ASURITE ID: vyaganti
ASU ID: 1223441505

SOCKET PROJECT MILESTONE
COMPUTER NETWORKS – CSE434

OVERVIEW OF METHODS:

DatagramPacket and **DatagramSocket** are the two main classes that are used to implement a UDP client/server application. DatagramPacket is a data container and DatagramSocket is a mechanism to send and receive DatagramPackets.

In UDP's terms, data transferred is encapsulated in a unit called datagram. A datagram is an independent, self-contained message sent over the network whose arrival, arrival time, and content are not guaranteed. And in Java, DatagramPacket represents a datagram.
You can create a DatagramPacket object by using one of the following constructors:
DatagramPacket(byte[] buf, int length), DatagramPacket(byte[] buf, int length, InetAddress address, int port). As you can see, the data must be in the form of an array of bytes. The first constructor is used to create a DatagramPacket to be received. The second constructor creates a DatagramPacket to be sent, so you need to specify the address and port number of the destination host. The parameter length specifies the amount of data in the byte array to be used, usually is the length of the array (buf.length). There are also other constructors that allow you to specify the offset in the byte array, as well as using a SocketAddress: DatagramPacket(byte[] buf, int offset, int length), DatagramPacket(byte[] buf, int offset, int length, SocketAddress address). In addition, the DatagramPacket provides setter and getter methods for address, data and port number.

METHODS DESCRIPTION:

1. public DatagramPacket(byte[] buf, int length)

   Description:
   Constructs a DatagramPacket for receiving packets of length length.
   The length argument must be less than or equal to buf.length.
   Parameters:
   buf - buffer for holding the incoming datagram.
   length - the number of bytes to read.

2. public DatagramPacket(byte[] buf, int offset, int length, InetAddress address, int port)

   Description:
   Constructs a datagram packet for sending packets of length length to the specified port number on the specified host. The length argument must be less than or equal to buf.length.
   Parameters:
   buf - the packet data.
   length - the packet data length.
   address - the destination address.
   port - the destination port number.

3. public InetAddress getAddress()

   Description:
   Returns the IP address of the machine to which this datagram is being sent or from which the datagram was received.
   Returns:
   the IP address of the machine to which this datagram is being sent or from which the datagram was received.

4. Byte[] getBytes()

   Description:
   The getBytes() method encodes a given String into a sequence of bytes and returns an array of bytes.
   Returns:
   It encodes the String using default charset method.

5. Send ()

   Description:
   Sends the DatagramPacket object from one end to another end.
   Parameters:
   DatagramPacket object which can only be accepted in another end.

6. Receive():

   Description:
   Receives a DatagramPacket pbject from another end and processes it.
   Parameters:
   DatagramPacket object which can only be accepted in another end.

7. printStackTrace()

   Description:
   Used to handle exceptions and errors. If one out of five methods in your code cause an exception, printStackTrace() will pinpoint the exact line in which the method raised the exception.

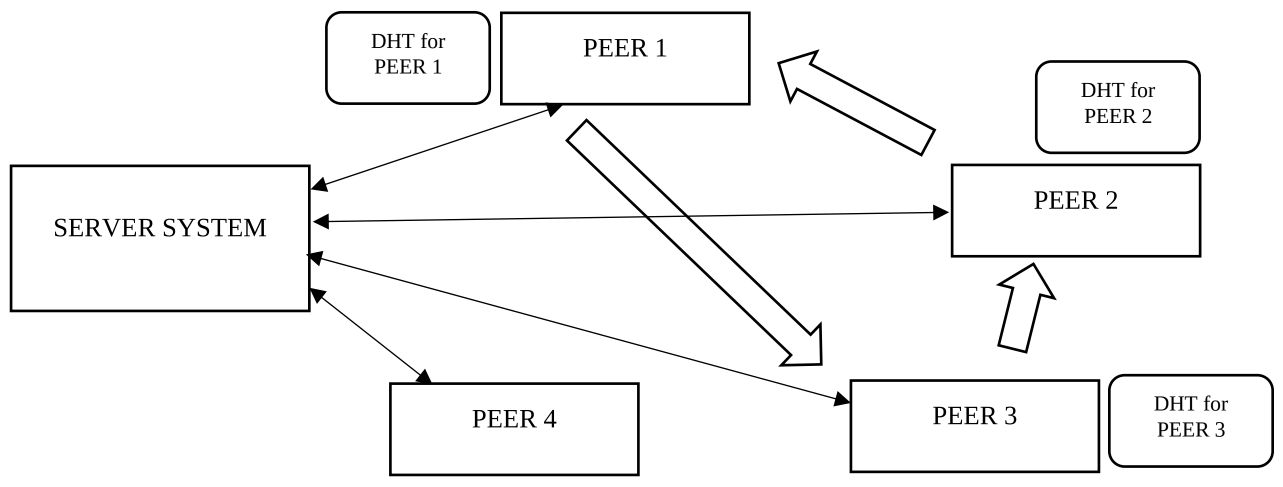
DATASTRUCTURES:

