GW Student
Internet-Draft
Intended status: Experimental
Expires: 12 May 2022

S. Fritz
George Washington University
8 November 2021

Networked Rummy 500

Abstract

This project is an application-level communication system for turn-based multiplayer games across the internet. Multiplayer games have been around for hundreds of years in many different implementations, from various board games to card games. In a digital age containing the internet, these games can be expanded to provide better experiences to players by giving them the ability to play with people across the world. This project specifically outlines a specification for the card game Rummy 500.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 12 May 2022.

Copyright Notice

Table of Contents

1. Introduction

This project is an application-level communication system for turn-based multiplayer games across the internet. Specifically, this document outlines information necessary to implement Rummy 500, but it could be easily applied to other turn-based multiplayer games. The specification allows for one user to act as a server and 'host' the game being played by them and their friends, reducing the overhead of needing to maintain a globally accessible server. This allows anyone to take over the role of the server and ensure that they can have access to games when they want to.

Some terminology that is used:

Host - Referring not to the end host systems of the network but rather to the player that activates a server protocol for others to connect to.

Player - A client that is operating this specific game and connecting to the host. Referring to both the machine and the person operating the machine.

Lobby - Effectively a waiting room for players to remain in after being connected to the host until the game starts.

Face card - A card in the standard 52-card deck that is either a Jack, Queen, King, or Ace.

Straight - A series of at least three cards in the same suit that all follow one after another (e.g. 5,6,7 of diamonds). Note that this does not allow 'rounding the corner' by using the Ace card as a low after the King card.

Matched set - A set of at least three cards that are all the same card type (eg. 3 Jack cards).

2. Rummy 500

Rummy 500 consists of a standard 52-card deck and multiple rounds between 2-5 players, each made up by players trying to use up their cards and getting the most points.

The objective of the game of Rummy 500 is to be the first player to reach 500 points, which are obtained by putting down straights of the same suit or matched number sets of cards, each with a size of at least 3. When a player runs out of cards, the round ends and points for the round are calculated. Points are given based on the cards that were put down during the round and using their face value (2-10) for the amount of points or a flat amount of 10 points for face cards (J, Q, K, A). If any cards are remaining in players' hands, the same point system is used to subtract from their score.

More rounds are played as needed until one player reaches at least 500 points. If two players reach 500 points on the same round (a tie), the player with the greater overall amount of points is given the victory. If both of these players end up with the same amount of points on the same round (an unlikely event), both players are credited with the victory.

3. Connection Information

The game is a single program that any host can have and can interact with without the input of another player, although to a very limited degree. In order to play multiplayer across the network, one user can activate a server (referred to as 'the host' from now on) to "host" a given match of the game. This allows other users (players) to connect to the host's instance of the game and play together across the internet.

The game will use a TCP (Transmission Control Protocol) connection to connect to the host on the public port 50100. Each player initiates a request with the public IP Address on the port 50100 to connect to the instance of the game from their own port of 55000. The host then sets up a new socket associated with the player so that it can continue to accept new players into the system while maintaining a private channel of communication with each individual player. A brief ping message is sent out every few seconds from the host to ensure that each player is still connected to the game. In the event that all players disconnect from the game before the game resolves itself normally, the host automatically cleans itself up and ends the game instance.

3.1. Connection Type

A TCP connection is used due to the focus on reliability. In an effort to ensure that each and every player action is recorded and reflected on the system, TCP is used as it ensures that messages will be sent between machines. TCP was also chosen due to the inherent handshake performed, as it will increase security of the game.

TCP communication was chosen over a UDP (User Datagram Protocol) connection due to UDP's unreliability and lack of security. While UDP has the advantage of having lower latency, ensuring that messages will arrive to their destination more quickly, it does not guarantee that the messages will actually ever arrive to their destination and it does not bother to verify the sender of the message. In order to ensure that communications are kept only between active players of the game, the TCP handshake is relied upon. UDP's lack of a reliability guarantee also prevents it from being considered, as the game relies on the fact that a player will make a move. If using UDP, the message could be easily lost in transmission, causing the game to come to a halt.

