Design Document of Gnutella P2P File Sharing Systems

1. Overview

Introduction

Distributed systems enable multiple computers to work together seamlessly, offering scalability, fault tolerance, and resource sharing. Peer-to-Peer (P2P) networks, such as Gnutella, exemplify decentralized architectures where each peer acts both as a client and a server. Maintaining file consistency in such environments is challenging, especially when multiple peers can modify or cache files concurrently.

This document outlines the design of a hierarchical Gnutella-style P2P system developed for CS485 Programming Assignment 3 (PA3). The system employs both push-based and pull-based mechanisms to ensure strong file consistency across the network. Utilizing Java's Remote Method Invocation (RMI) framework, the system establishes communication channels between SuperPeers and LeafNodes, facilitating efficient file search, download, and synchronization.

2. System Overview

The designed P2P system comprises two primary roles:

1. **SuperPeers:**
   Act as intermediaries between LeafNodes, managing file indexing, handling queries, and propagating consistency mechanisms (push or pull).
2. **LeafNodes:**
   Represent individual peers in the network, responsible for storing master and cached copies of files, handling user interactions, and maintaining consistency based on directives from SuperPeers.

The system supports two consistency mechanisms:

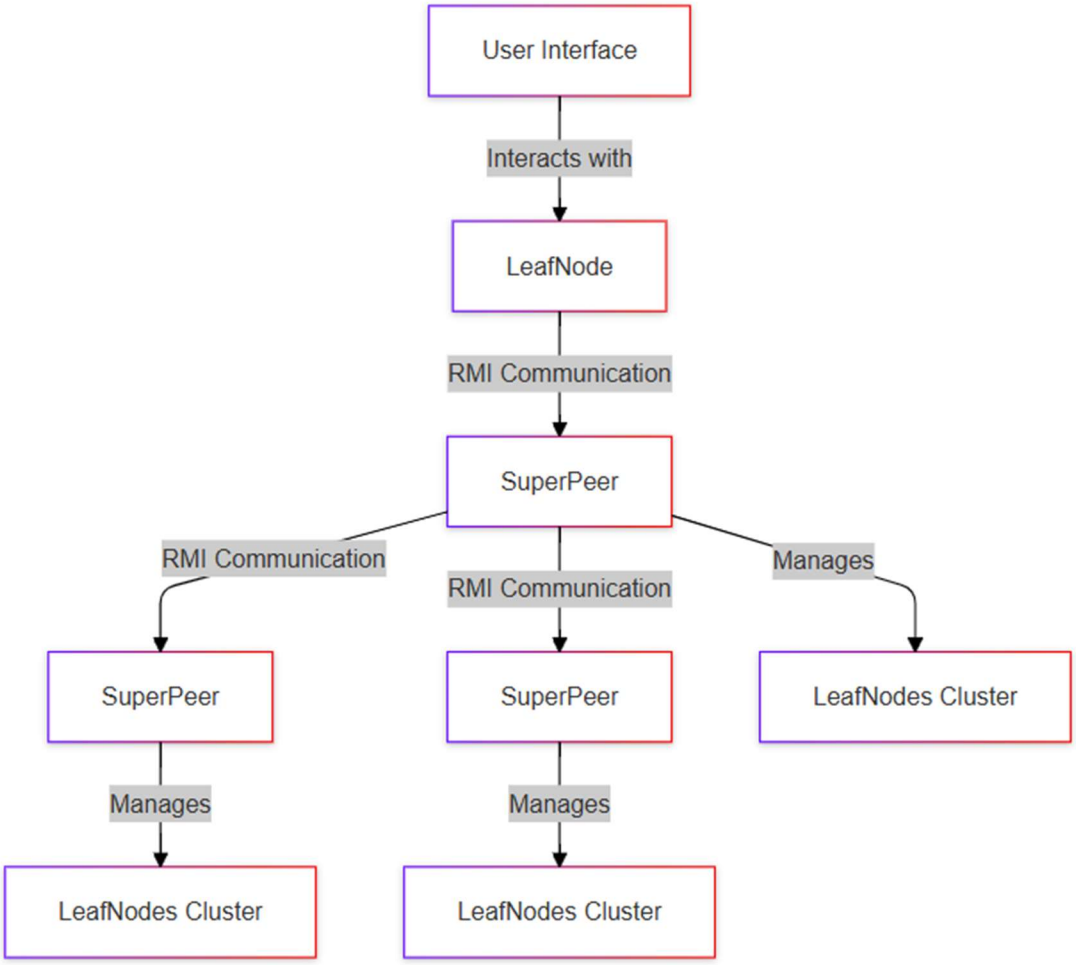
- **Push-Based Consistency:**
  Upon modifying a master copy, the origin SuperPeer proactively broadcasts invalidation messages to all relevant LeafNodes to invalidate their cached copies.
- **Pull-Based Consistency:**
  LeafNodes periodically poll their origin SuperPeers based on a Time-To-Refresh (TTR) value to verify the validity of their cached copies.

