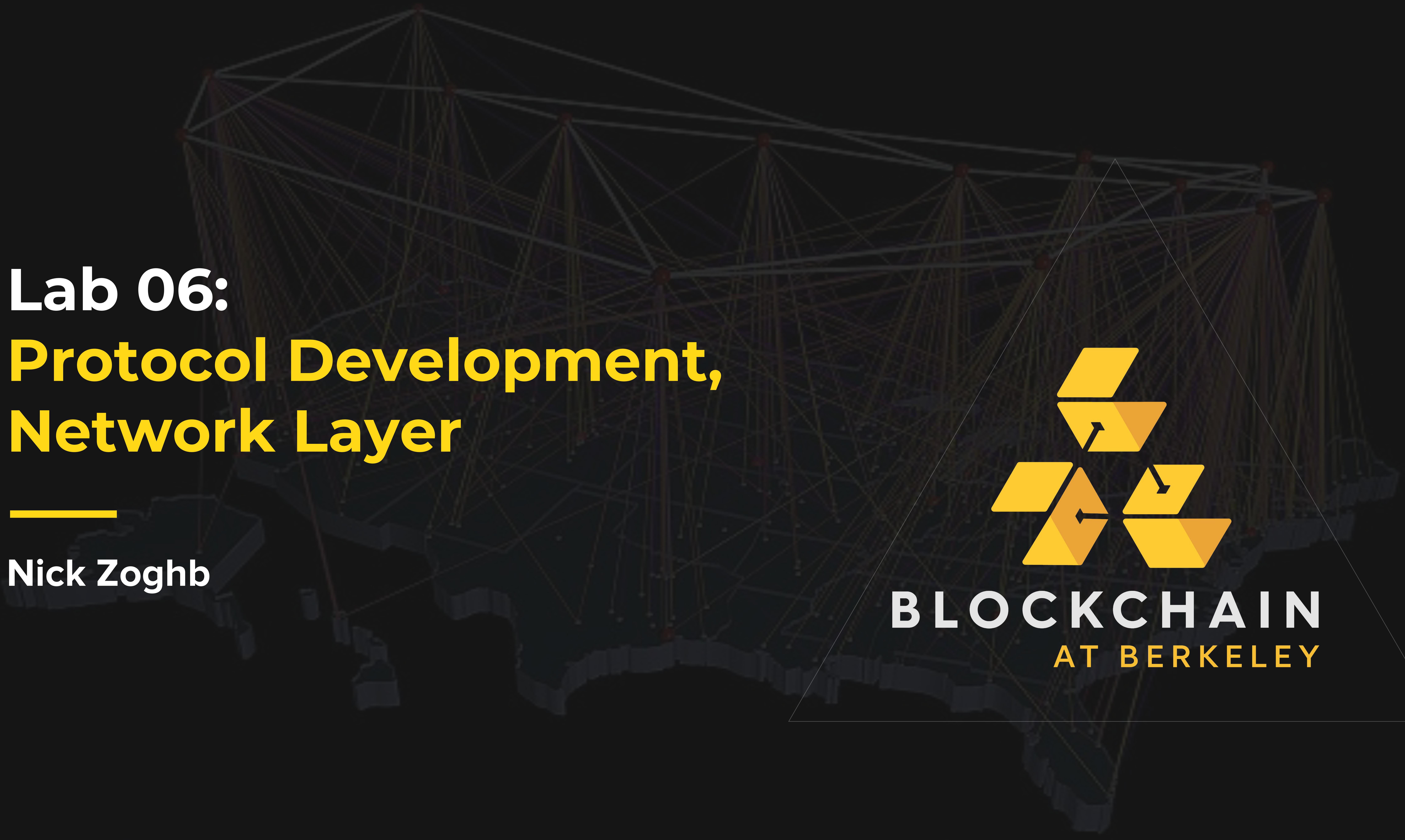
Lab 06:
Protocol Development, Network Layer