Thus, the need for reliability and security from each of the players requires a TCP connection to be utilized.

3.2. Status Messages

The game consists of a set of status messages to be interpreted by both the host and each player, relaying information about the current state of the game, moves, and connection information.

The status messages all follow the same general format. The message header consists of the type of message and the sender of the message (designated as host or player alias). The header is ended with two newline characters before the message body begins. This is then followed by the message body that could contain whatever information is necessary for each message, terminated by two newline characters in a row.

The format of the message header is as follows: Message Type: < CONNECT | TURN | MOVE | BEGIN | END | OK | ERR > Sender: < host | player alias > \n\n

The format of each message's body is included in the description of each message.

3.2.1. CONNECT

The CONNECT status message is a message used by both players and host. On the player side, it is used to send a connection request to the acting host of the game. It contains information about their ip address, alias, and a request to connect to the host's game instance.

In this instance, the message body will contain the following format: IP Address: < player ip > Port Num: < player port number > Player: < player alias >

If the host approves their request, an OK message will be sent back indicating that they are connected and to keep the connection open for game state data. At the same time, the host will send another version of a CONNECT message to all other players that were previously connected to notify them that a new player has connected to the lobby. This message expects another OK message in response. If other players fail to respond to this message, the server will continue to send out CONNECT messages to those that have not responded in order to make sure that all players are aware of the new player joining the game.

In this instance, the message body will contain the following format: New player: < player alias >

If the host does not approve their request, an ERR message will be sent back indicating that they cannot connect to the game, along with a reason why. This could be due to the game already being in progress, the desired name already being in use, or the given ip address being on the ban list of the host's program. The other main reason for an error is that the lobby could already be full; the maximum number of players has already been reached (5).

3.2.2. TURN

The TURN status message is a player-originating message that is sent to the acting host of the game. It contains the current player's turn for the given round to be submitted and confirmed by the host.

The message body will depend on the specific game in play, but generally will fall into the following format: player: < player alias > previous game state: < turn made by previous player > new turn: < desired turn > cards remaining: < number of cards in hand >

If the host approves the turn, an OK message is sent back to the player to indicate their turn was correct. At the same time, a MOVE message will be sent from the server to all players to relay what moves the current player made in their turn, as well as to update each player with whose turn it is next. The only exception to the MOVE-sending rule is when a player wants to draw a card from the pile. This is an action that needs to be communicated to the host so that it can generate a random card that is not currently held by another player and so that the host can continue to track the location of each card in the game.

If the host does not approve their request, an ERR message will be sent back to indicate that the desired turn is invalid. For Rummy 500, this could include the player trying to put down a straight or matched set too early (before they have 3 cards for it), or the player trying to take from the discard pile without immediately using the taken card.

3.2.3. MOVE

The MOVE status message is a host-originating message that is sent to all players in the game. It contains information about a player's successful move that was just made via a TURN message and indicates who the next player to go is.

The message body will be as follows: Player: < player alias > Move: < turn player just made > Next player: < player alias >

The host waits to hear that each player has received and acknowledged the move on their machine by receiving an OK message from each player.

If the host receives an ERR message from a player or the player fails to respond within a given timeout period, it will resend the MOVE message to the given player to ensure that all players are on the same page when proceeding through the game.

3.2.4. BEGIN

The BEGIN status message is a host-originating message that is sent to all players in the game when the hosting player manually ends the lobby phase of the host program and starts the game. It tells each player's program to display the starting setup for the game and contains information about the first player to go. When sending this message, the host randomly chooses hands for each player in order to make sure that it maintains information about the cards and that no players end up with the same cards. This message is also used whenever a new round is being started, to indicate that players' programs should reset to a pseudo-default state, with the state of cards resetting to a default initial draw state but the scores from previous rounds should be carried over.