Additionally, the network topology can be configured as either **All-to-All** or **Linear**, influencing how SuperPeers communicate and propagate messages.

3. Architecture

3.1 High-Level Architecture

The system employs a hierarchical P2P architecture with SuperPeers at the upper tier and LeafNodes at the lower tier. SuperPeers manage clusters of LeafNodes, facilitating efficient file searches and consistency maintenance across the network.



4. Components Detail

4.1 SuperPeer

Responsibilities:

- **File Indexing:**
  Maintains an index of all files hosted by its associated LeafNodes, including version numbers and statuses.

- **Query Handling:**
  Processes search queries from LeafNodes, returns relevant file information, and coordinates with other SuperPeers based on the network topology.
- **Consistency Management:**
  Implements push-based and pull-based mechanisms to ensure file consistency across LeafNodes. Broadcasts invalidation messages in push-based consistency and responds to poll requests in pull-based consistency.

Key Classes:

- SuperPeer.java:
  Initializes the SuperPeer, sets up the RMI registry, and binds the SuperPeerImpl object.
- SuperPeerImpl.java:
  Implements SuperPeerInterface, providing methods for file registration, searching, querying, broadcasting, and polling.
- SuperPeerInterface.java:
  Defines the remote methods that SuperPeers must implement for RMI communication.

4.2 LeafNode

Responsibilities:

- **File Storage:**
  Manages both master copies (owned files) and cached copies (downloaded from other peers) in designated directories.
- **User Interaction:**
  Provides a console-based interface for users to search, download, edit, and delete files.
- **Consistency Maintenance:**
  Executes push-based invalidations received from SuperPeers and performs periodic polls in pull-based consistency to verify cached file validity.

Key Classes:

- MultiClient.java:
  Acts as the entry point for setting up a LeafNode client. Initializes the LeafNode by reading configurations, setting up RMI registries, and binding the AvgRespFileSearch implementation.

- AvgRespFileSearch.java:
  Implements LeafNodeInterface, providing methods for file download, query handling, invalidation processing, and polling.
- LeafNodeInterface.java:
  Defines the remote methods that LeafNodes must implement for RMI communication.

4.3 Configuration Management

Components:

- **Configuration Files:**

  consistency_config.txt:
  Specifies the consistency mechanism (PUSH or PULL) to be used.

  peerconfig.txt:
  Details of all LeafNodes including their IDs, ports, directories, and associated SuperPeers.

  superpeerconfig.txt:
  Details of all SuperPeers including their IDs, ports, and associated LeafNodes.

  topology.txt:
  Defines the network topology (all for All-to-All or linear for Linear Topology).

  topology_config.txt:
  Specifics of the network topology, listing neighbors for each SuperPeer.