Nick Zoghb



BLOCKCHAIN AT BERKELEY

LAB OUTLINE

1 ▶ REFRESHER

2 ▶ THE (NEAR) FUTURE

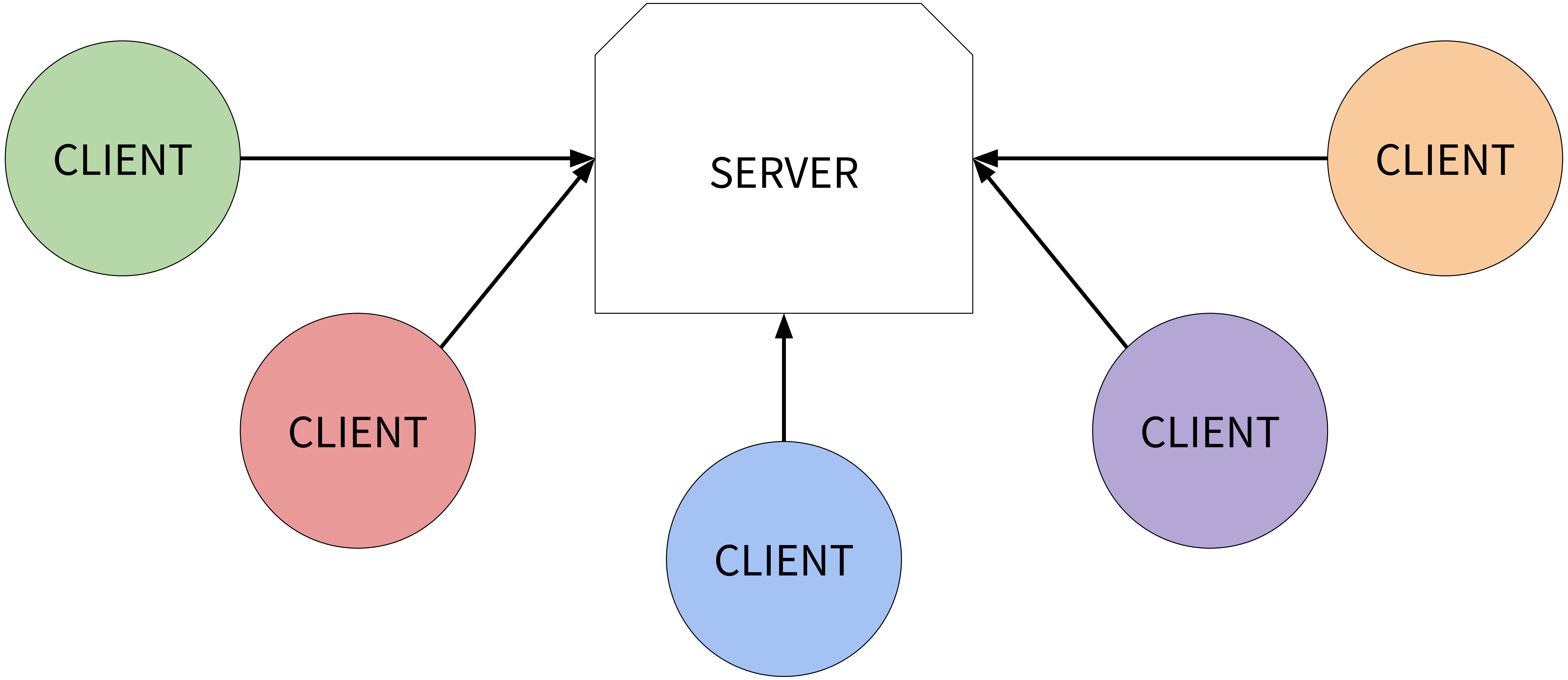
3 ▶ ASSIGNMENT

1

REFRESHER