The message body will contain information about the starting hand for the player as well as the first card in the discard pile. This ensures that the player has only the information that they should have when starting the game. The message indicates the scores of all players in the game as well, used to indicate the current standing of players between each rounds. First player: < player alias > Setup info: < game, things player needs to know > Round: < current round number > Scores: < list each player and their score >

The host ensures that each player responds with an OK message and will rebroadcast if an ERR message is sent in reply or no response is given before a timeout occurs, in order to ensure that each player's system is starting off on the same page at the same time.

3.2.5. END

The END status message is a host-originating message that is sent to all players in the game when one player has won the game. It contains the information about the player who won and tells player machines to stop accepting inputs from users.

The message body is as follows: Winner: < player alias > End Connection

The host ensures that each player responds with an OK message and will rebroadcast if an ERR message is sent back, in order to ensure that no more game systems continue. If no response is given before a timeout, the host will try to send the message again automatically. After three failed responses, the host automatically cuts off the connection with the player.

This message also indicates to player machines to close out their connections to the host, as the host will close their connections and stop listening to requests from players that have responded with an OK message.

3.2.6. OK

The OK status message is a universal acceptance or acknowledgment message of a previous message, depending on what it was. - For a previous CONNECT message, it is used to indicate to a player that the host recognizes them in the game and that they are connected to the game instance. - Similarly, it is used by a player to indicate to the host that they recognize a new player after a CONNECT message was sent globally to players when a new player successfully connected. - For a previous BEGIN message, it is used to indicate that the starting state of the game has been setup on the player's machine according to the specified rules in the message. - For a previous TURN message, it is used to indicate that a player's turn was correct and will be relayed to all players in the game. - For a previous MOVE message, it indicates that the move was received by a player's machine and is acknowledged in its representation of the current game state. - For a previous END message, it indicates that the player machine acknowledges that the game has ended and will close out their connection with the host.

The message body will be short: < connected / begun / valid / move received / ending >

3.2.7. ERR

The ERR status message is a universal issue message to indicate that there were errors with a previous request. The message body contains the reason for the error, depending on the request: - For a previous CONNECT message, it indicates that the player cannot connect to the host's game. This could be for a variety of reasons, including the player being on a ban list for the host's machine, the desired alias being currently unavailable, the game they are trying to connect to already being in progress, which does not allow new connection, or that the maximum number of players has already been reached. - For a previous BEGIN message, it indicates that the player did not get all of the starting information it needed. - For a previous TURN message,

it indicates that the player's desired turn is invalid and needs to be re-submitted by the player to follow the rules of the game. - This could also be sent in the event that the host is waiting on other players to resolve a previous MOVE or BEGIN message before accepting new turn input. - For a previous MOVE message, it indicates that a player has not received the new MOVE correctly and that it needs to be sent to the given player again. - For a previous END message, it indicates that a player did not correctly receive the END message and that it needs to be sent to the given player again.

The message body is generally the same for all non-CONNECT messages: Resend < previous message type >

For a previous connect message, the message body will contain an indication of why there was an error: Error Connecting < ip banned / need new alias / game in progress already / ip in use >

4. Host Program

The host program is a specific program instance used to act as the host for an instance of the game. It is a program that can be initiated by any player that wants to. When the player initiates the program, however, they are unable to join any other lobbies set up by other players until they end the host program.

Generally, the host goes through three main phases: the lobby phase, the game loop phase, and the clean-up phase. The lobby phase sets the publicly available connection and handles new connection requests from incoming players. The game loop phase handles all of the game logic and sending and retrieving players' turns across the network. The clean-up phase takes place after a player has won the game, where all of the connections will be cleaned up and the program will self-terminate.

4.1. Lobby Phase

In the lobby phase, the host has just been activated by a player. The host will make the publicly available port (50100) open and listen for connection requests from the network. Upon receiving a connection request, two checks are made: the source port of the requester's machine matches the port that the player program uses (55000) and that the message is sent with the proper CONNECT status message.

When the request has been verified, the program takes in the information about the requester and decides whether or not to allow them enter the lobby.

