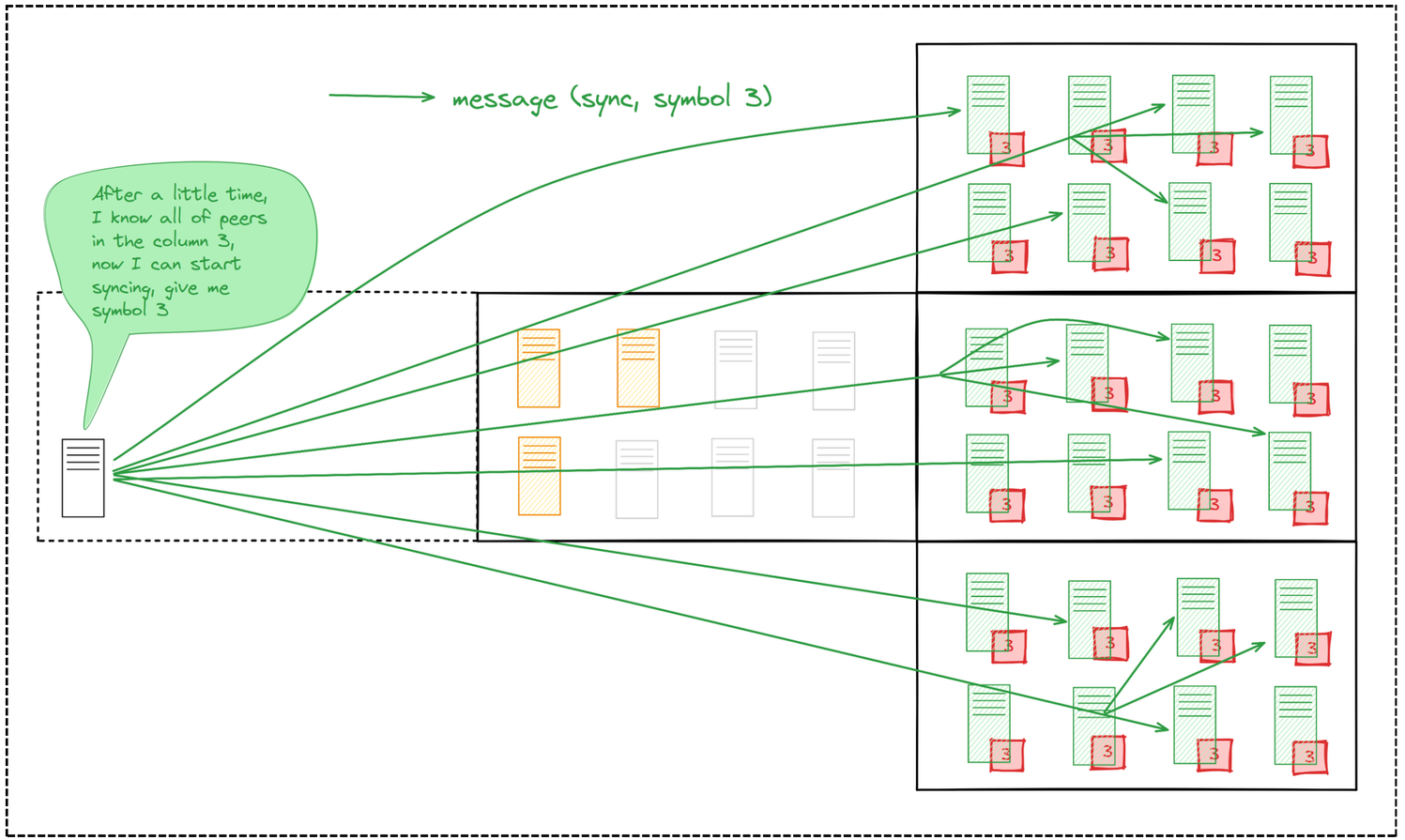
JOIN protocol in detail - 2



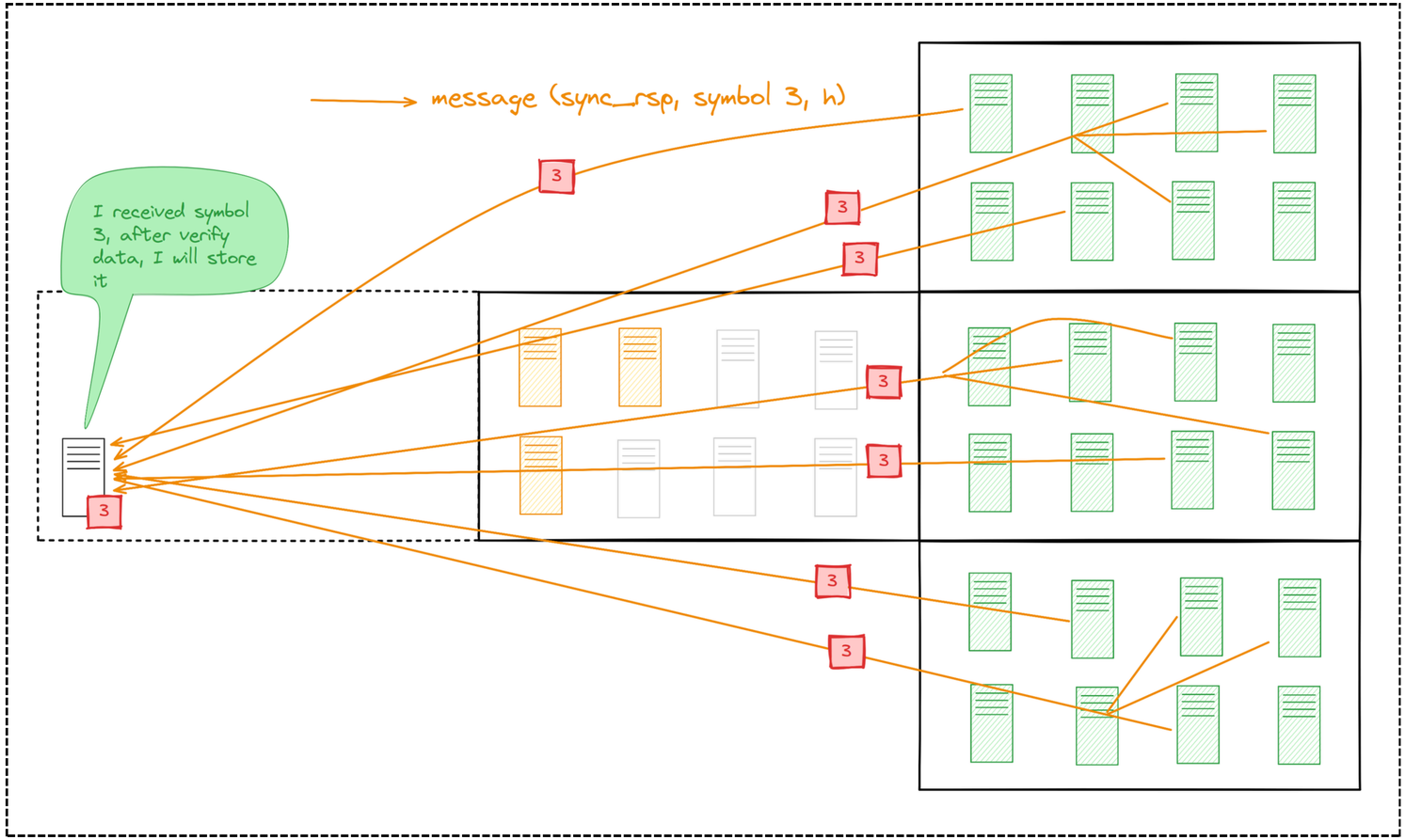
JOIN protocol in detail - 3



JOIN protocol in detail - 4



JOIN protocol in detail - 5



JOIN protocol in detail - 6

Overall

1. Advantages:

+) RDA can replace with current DHT + Gossipsub of Ethereum with **low latency queries**, where DHT is peer discovery domain and Gossipsub is data propagating domain to improve security for DHT

+) RDA has proved about it’s security and robustness

2. Disadvantages:

+) If RDA is applied for Ethereum, a lot of upgrades has to be done.

+) Currently, there is no proof that with RDA, sampling time + distributing blob for 140MB block is under 4s

+) One node has to keep a large amount of neighbors - O(N) - not scalable - peers overhead