Hashmaps in Java can be a good choice for storing the required values of users and their details. Hashmap<String, String[]> can be used here. In the String we store the <user-name> and in the String[] we can store <IPv4 Address, $port_{left}$, $port_{right}$, $port_{query}$>. When the client sends the username to server we can extract the <username> and compare whether it is in the hashmap(if statically data is given, or else add the <username> to a hashmap every time when client sends). If the <username> is in the hashmap, compare the IPv4 address from the string[] and DatagramPacket object received from client. If matches, send SUCCESS as response to client. If NOT, send FAILURE.

CONSIDERATIONS:
A hashmmap with users data in the server which when used client requests for communication. At client(P2P) side we use different methods for computing different operations like register, deregister, setup-dht, query-dht, dht-complete. Every single method gets invoked based on the command line arguments. We will split the command line argument and compute related method.

ASURITE ID: vyaganti
ASU ID: 1223441505

Application Overview:



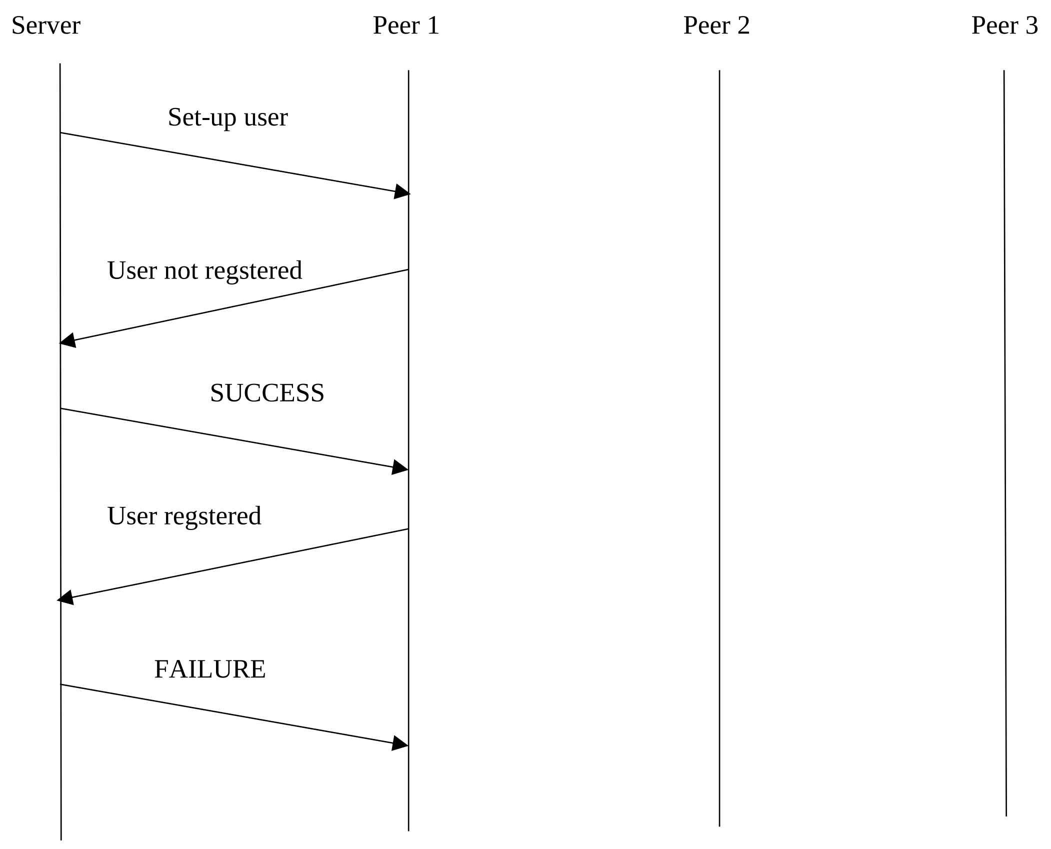
NOTE:

⇨ Denotes the network

↔ Denotes the transmission link

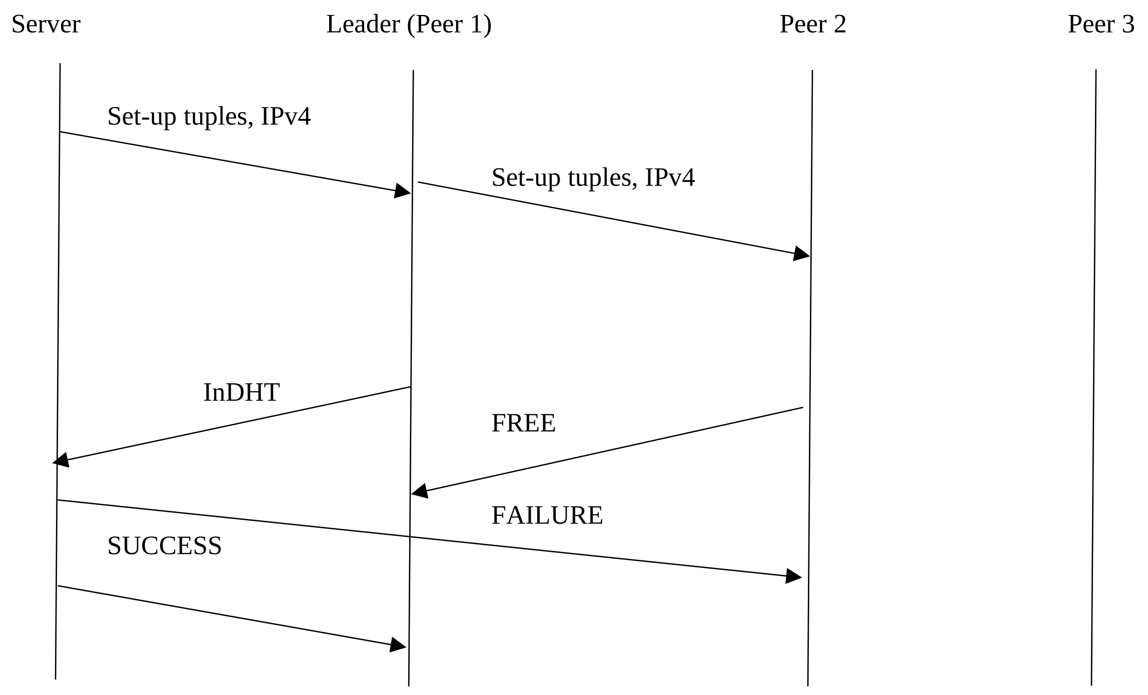
TIME SPACES DIAGRAMS:

1. setup-dht.

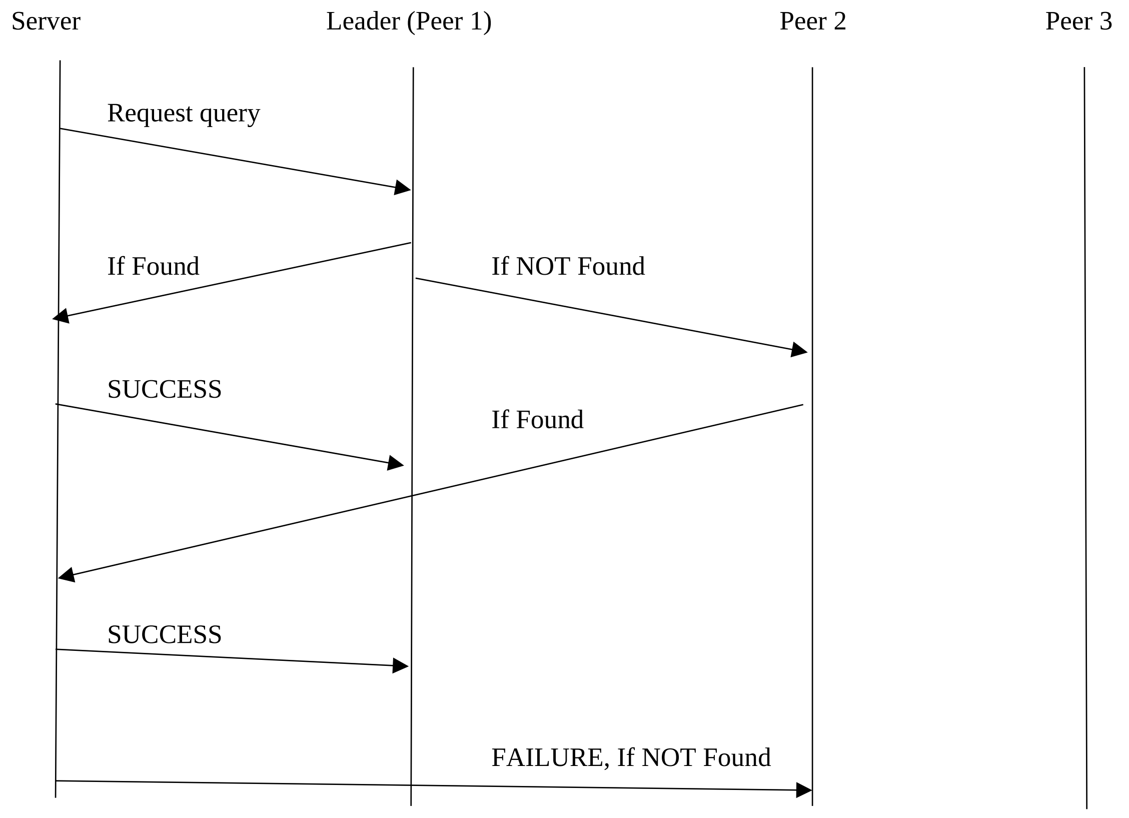


ASURITE ID: vyaganti
ASU ID: 1223441505

2. dht-complete.

