Implementation Tasks:

Understanding BitTorrent

- Grasp BitTorrent protocol fundamentals (torrent files, peers, seeders, etc.)
- Learn the file-sharing mechanism among peers
- Acquire knowledge of vital networking concepts (sockets, IP addresses, ports)

Setting Up the Development Environment

- Choose a development language: **Java** or C/C++
- Setup necessary libraries and dependencies (networking, file I/O, threading)

Designing the Architecture

- Outline the P2P system architecture, defining peer communication and file-sharing
- Establish communication protocols and data exchange methods among peers

Implementing Core Features

- Enable peer communication
- Implement file segmentation and reassembly logic
- Develop features for uploading, downloading, and tracking file pieces

Testing and Debugging

- Conduct unit testing for individual components
- Perform integration testing to ensure smooth interaction between components
- Debug and rectify issues in the implementation

Optimization and Scalability

- Optimize code for enhanced performance and resource utilization
- Ensure the P2P system is scalable for numerous peers and various file sizes

Documentation and Submission

- Provide thorough code documentation adhering to coding standards
- Optionally, craft a user guide or README for software usage
- Submit all project files, documentation, and additional requirements before the deadline

Reliable Transport Protocol:

- Utilizes TCP for dependable communication.

Symmetrical Interaction:

- Equal peer-to-peer interaction in both directions.

Handshake Mechanism:

- Initiates communication with a handshake message.
- Message structure:
    - 18-byte header: 'P2PFILESHARINGPROJ'.
    - 10 zero bytes.
    - 4-byte integer: Peer ID.
- Total message length: 32 bytes.

TCP Communication:

- Establish TCP sockets using suitable libraries/system calls.
- Implement connection establishment between peers.

Handshake Implementation:

- Develop a function for handshake message creation.
- Combine header, zero bits, and peer ID.
- Develop a function to parse handshake messages.
- Extract peer ID and validate format.
- Ensure handshake exchange upon connection.

Message Exchange (Post-Handshake):

- Prepare for actual message exchange post-handshake, pending further details on message structure and purpose.

Error Handling:

- Manage scenarios like absence of handshake message, incorrect format, and other communication errors.

Logging and Monitoring:

- Implement logging to facilitate the tracking and debugging of the communication process, ensuring effective monitoring of handshakes, message exchanges, and potential errors.