From a networking and communication viewpoint:  
The present IRC server implementation doesn’t include the idea of server operators, who have unique permissions to ban users, punish users, and establish topics in channels. To illuminate this, different upgrades are vital, such as the arrangement of an Administrator Mode for clients that recognizes them as administrators and the utilisation of the 'MODE' command to advance or downgrade clients. Operators need to too be able to use commands like 'BAN' and 'KICK' to force clients to exit channels or avoid them from entering. The IRC protocol holds up both channel and client modes, permitting it to oversee client conduct and control message preparation.  
Adding password protection to IRC servers and channels might improve security and provide administrators more control over who can join the server or particular channels. Server and channel passwords can be entered, with the latter being saved and verified when a client attempts to connect. Clients can use the ‘WHOIS’ command to get information about other server users, including their nickname, username, and channels. The IRC server might increase privacy by allowing invite-only channels and encouraging users to join private ones. This would need operators or members with adequate rights implementing the 'INVITE' command.  
Logging and monitoring are critical for administration and troubleshooting in production situations. All key server events are recorded via activity logging, including user connections/disconnections, channel joins, and messages. Error monitoring detects and reports failures such as failed connections, incorrect instructions, and repeated login attempts.  
However, these extensions complicate the server implementation by affecting message processing, data flow over sockets, and server state preservation across many connections. Maintaining scalability, performance, and security takes careful consideration, particularly as the number of consumers increases. Furthermore, to manage network resources and minimise resource depletion, sophisticated error handling and dynamic resource allocation algorithms are required.

Moreover, with the expansion of the code base, it will need to be easily maintainable. This can be achieved by utilising techniques like dependency injection and rethinking the datatypes used in the system. Implementing new features and further development of the system requires more thought through core data types. Splitting the socket functionality and IRC protocol into 2 different libraries may also help the code being more maintainable.