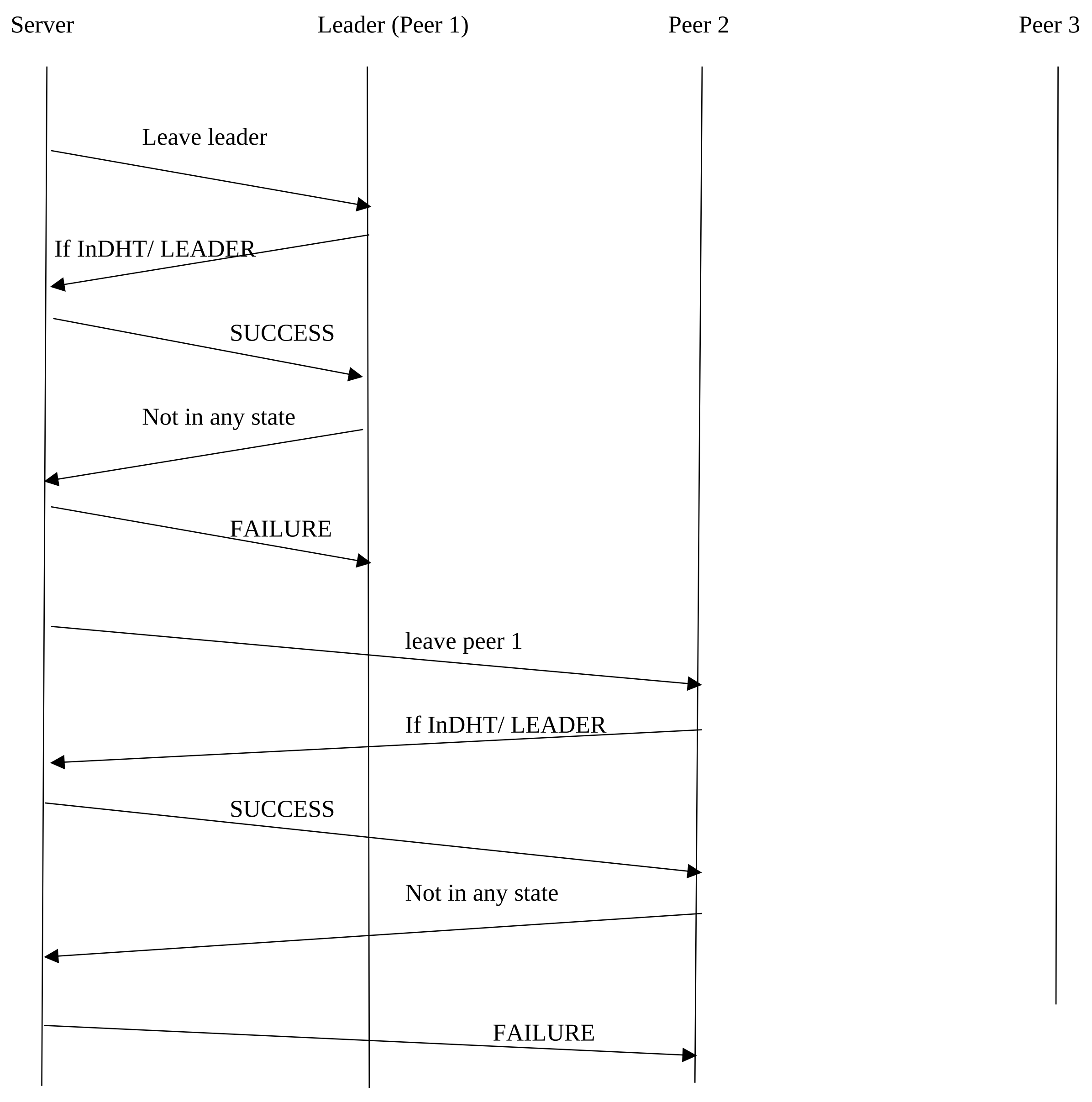


3. query-dht.

