Overview

The networking module is responsible for facilitating reliable communication between clients and the server. It ensures seamless data transmission, efficient matchmaking, authentication, and real-time messaging.

Design Principles

1. **Modular Architecture**: The system is structured into independent components (Client, Server, Protocol, Connection, etc.) to promote maintainability and scalability.
2. **Separation of Concerns**: Each class is responsible for a specific task, reducing dependencies and improving testability.
3. **Error Handling & Resilience**: Built-in mechanisms for handling failures, such as retries, logging, and error codes, ensure robustness.
4. **Security & Authentication**: The AuthenticationManager ensures secure access to the system, handling user logins and logouts.
5. **Scalability**: The system supports multiple concurrent connections and manages matchmaking dynamically.

Development Approach

1. Initial Requirements

- Key functionalities (authentication, matchmaking, message exchange, etc.).
- Ensure compliance with security and performance requirements.

2. Component Design

- Define clear interfaces for each class.
- Implement data encoding/decoding for efficient transmission.
- Ensure loose coupling to enhance maintainability.

3. Implementation Details

- **Protocol Handling**: Implements encoding/decoding for data consistency.
- **Error Management**: Graceful failure handling through logging and retries.
- **Thread Management**: Supports multi-threading for handling multiple clients.

Potential Criticism & Considerations

1. **Performance Bottlenecks**: Handling a large number of concurrent users may require further optimizations.
2. **Security Risks**: Requires continuous updates to address vulnerabilities.

3. **Extensibility**: Future enhancements, such as WebSocket support, may require refactoring.
4. **Complexity:** Increasing complexity means challenges for development speed/progress.