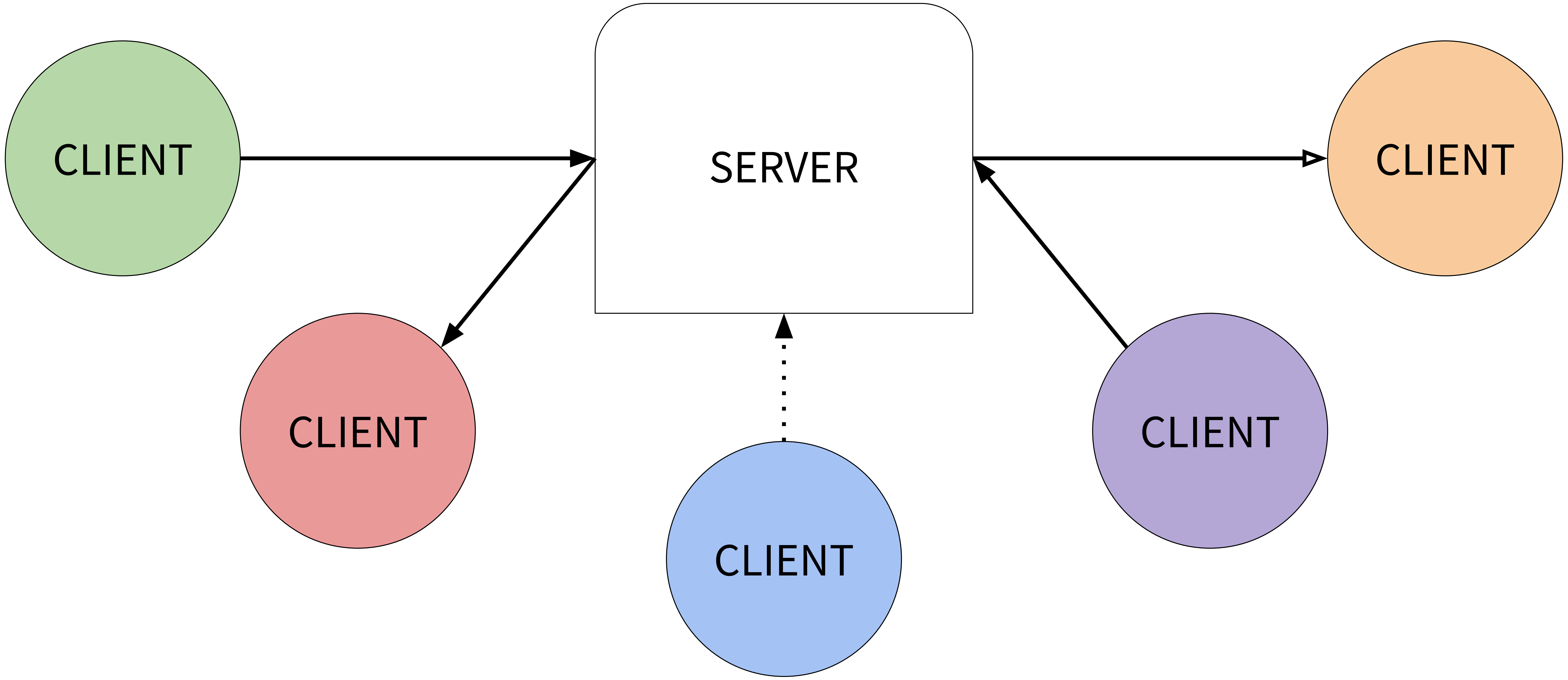
REFRESHER

CENTRALIZED EXAMPLE



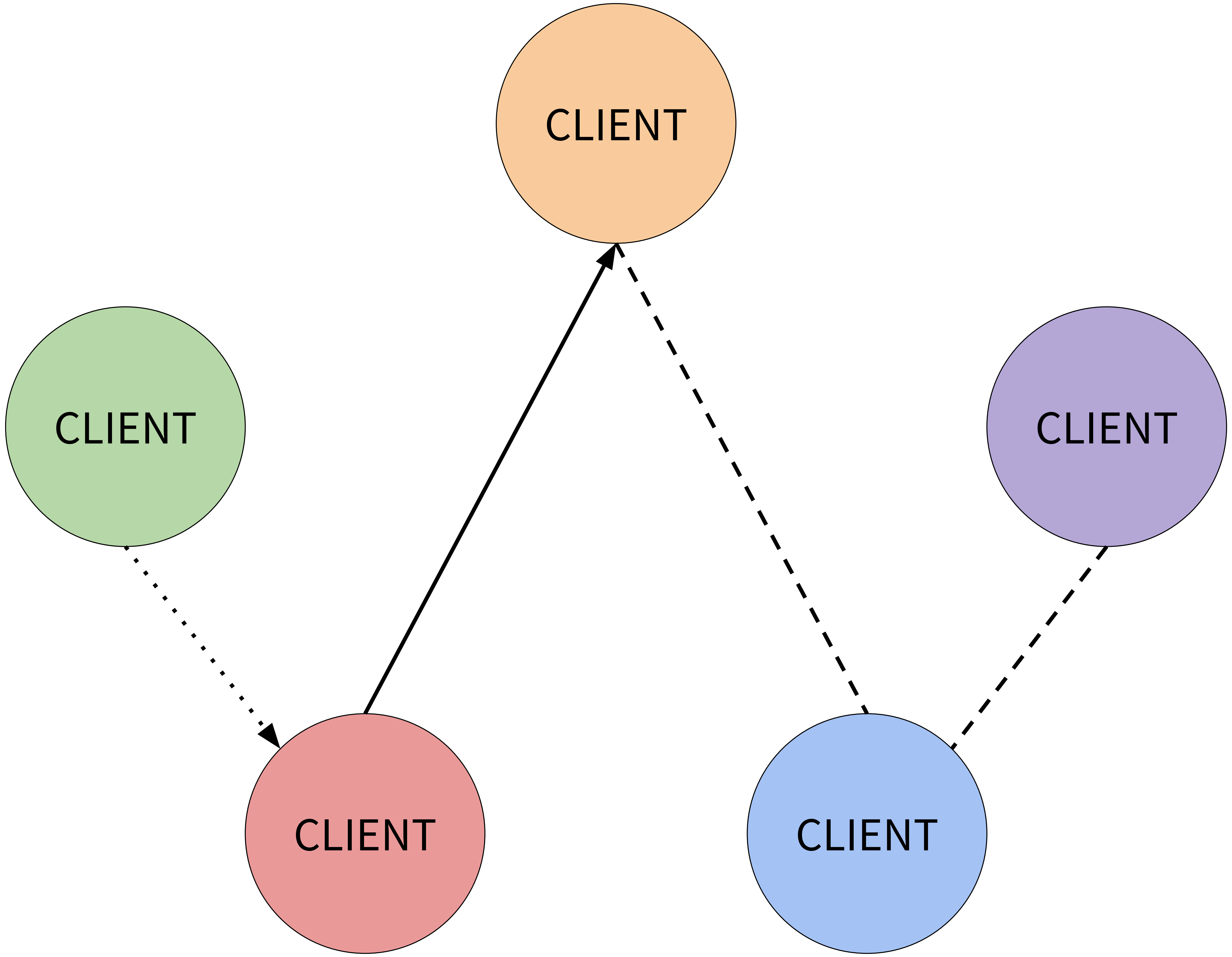
REFRESHER

SEMI-P2P EXAMPLE



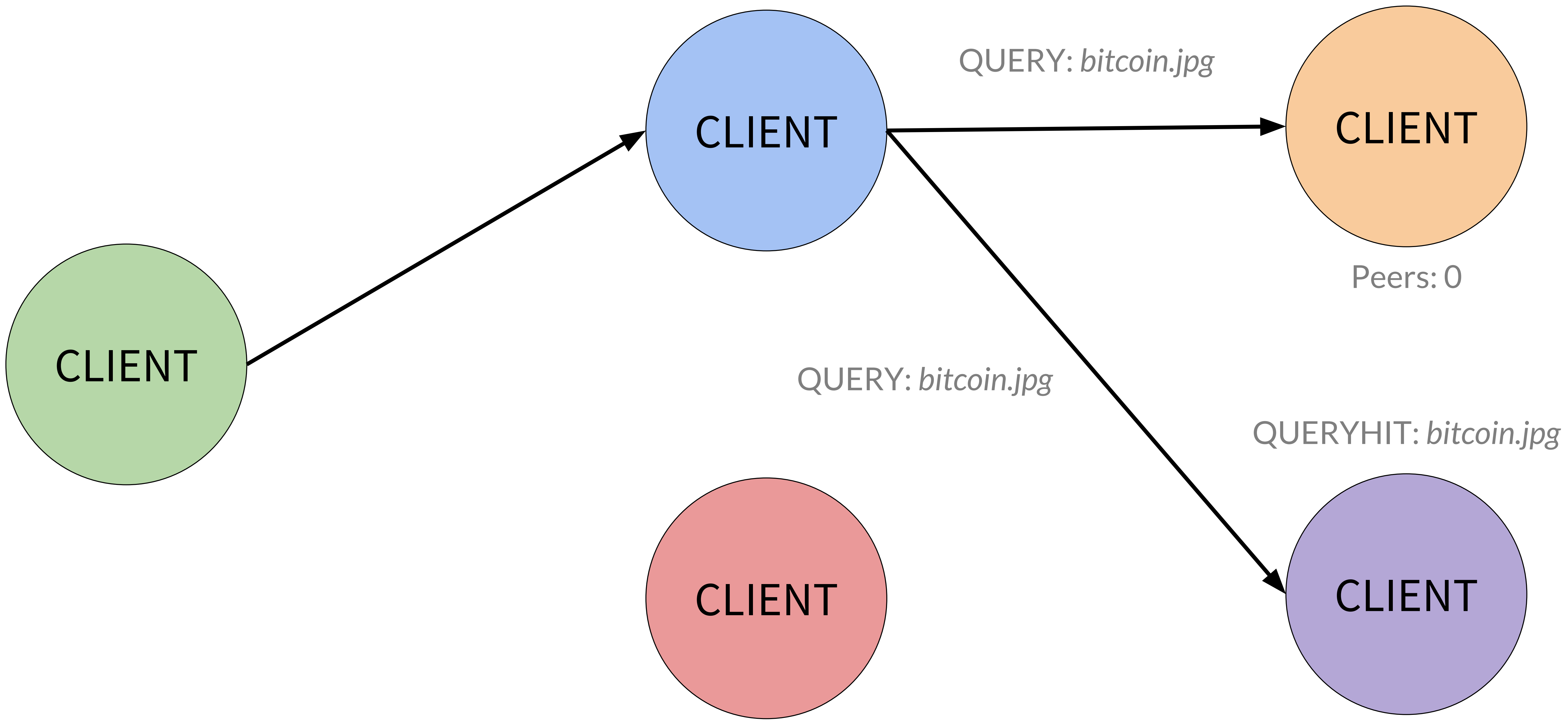
REFRESHER