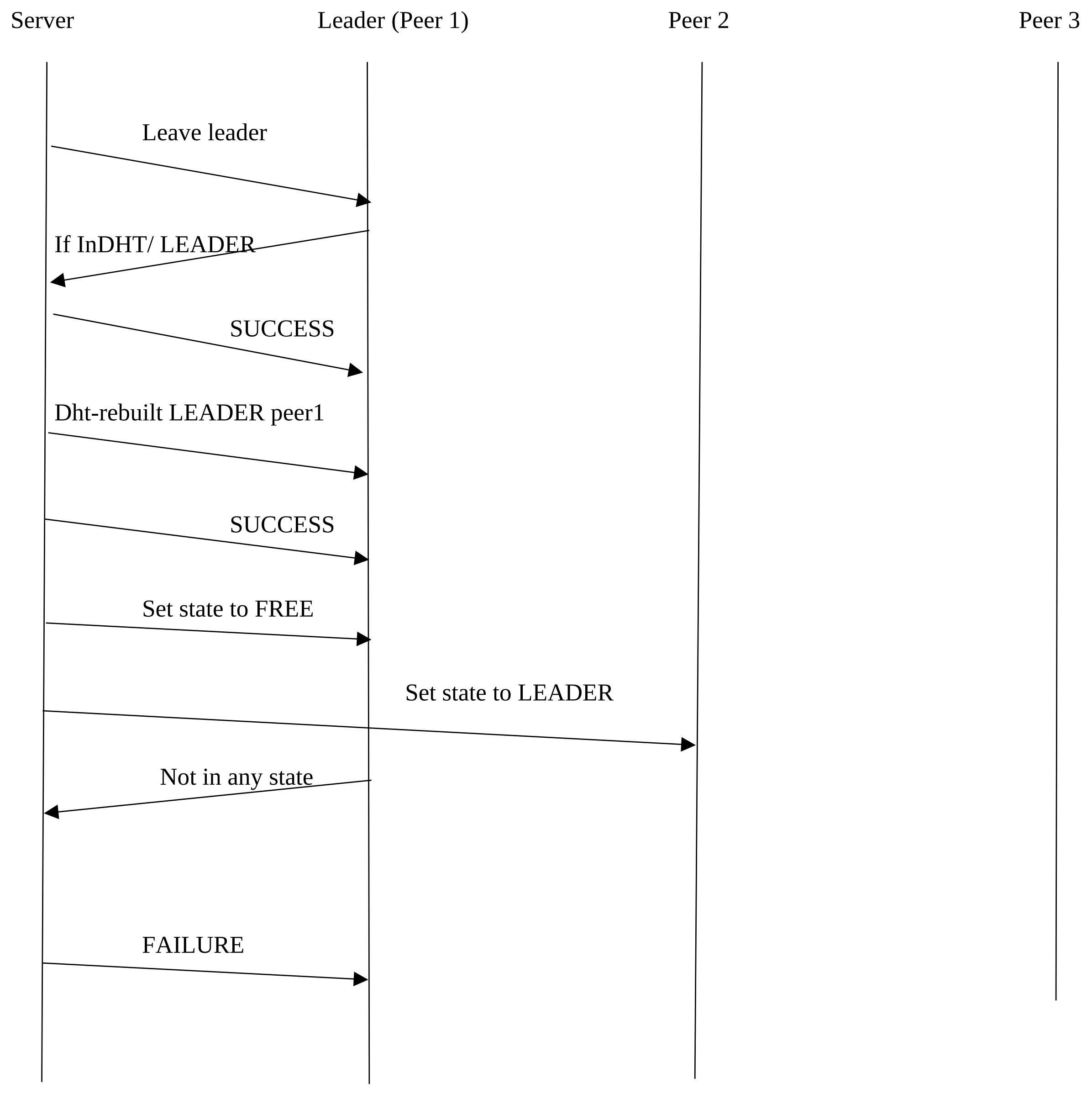


ASURITE ID: vyaganti
ASU ID: 1223441505

4. leave-dht.