There are five error conditions for denying entry: the IP address is already in use, the alias is in use, the IP address connecting has been banned by the host in a previous lobby, the match has already begun, or the maximum number of players have already connected. If any of these conditions are met, an ERR status message will be sent back to the requester containing the reason.

Upon acceptance into the lobby, a new socket is set up for the player to maintain a private communication channel and still keep connections open for new players. The player is added to the list of current players in the match and a new thread is created to handle additional requests from the player without blocking the program. Once all the overhead has been setup on the server, an OK status message is sent back to the player's program to indicate the player has been accepted into the lobby and should wait for the game to begin.

Once the player has been connected to the lobby, the host will regularly ping the player's machine throughout the entire lifetime of the program to make sure that the player is still connected to the game. In the event that a player becomes disconnected from the game and misses three pings in a row, their data is cleaned up from the host program and the associated socket is closed.

This same process takes place behind the scenes when the hosting player initiates the host program. It records the IP address as being 'localhost' and takes in a specified username for the hosting player. If there is an error in setting up the host program and connecting the hosting player to the lobby, the host program immediately shuts down.

For the lobby phase to come to an end, the hosting player will manually send a signal to the host program to begin the match, at which point a BEGIN status message will be sent to all players connected and the program will prevent any new players from connecting to the lobby. This message contains the randomly chosen first player of the game as well as the starting hand for each player and the starting card in the discard pile.

The host ensures that all players respond with an OK message, by either being signaled by a player-originating ERR message or a timer expiration before allowing the next phase to begin or before processing any new messages. If the chosen first player tries to make their turn before all players have responded with an OK message, it is ignored and an ERR message is sent to the player to tell them to try again later.

4.2. Game Loop Phase

The game loop phase makes up the bulk of the game, as it handles designating the current turn and ensuring that all of the turns follow the rules of the game. When the loop starts, the first player has been chosen by the transition from the lobby phase to the game loop phase, via the BEGIN message sent to all players.

The program then waits for the player to submit their turn in a TURN status message. The turn is checked to make sure it is a valid move against the rules of the game. Valid turns consist of putting down a straight or matched set with at least 3 cards and the conditions of the chosen straight/matched set being met. They could also consist of just simply drawing from the deck or discard pile. If the turn is not valid (a straight or matched set is attempted without meeting the conditions or the discard pile is drawn from without immediately using it), the host responds with an ERR status message, to indicate that the player needs to make a valid move, and waits for the player to send a new TURN message.

If the turn is valid, the host responds to the player with an OK message and then relays the turn to all players inside a MOVE status message along with the next person to go. The host waits for each player to respond with an OK message to indicate that the turn is reflected on the player's machine. If the host receives an ERR message from a player to indicate that the move was not received correctly, the host sends the data again. While waiting for all players to respond with an OK message, the host will not process new TURN messages from the next player that already responded with an OK message. In this case, the host would respond with an ERR message to tell the player to try again in a little bit due to the wait from one player failing to acknowledge the last move.

Once one player runs out of cards, the point totals for the round are calculated and a new round is started up via a new BEGIN message with the next round started and a new set of cards for each player.

This loop repeats over and over until the win condition for the game has been met, when the host sends each player an END status message and the clean-up phase begins. During the entire game loop phase, the host is frequently pinging each player to make sure that they are still connected to the lobby. In the event that a player misses three pings in a row, they automatically are cleaned up and disconnected from the lobby in a manner similar to how it is done in the clean-up phase of the host program.

4.3. Clean-Up Phase

The clean-up phase is begun by the sending of the END status message to all players. It indicates the winner of the game, as well as a sign for each player to end their connection with the host. The host makes sure that each player responds to the message with an OK status message. If an ERR message is received in reply, or no reply is received before a timeout, the host will send the END message again to the player. If a player misses three timer expirations, they are automatically disconnected and cleaned up by the host.

Upon receiving an OK status message in reply to the END message, the host program cleans up the data and resources associated with handling the player's connection. The socket and port are closed off, the player is deleted from the list of current players, and the associated thread terminates. Once all of the players have been cleaned up, the host program self-terminates.

