CS 451
Fall 2024
Assignment 2

1. Client-Server Architecture

The Hangman game fits into the client-server architecture by separating the responsibilities between the server and client. The server hosts the game logic, managing the state of the game by maintaining a list of available words and tracking progress for each connected client. When a client connects, the server creates a new thread to handle that specific client's game session, allowing multiple clients to play simultaneously without interference. In contrast, the client is responsible for the user interface, collecting player input and displaying the game state. This separation enhances resource efficiency and scalability.

2. Impact of Not Using Threads and Differences Between Linux and Windows

If the server did not utilize threads, it would operate in a single-threaded manner, meaning it could only handle one client at a time. While one client is playing, others would be forced to wait. The differences between Linux and Windows include how they manage system calls and their networking APIs. However, using a high-level language like Java removes most of these differences and allows for seamless implementation.

3. Flow of Implementation and Design Choices

My program begins with server initialization, where it loads the word list and sets up a socket listener on a designated port (12345). Once a client connects, a new thread is created to manage that specific session. Then, the server randomly selects a word, sends the initial game state to the client, and enters a loop to get guesses from the client. When a letter guess is sent, the server checks if the chosen word has that letter, updates the game state accordingly, and communicates this back to the client. The game continues until the word is guessed or the client uses all their lives, after which the server prompts the client to either play again or exit.

4. TCP vs. UDP and Additional Logic for UDP

TCP is connection-oriented and ensures that data is delivered in the correct order, making it suitable for applications that require consistent data flow. In contrast, UDP is connectionless and lacks guarantees for packet delivery, which can result in lost or out-of-order messages. If the

Hangman game was implemented using UDP, it would need additional logic to handle these issues. It will have to keep asking the client if the information it received was accurate using acknowledgment messages and will need to allow for retries in order to ensure reliable communication. Since there is no connection, the program will need to ensure that both the server and client are synchronized during gameplay. Using UDP will require a lot more complex logic to ensure the same level of reliability.