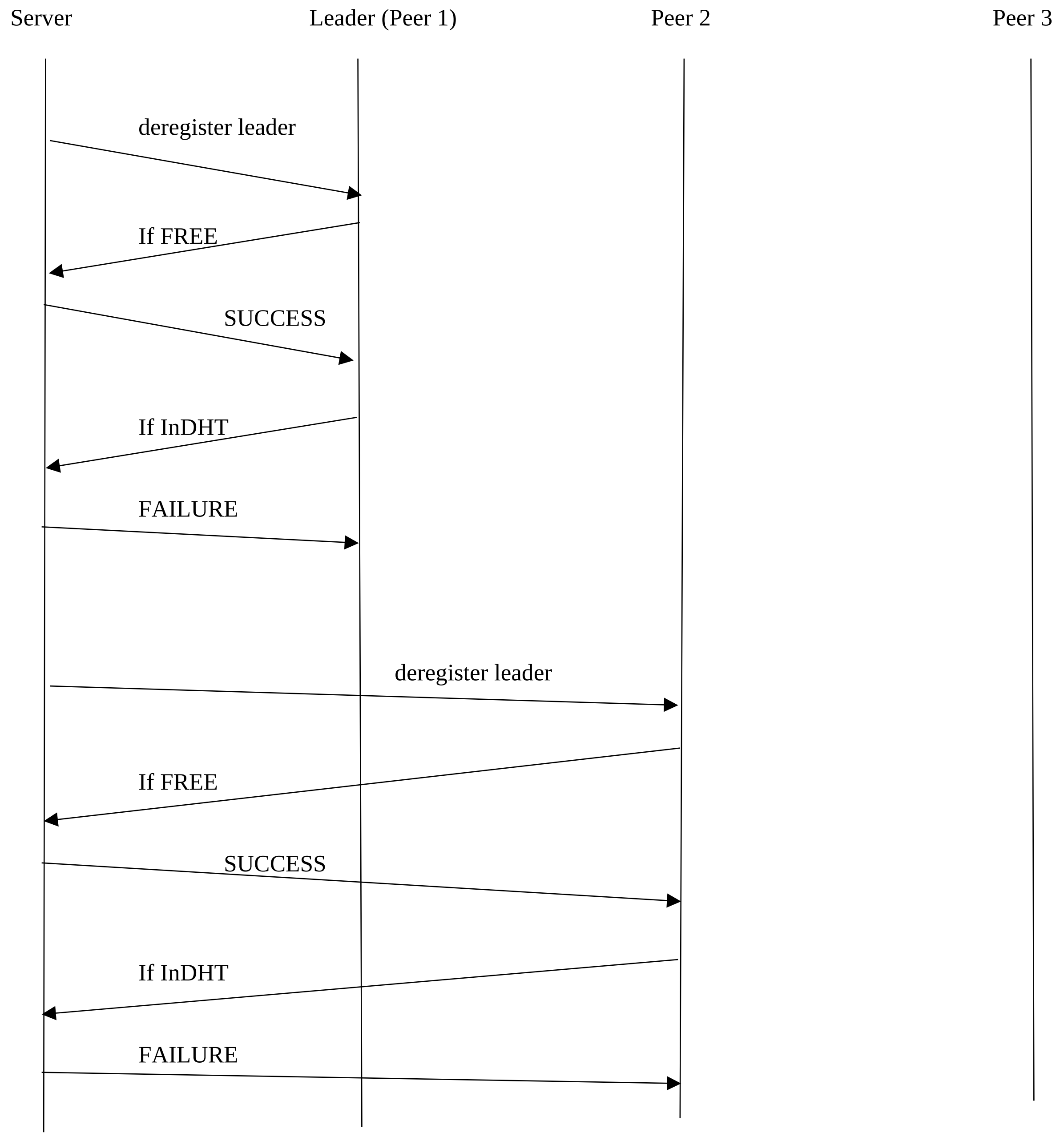


5. dht-rebuilt.