ALTERNATIVE P2P EXAMPLE



REFRESHER

GNUTELLA EXAMPLE

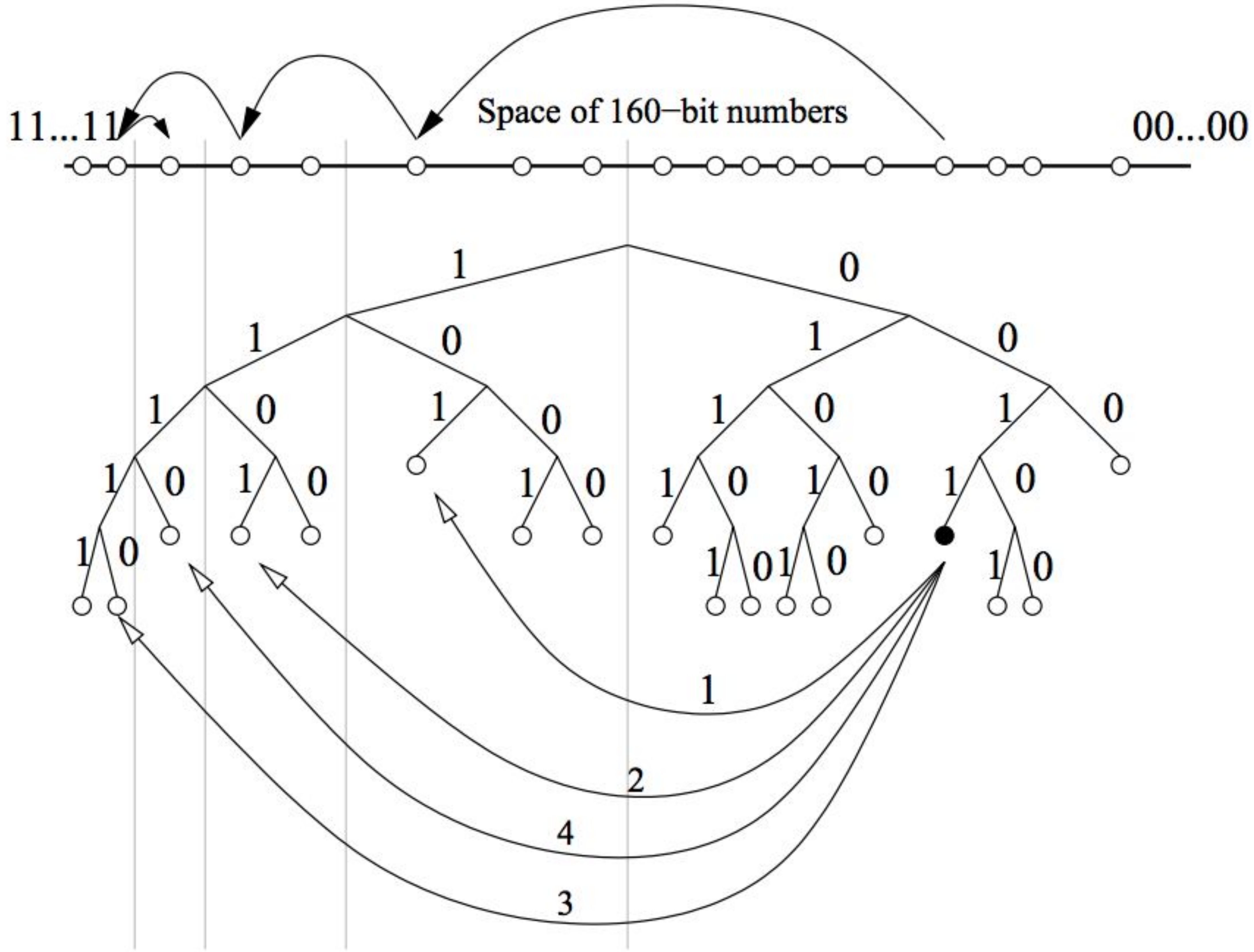


2 THE (NEAR) FUTURE

KADEMLIA DHT

WHAT ETHEREUM’S P2P INFO SYSTEM IS BASED ON

- **How do we locate other nodes by ID?**
  - Node with prefix 0011 finds node with prefix 1110 by successively learning of and querying closer and closer nodes
- Nodes have 160 bits IDs, and the keys for the KV pairs are also 160-bit IDs.
  - Because Kademlia relies on distance between two IDs, the distance is defined as: $d(x, y) = x \oplus y$ **(XOR)**