- **SetupConfig.java:**
  Parses the above configuration files, loading details into respective data structures (GetPeerDetails, GetSuperPeerDetails, GetTopologyDetails). This class ensures that all components are initialized with the correct configurations upon startup.

4.4 Communication Mechanism

Java RMI (Remote Method Invocation):

- **RMI Framework:**
  Facilitates communication between SuperPeers and LeafNodes by allowing them to invoke methods on remote objects as if they were local.
- **Registry Setup:**
  Each SuperPeer and LeafNode sets up an RMI registry on its designated port and binds

its implementation object (SuperPeerImpl or AvgRespFileSearch) with a unique identifier.

- **Remote Interfaces:**
  Both SuperPeerInterface and LeafNodeInterface extend java.rmi.Remote and declare remote methods that can be invoked by other peers.

Advantages of Using RMI:

- **Simplicity:**
  Abstracts the complexities of network communication, allowing developers to focus on implementing business logic.
- **Object-Oriented:**
  Supports invoking methods on objects residing on different JVMs, aligning with the object-oriented nature of Java.
- **Built-in Support:**
  Java provides robust support for RMI, including security, serialization, and dynamic class loading.

---

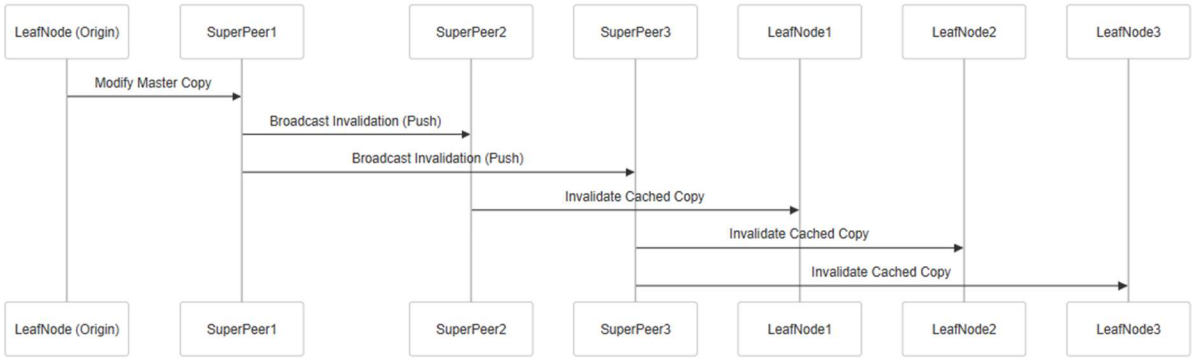
5. Consistency Mechanisms

Ensuring file consistency across a distributed P2P network is critical, especially when multiple peers can access and modify files simultaneously. This system employs two primary consistency mechanisms: **Push-Based Consistency** and **Pull-Based Consistency**.

5.1 Push-Based Consistency

Mechanism:

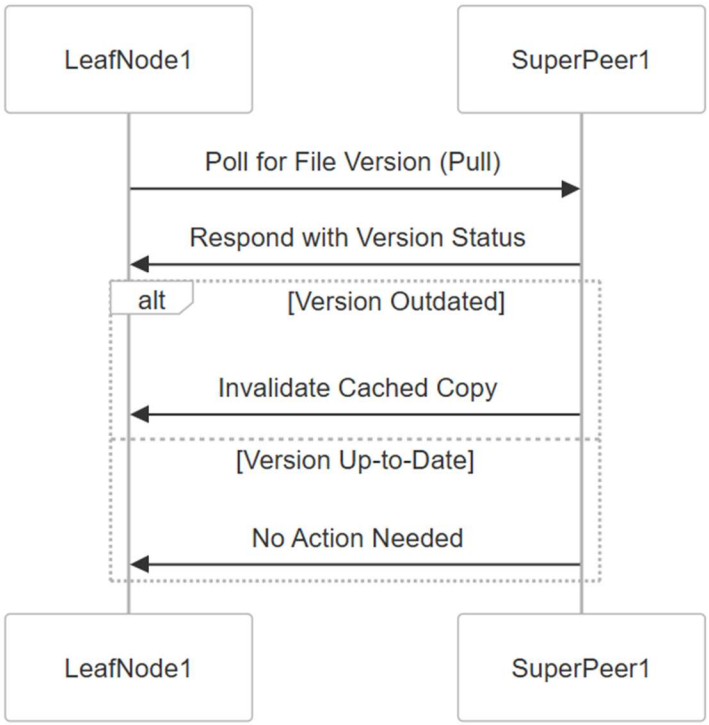
- When a master copy of a file is modified by a LeafNode, the origin SuperPeer proactively broadcasts an invalidation message to all SuperPeers in the network.
- Each SuperPeer, upon receiving the invalidation message, propagates it to their associated LeafNodes.
- LeafNodes receiving the invalidation message mark their cached copies of the file as invalid.