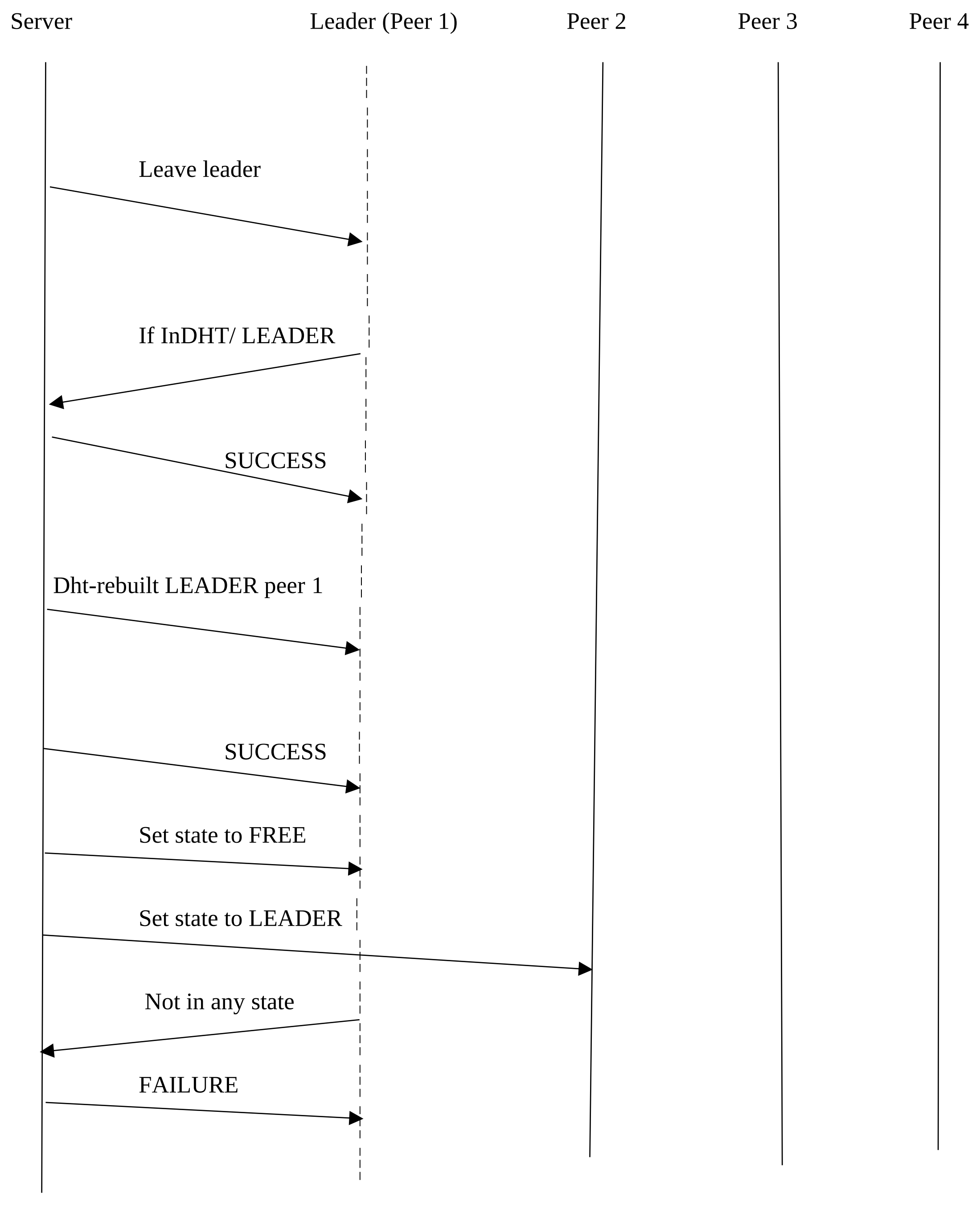
ASURITE ID: vyaganti
ASU ID: 1223441505

6. deregister.



ASURITE ID: vyaganti
ASU ID: 1223441505

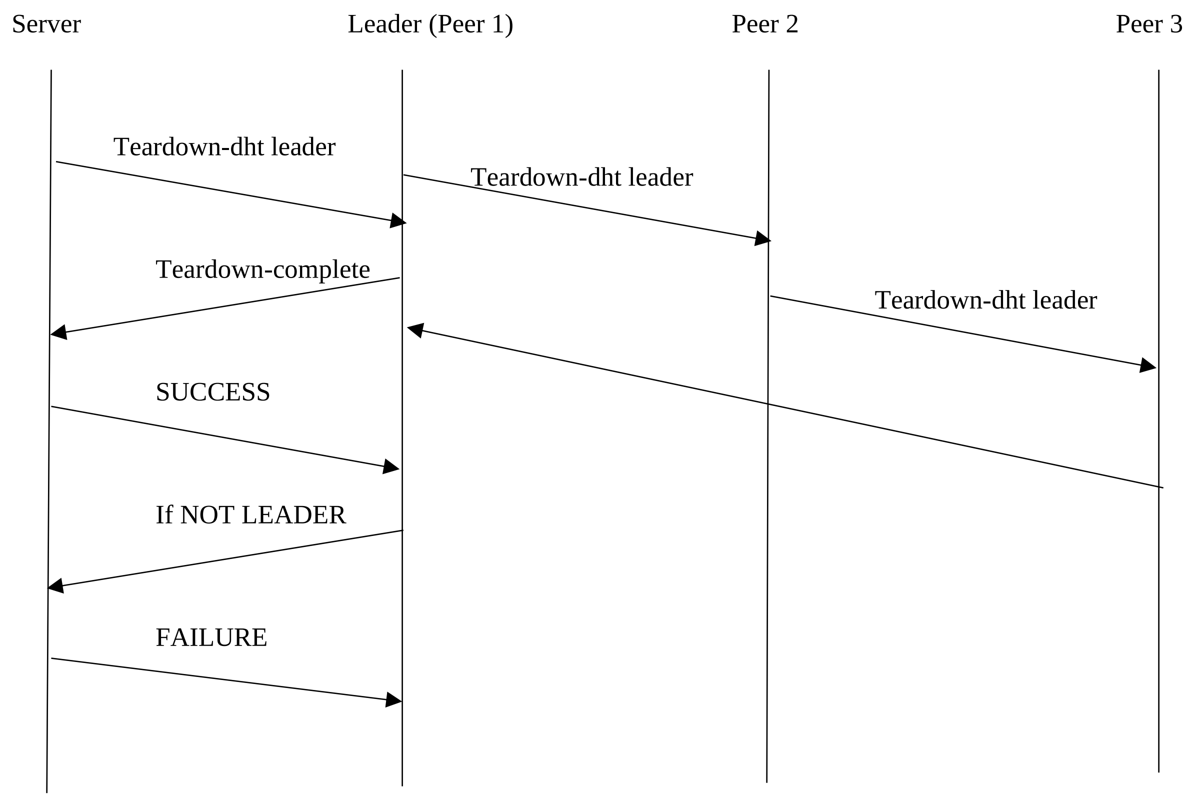
7. join-dht.