KADEMLIA DHT

WHAT ETHEREUM’S P2P INFO SYSTEM IS BASED ON

- **Why does XORing key and node id work?**
- In a fully populated binary tree of 160 bit IDs, the magnitude of distance between IDs is the height of the smallest subtree containing them both
- When a tree is not fully populated, the closest leaf to and ID **x** is the leaf who’s ID shares the **longest common prefix** (LCP) of **x**
  - If there are empty branches, there might be more than one leaf with the LCP
  - In that case, the closest leaf to **x** will be the closest leaf to ID **x** produced by flipping the bits in **x** corresponding to the empty branches of the tree

KADEMLIA DHT

HOW ABOUT ROUTING TO OTHER NODES

- Each node keeps k-buckets in its routing table to store the peer node information (16 in Ethereum)
- Learn more about the routing tables here and how k-buckets works (maybe for lab!)

RLP ENCODING

HOW TO SERIALIZE DATA

- The purpose of **RLP (Recursive Length Prefix)** is to encode arbitrarily nested arrays of binary data, and RLP is the main encoding method used to serialize objects in Ethereum
- The only purpose of RLP is to encode structure; encoding specific data types (eg. strings, floats) is left up to higher-order protocol - there's also a way to decode data to use it
- Starts off with **byte value (0x80) + the length of the string**, and sometimes a **byte value (0xc0) + length of the list**
- Examples:
  - The string "dog" = [0x83, 'd', 'o', 'g']
  - The list ["cat", "dog"] = [0xc8, 0x83, 'c', 'a', 't', 0x83, 'd', 'o', 'g']

Worth reading the spec: https://github.com/ethereum/wiki/wiki/RLP

TRANSPORT PROTOCOL

HOW DOES COMMUNICATION BETWEEN NODES WORK

- When **ĐΞVp2p nodes communicate, they use TCP via the Internet**
- But on top of that are the messages that are defined by **RLPx**, allowing them communicate the sending and receiving of packets
- Packets are dynamically framed, prefixed with an **RLP encoded** header, encrypted and authenticated
- **Multiplexing** is achieved via the **frame header** which specifies the **destination protocol** of a packet
  - Think about how packets might be received asynchronously (benefits of UDP's asynchronous transfer with TCP's reliability)

TRANSPORT PROTOCOL

WHAT IS MULTIPLEXING?

- In **statistical multiplexing**, a communication channel is divided into an arbitrary number of variable bitrate digital channels or **data streams**
- Each stream is divided into packets that normally are delivered asynchronously in a first-come first-served fashion
    - In alternative fashion, the packets may be delivered according to some scheduling discipline for fair queuing or differentiated and/or guaranteed quality of service
- Normally implies "on-demand" service rather than one that preallocates resources for each data stream

TRANSPORT PROTOCOL

ENCRYPTED HANDSHAKE

- Connections are established via cryptographic handshake, and once established, packets are encrypted and encapsulated as frames
- **Phase 1:** Peer authentication with an encryption handshake
  - **Peer authentication**: to establish a secure communication channel by setting up an encrypted, authenticated message stream
  - **Encryption handshake**: to exchange temporary keys to set initial values for this secure session
- **Phase 2:** The base protocol handshake
  - Negotiate supported protocols, checking versions and network IDs

TRANSPORT PROTOCOL

SECURE CONNECTIONS

Side Note: If the connection was initiated by a peer, we say that they are the **initiator**, and the other peer is **receiver**. The word **remote** is used to describe the 'other' peer in a connection when talking from a point of view of a node.

Creating a secure connection consists of the following steps:

1. **Initiator** sends an authentication message to **receiver**
2. **Receiver** responds with an authentication response message and sets up a secure session
3. **Initiator** checks receiver's response and establishes a secure session
4. **Receiver** and **Initiator** then send base protocol handshake on the established secure channel

Either side may disconnect if authentication fails or if the protocol handshake isn’t appropriate.

TRANSPORT PROTOCOL

WHAT IF FAILURE?

Side Note: The other distinction is whether the remote peer is **known** or **new**. A **known** peer is one which has previously been connected and for which a corresponding session token is remembered.

