Quoridor
Networking Implementation Ability

The Goal:

At the beginning of the project, the client expressed a desire to enable network play of the game among players on separate computers. Network play can be accomplished in several different ways, and this document outlines the software's current ability to accommodate one of those methods.

Network Strategy:

Our plan for connecting four players involves a typical server/client relationship. One player acts as the server and accepts three opponents. AI opponents would run directly on the server host, and all human opponents would connect as clients. Login authentication and security can be layered if required.

Communication between server and clients would likely involve a TCP/IP connection. The server would assign an IP address and port for connection. These connection details could be relayed to the other opponents via some method outside the system (email, text message, etc.). Once clients are connected, packets describing game states can be transferred between server and client.

At each turn, get the GameState, send it to the server, which forwards to all clients. Everybody calls BuildAssets with the latest game state. Each player is now synced, and play can continue.

Available Framework:

GameState Class: To capture the current state of the game, the GameState class has been provided to capture three pieces of data (Player currentPlayers[4], GameTile allTiles[rows][cols], and int nextToPlay). All games can be reconstructed from their GameState using BuildAssets.

BuildAssets: This factory class takes the pertinent information for starting/resuming a game and can use it to construct a game board and in-game UI, allowing play to resume. Using existing images and assets, BuildAssets can finish in a fraction of a second, with no visible lag or interruptions in the display.

LoadGame: This class can be extended to allow a server to set a game state and offer the client machine directions in loading the game state. It's pretty simple. At each turn, get the three packages above, send them to the server and all clients, and have everybody call BuildAssets with the latest game state. These asset packages are very lightweight, so packet transfer should be negligible.

MessagePanel: The InGameUI features a message panel text box in the lower-right. It is currently being used to pass game info to the player. It can easily be extended to pass network information as well. Features like in-game chatting, or server connection details, can be posted here with simple getters/setters.