5.2 Pull-Based Consistency

Mechanism:

- LeafNodes periodically poll their origin SuperPeers based on a Time-To-Refresh (TTR) value to verify the validity of their cached copies.
- Upon receiving a poll request, the SuperPeer checks the current version number of the master copy.
- If the cached copy's version is outdated, the SuperPeer instructs the LeafNode to invalidate or update its cached copy.



6. Design Considerations

6.1 Choice of RMI

Rationale:

- **Language Consistency:**
  The system is entirely implemented in Java, making RMI a natural choice for facilitating remote communications without interoperability issues.
- **Ease of Use:**
  RMI simplifies the process of invoking methods across different JVMs, reducing the boilerplate code required for socket programming.
- **Feature-Rich:**
  RMI provides built-in support for features like object serialization, dynamic class loading, and security, which are beneficial for distributed applications.

Alternatives Considered:

- **Socket Programming:**
  Offers more control over network communications but requires handling low-level details such as connection management, data serialization, and protocol design.
- **Web Services (e.g., RESTful APIs):**
  Suitable for heterogeneous environments but introduces additional complexity in defining and parsing request/response formats.

Trade-offs:

- **Performance:**
  RMI may introduce overhead due to serialization and method invocation processes. However, for the scope of this assignment, this overhead is negligible.
- **Scalability:**
  RMI is suitable for moderate-sized networks but may face challenges in extremely large or dynamic environments compared to more scalable communication frameworks.

6.2 Push vs. Pull Consistency Mechanisms

Push-Based Consistency:

- **Advantages:**

Strong Consistency:
Ensures that all cached copies are invalidated promptly, maintaining high consistency.

Simplicity:
The broadcasting mechanism is straightforward to implement using RMI's remote method invocation.

- **Disadvantages:**

Network Overhead:
Broadcasting invalidation messages to all peers consumes significant bandwidth, especially in large networks.

Scalability Issues:
As the number of peers increases, the volume of invalidation messages can lead to congestion and reduced performance.

Pull-Based Consistency:

- **Advantages:**

Reduced Network Traffic:
Invalidation messages are only sent when LeafNodes poll, potentially reducing unnecessary traffic.

Scalability:
More scalable for larger networks as the load is distributed over time.

- **Disadvantages:**

Stale Data Window:
There's a window between data modification and the next poll during which LeafNodes may serve stale data.

Latency in Consistency:
Updates are not instantaneous, leading to possible delays in reflecting the latest file versions.

Design Decision:

- **Hybrid Approach (Graduate Extension):**
  Implement both push and pull mechanisms with configurable parameters, allowing the system to toggle between them based on specific requirements or network conditions.