- If the handshake fails, if and only if initiating a connection TO a known peer, then the nodes information should be removed from the node table and the connection MUST NOT be reattempted.
- Due to the limited IPv4 space and common ISP practices, this is likely a common and normal occurrence, therefore, no other action should occur.
- If a handshake fails for a connection which is received, no action pertaining to the node table should occur.

TRANSPORT PROTOCOL

WHAT IF SUCCESS?

Side Note: The other distinction is whether the remote peer is **known** or **new**. A **known** peer is one which has previously been connected and for which a corresponding session token is remembered.

- If the handshakes succeed, the fixed array of protocols supported by both peers will run on the connection parallelly to send and receive messages
- Once established, packets are encapsulated as frames which are encrypted using **AES-256 in CTR mode**
  - Initial values for the message authentication and cipher are never the same
    - Key material for the session is derived via a KDF (key derivation function) and ECDHE-derived (Elliptic-curve Diffie–Hellman) shared-secret
    - ECC uses secp256k1 curve (ECP).
- It is the purpose of the encryption handshake to negotiate these key values for a new secure session

Read more here: https://github.com/ethereumproject/go-ethereum/wiki/RLPx-Encryption

TRANSPORT PROTOCOL

WHAT HAPPENS WITH TRANSACTIONS

- **ĐΞVp2p** has a capabilities list as we saw in the earlier diagrams.
- With that two peers can decide to communicate with the ETH protocol capability
- The ETH protocol describes the forwarding of blocks and transactions
- If a user creates a transaction (that interacts with a smart contract) the transaction will be forwarded to all nodes/miners
- The miners will include it in a valid block and the nodes will then verify the block (including the execution of transactions) forward that block to every other node and add it to the chain

PROBLEMS

WHY DOES DEVP2P KINDA SUCK

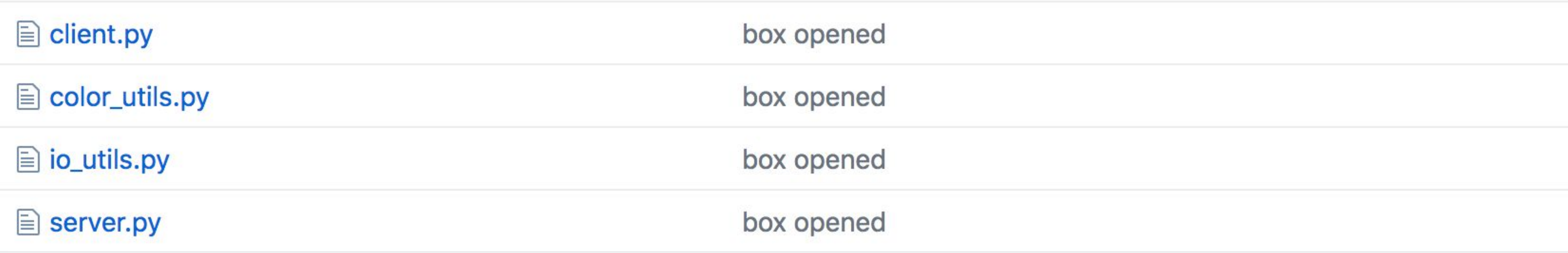
- So many roundtrips just to figure out if someone is on the right blockchain
- Upgrades need tight coordination - the whole system is frozen - everything needs to be backward compatible
- Achieving upgrades have been tied to mainet hard forks
- Everyone has to upgrade their own nodes and only speak their own protocol
- We can’t make changes on an accelerated schedule
  - Heavily tied to RLPx protocol, secp256k1, keccak256

3 ASSIGNMENT

THE ASSIGNMENT

P2P NETWORKING MODELS

https://github.com/Blockchain-for-Developers/sp18-lab06



client.py	box opened
color_utils.py	box opened
io_utils.py	box opened
server.py	box opened

SEE YOU NEXT TIME

BLOCKCHAIN
AT BERKELEY