IP Assignment

Code Description:

The overall program design of the Peer-to-Peer system can be described as follows:

1. Initialization: The program starts by accepting command-line arguments, including the port number, peer ports, filenames, and query file. It generates a random ID for the current node and creates a Node object with the provided information.
2. Node Creation: The Node object represents the current node in the system. It contains the ID, port, peer ports, filenames, query file, and other necessary data structures. It also creates instances of ConnectionHandler, PrintWriter, and Socket objects for communication with other nodes.
3. Connection Handling: The Node object creates a server socket to listen for incoming connections from other nodes. It also establishes client sockets to connect to the peers specified in the command-line arguments. ConnectionHandler threads are created for each incoming connection, which handle the communication with the connected peers.
4. Message Exchange: The ConnectionHandler threads continuously listen for incoming messages from the connected peers. They process the received messages based on their content and take appropriate actions. The messages can include ID exchange, leader election, file sharing, file queries, and other coordination messages.
5. Leader Election: The nodes participate in a leader election process to determine the leader node among the connected peers. The leader is chosen based on a criterion, such as having the smallest ID among the connected peers. The leader coordinates file exchanges and queries in the system.
6. File Sharing: Each node shares its filenames with the leader. The leader maintains a mapping of node ports to their associated filenames. This allows other nodes to query the leader for specific files.
7. File Queries: Nodes can query the leader for a file by sending a file query message. The leader checks its own filenames and the filenames of other connected nodes to determine the availability of the requested file. It then sends

a response message to the querying node with the file's location or a "NOT FOUND" message.

Design Tradeoffs:

- Decentralization vs. Centralization: The system follows a decentralized architecture where nodes communicate with each other in a peer-to-peer manner. This allows for better scalability, fault tolerance, and load balancing. However, it introduces complexities in leader election and coordination among nodes.
- Leader Election Criterion: The current design chooses the leader based on having the smallest ID among the connected peers. This criterion may favor some nodes over others and may not be suitable for all scenarios. Alternative leader election algorithms could be explored based on specific requirements and system dynamics.
- Message Passing: The communication between nodes is based on TCP/IP sockets and message passing. While this provides reliable and ordered message delivery, it also introduces overhead due to the message format, parsing, and network latency. Other communication mechanisms, such as message queues or protocols, could be considered based on the system's needs.
- Scalability vs. Resource Utilization: The program design assumes a fixed number of peer ports in the command-line arguments. This limits the scalability of the system as it can only accommodate a predefined number of peers. If scalability is a major concern, a dynamic peer discovery mechanism or a decentralized routing protocol could be considered to handle a larger number of nodes.

These design tradeoffs were considered while developing the system to balance factors such as simplicity, performance, scalability, and fault tolerance based on the requirements and constraints of the peer-to-peer network.

REMARKS: The File Transfer between peers are not implemented in this program.