6.3 Network Topology

Options:

1. **All-to-All Topology:**
    - **Description:**
      Every SuperPeer is connected to every other SuperPeer, enabling direct communication and broadcasting.
    - **Advantages:**
        - **Robustness:**
          Multiple communication paths reduce the risk of message loss.
        - **Low Latency:**
          Direct connections facilitate faster message propagation.
    - **Disadvantages:**
        - **High Overhead:**
          Each SuperPeer must maintain connections with all others, leading to increased resource consumption.
        - **Scalability Issues:**
          The number of connections grows quadratically with the number of SuperPeers.
2. **Linear Topology:**

    Description:
    SuperPeers are arranged in a linear sequence, where each SuperPeer connects only to its immediate neighbors.

    Advantages:

    - **Reduced Overhead:**
      Each SuperPeer maintains fewer connections, conserving resources.
    - **Scalability:**
      More scalable as the number of connections grows linearly with the number of SuperPeers.

    Disadvantages:

    - **Higher Latency:**
      Messages may need to traverse multiple SuperPeers to reach their destination, increasing latency.
    - **Single Points of Failure:**
      Disruptions in the linear chain can fragment the network, affecting connectivity.

Design Decision:

- **Configurable Topology:**
  The system supports both All-to-All and Linear topologies, allowing flexibility based on network size and performance requirements. Configuration is managed via topology.txt and topology_config.txt.

---

7. Design Trade-offs

7.1 Push-Based Approach

Trade-offs Considered:

- **Consistency vs. Bandwidth:**
  Achieves strong consistency by promptly invalidating cached copies but at the cost of increased bandwidth usage due to broadcasting.
- **Simplicity vs. Efficiency:**
  Simple to implement using RMI's remote method invocation but inefficient in large-scale networks due to the volume of invalidation messages.

Design Choice:

- Implemented push-based consistency for scenarios requiring immediate consistency assurance, suitable for environments where bandwidth overhead is manageable.

7.2 Pull-Based Approach

Trade-offs Considered:

- **Consistency Latency vs. Bandwidth Usage:**
  Reduces bandwidth usage by limiting invalidation messages but introduces latency in consistency due to periodic polling intervals.

- **Configurability vs. Complexity:**
  Polling frequency can be configured to balance consistency assurance and network load, adding flexibility at the expense of additional configuration management.

Design Choice:

- Implemented pull-based consistency with a configurable TTR (Time-To-Refresh) value, allowing LeafNodes to balance between consistency and network efficiency based on application needs.

7.3 Scalability and Efficiency

Considerations:

- **Broadcast Mechanism:**
  In push-based consistency, broadcasting to all SuperPeers can lead to network congestion. The system mitigates this by leveraging the underlying RMI framework's efficient message routing.
- **Polling Frequency:**
  In pull-based consistency, high polling frequencies enhance consistency but increase network load. The system allows configuring TTR values to optimize this balance.

Trade-offs:

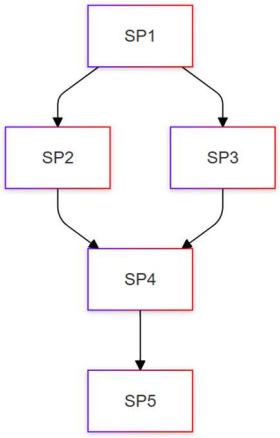
- **Push-Based:**
  Better consistency assurance at the cost of higher network overhead.
- **Pull-Based:**
  Reduced network load but potential for stale data during the polling interval.

8. Possible Extensions and Improvements

8.1 Advanced Broadcasting Techniques

Description:

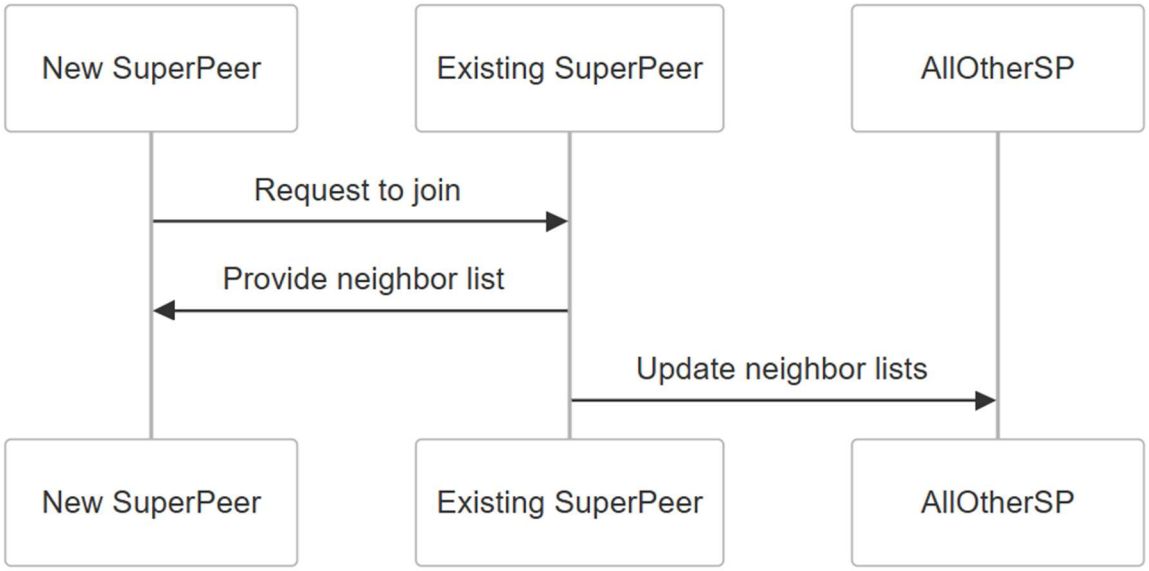
- **Hierarchical Broadcasting:**
  Implement a tiered broadcasting mechanism where invalidation messages propagate through subsets of SuperPeers, reducing overall message volume.
- **Multicast Communication:**
  Utilize multicast protocols to efficiently distribute invalidation messages to multiple SuperPeers simultaneously.



8.2 Dynamic Topology Handling

Description:

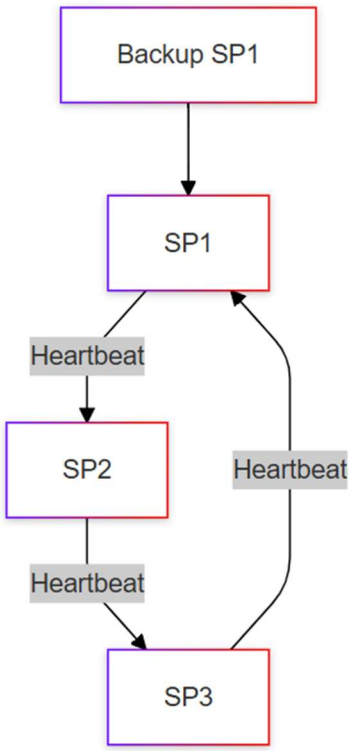
- **Peer Join/Leave Mechanism:**
  Enable SuperPeers and LeafNodes to dynamically join or leave the network, updating their connections and indices accordingly.
- **Topology Reconfiguration:**
  Allow the system to adapt its topology (e.g., switching between All-to-All and Linear) based on network conditions or peer availability.



8.3 Enhanced Fault Tolerance

Description:

- **Redundant SuperPeers:**
  Implement redundancy by having backup SuperPeers that can take over in case primary SuperPeers fail.
- **Heartbeat Mechanism:**
  Use heartbeats to monitor the health of SuperPeers and LeafNodes, triggering failover procedures upon detection of failures.



9. Conclusion

The designed hierarchical Gnutella-style P2P system effectively balances the need for strong file consistency with network efficiency by implementing both push-based and pull-based mechanisms. Leveraging Java's RMI framework facilitates seamless communication between SuperPeers and LeafNodes, while configurable topologies allow the system to adapt to various network sizes and conditions. The system's modular design ensures scalability and maintainability, with clear separation of concerns across its components.