NOTE:
┆ Denotes that the peer is Terminated.

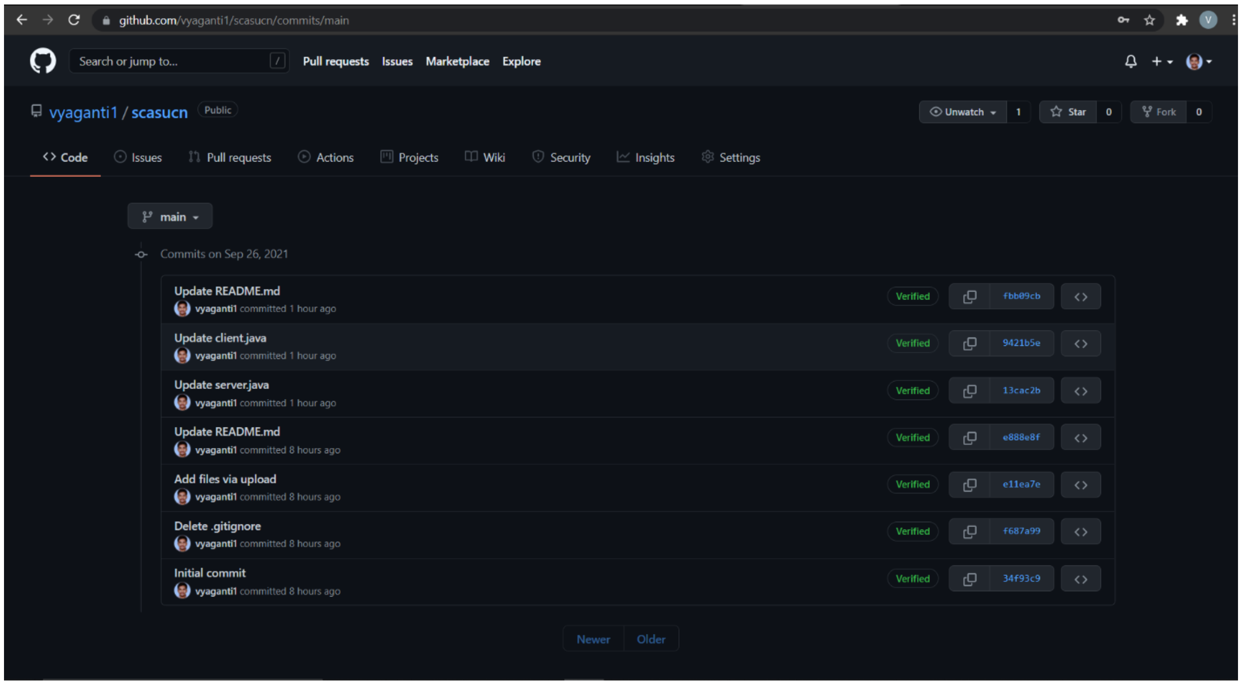
│ Denotes that the peer is in DHT.

8. Teardown-dht.



ASURITE ID: vyaganti
ASU ID: 1223441505

MYCOMMITS:



github.com/vyaganti1/scasucn/commits/main

vyaganti1 / scasucn Public

Commits on Sep 26, 2021

Commit	Author	Status	SHA
Update README.md	vyaganti1 committed 1 hour ago	Verified	fbb09cb
Update client.java	vyaganti1 committed 1 hour ago	Verified	9421b5e
Update server.java	vyaganti1 committed 1 hour ago	Verified	13cac2b
Update README.md	vyaganti1 committed 8 hours ago	Verified	e888e8f
Add files via upload	vyaganti1 committed 8 hours ago	Verified	e11ea7e
Delete .gitignore	vyaganti1 committed 8 hours ago	Verified	f687a99
Initial commit	vyaganti1 committed 8 hours ago	Verified	34f93c9

LINK FOR VIDEO DEMO:

https://youtu.be/tJJL3kwVQ7Q

TimeStamps:

0:00 Introduction
0:36 Server Class Description
2:17 Client Class Description
3:29 Output Execution