Future enhancements, such as advanced broadcasting techniques, dynamic topology handling, enhanced fault tolerance, and security features, can further bolster the system's robustness and

applicability in real-world scenarios. By addressing the outlined design trade-offs and implementing the suggested improvements, the system can achieve higher efficiency, scalability, and resilience, aligning with the objectives of distributed file consistency management in P2P networks.

10. Configuration Files

1. consistency_config.txt:

PUSH or PULL

Description:
Specifies the consistency mechanism to be used across the P2P network. It can be set to either PUSH or PULL based on the desired consistency strategy.

2. peerconfig.txt:

P01,4001,master_dir_P01,cached_dir_P01,SP01

P02,4002,master_dir_P02,cached_dir_P02,SP01

P03,4003,master_dir_P03,cached_dir_P03,SP01

P04,4004,master_dir_P04,cached_dir_P04,SP02

P05,4005,master_dir_P05,cached_dir_P05,SP02

P06,4006,master_dir_P06,cached_dir_P06,SP02

P07,4007,master_dir_P07,cached_dir_P07,SP03

P08,4008,master_dir_P08,cached_dir_P08,SP03

P09,4009,master_dir_P09,cached_dir_P09,SP03

P10,4010,master_dir_P10,cached_dir_P10,SP04

Description:
Lists all LeafNodes (peers) with their respective configurations:

- Peer ID
- Port Number
- Master Directory Path
- Cached Directory Path
- Associated SuperPeer ID

3.superpeerconfig.txt:

SP01,5001,P01,P02,P03

SP02,5002,P04,P05,P06

SP03,5003,P07,P08,P09

SP04,5004,P10,P11,P12

SP05,5005,P13,P14,P15

SP06,5006,P16,P17,P18

SP07,5007,P19,P20,P21

SP08,5008,P22,P23,P24

SP09,5009,P25,P26,P27

SP10,5010,P28,P29,P30

Description:
Details all SuperPeers with their configurations:

- SuperPeer ID
- Port Number
- Associated LeafNodes IDs

4. topology.txt:

All or Linear

Description:

Defines the network topology to be used. It can be set to all for All-to-All Topology or linear for Linear Topology.

5. topology_config.txt:

All_To_All_Topology and Linear Topology Properties : Peer_ID; Linear_Neigbour; All_Neighbours

SP01;SP02;SP02,SP03,SP04,SP05,SP06,SP07,SP08,SP09,SP10

SP02;SP03;SP01,SP03,SP04,SP05,SP06,SP07,SP08,SP09,SP10

SP03;SP04;SP01,SP02,SP04,SP05,SP06,SP07,SP08,SP09,SP10

SP04;SP05;SP01,SP02,SP03,SP05,SP06,SP07,SP08,SP09,SP10

SP05;SP06;SP01,SP02,SP03,SP04,SP06,SP07,SP08,SP09,SP10

SP06;SP07;SP01,SP02,SP03,SP04,SP05,SP07,SP08,SP09,SP10

SP07;SP08;SP01,SP02,SP03,SP04,SP05,SP06,SP08,SP09,SP10

SP08;SP09;SP01,SP02,SP03,SP04,SP05,SP06,SP07,SP09,SP10

SP09;SP10;SP01,SP02,SP03,SP04,SP05,SP06,SP07,SP08,SP10

SP10;SP01;SP01,SP02,SP03,SP04,SP05,SP06,SP07,SP08,SP09

Description:
Defines the neighbors for each SuperPeer based on the chosen topology:

- **Peer_ID:** SuperPeer ID
- **Linear_Neighbor:** Immediate neighbor in Linear Topology
- **All_Neighbors:** All connected neighbors in All-to-All Topology

11) References

- **Java RMI Documentation:** https://docs.oracle.com/javase/8/docs/technotes/guides/rmi/index.html
- **Gnutella Protocol:** https://en.wikipedia.org/wiki/Gnutella