5. Player Program

The player program is the general instance of the game that each player runs. It translates all of the data from the host and translates it into something that can actually be easily seen by the player. This program serves as the input and output of all of the players' moves and connections to the host lobby.

The host program has 4 main phases: the idle phase, the lobby phase, the game phase, and the clean-up phase. The idle phase represents the running state of the program when not connecting to a host lobby. It can perform some basic tasks and will remain running when not connected to a host, but ultimately serves to give the basic choice of whether the player wants to host a game or connect to an existing host lobby. The lobby phase is similar to the host's lobby phase, but on the other side of the connection process. The game phase is the actual time in the game, handling opponents' turns and the input of the specific player's turn. The clean-up phase is the other side of the host's clean-up phase; it handles cleaning up the connection and returning to idle.

5.1. Idle Phase

The idle phase is the basic version of the program. It is not connected to any other computer and will just wait until the player manually closes down the program. It allows basic options, like choosing what alias the player wants to go by and whether or not they want to host a game or connect to an existing host lobby.

Since the player is not connected to any other player or host and is not waiting on any input outside of the player, there are not error conditions during this phase.

5.2. Lobby Phase

The lobby phase is similar to the lobby phase of the host program by handling the other side of the connection that the host program deals with. When a player decides to connect to a host lobby, it takes the current alias the player has decided on and the current IP address of the host and sends the CONNECT status message out of the player port (55000) to the host's public connection port (50100) as a request for connection.

If an ERR message is received in reply, the player program will display a warning accordingly to the screen, and whether or not the player can do anything to remedy it. For example, if the ERR message is due to the fact that the alias is already in use, the warning will contain the ability to change the alias and try again. On the other hand, if the ERR message is due to the fact that IP address has been banned by the host, the warning message will offer no options beyond returning to the idle phase.

If an ERR popup message requires new input, it is packaged into a new CONNECT message to try to connect to the host again.

Once an OK message is received from the host, the program waits until the host player decides to start the game, when it receives a BEGIN message from the host. At this point, it will enter the game phase.

5.3. Game Phase

The game phase is begun by the initial BEGIN message from the lobby phase. The player gets their assigned starting hand and information about the starting discard pile and uses it to initialize on-screen information for the player's machine.

The game phase is the main aspect of the program, as it handles displaying opponents' turns and taking input for the player's turn and converting on-screen data to and from a network message.

If it is not the player's turn, as designated by the MOVE status messages received from the host, the program takes the associated move from the message and converts it into something on-screen that can be seen by the player. Upon receiving any MOVE message from the host, the player program sends back an OK message to acknowledge that the move has been received by the program. If it is not the player's turn, the player program will prevent any input from being given by

the player and will only send OK messages behind the scenes and display the output of the incoming messages until it is the player's turn, at which point they will be allowed to give input to the program.

Sending messages is done by temporarily creating a new thread to send a message to the host while maintaining the main thread of the program to be listening for messages from the host. If the MOVE message received from the host is unclear or missing pieces, an ERR message is sent back in reply to indicate that the MOVE message needs to be sent again.

If it is the player's turn, they can decide on the next move that they should make based on the information available to them on the screen. Upon submission of the player's turn, a TURN message is sent to the host containing the move that the player wants to make. The player program then waits for an OK in reply to indicate that the turn was valid and will be relayed to all other players. This will then be followed soon afterwards by a MOVE message containing their move. This will be used to update the player's visual interface to match all other player's interfaces. If an ERR is received in reply to the TURN message instead, the turn was not valid and the program will re-open the ability for the player to make a move until the host returns an OK message. The player program will provide some error checking to prevent invalid turns before they are sent to the host by checking to make sure that the number of cards in a straight or matched set being put down is at least the right amount, but the host will check more thoroughly that the move is valid.

The game phase bounces between the player performing their turn and recording opponents' moves. When a round finishes and a new round begins, the host sends another BEGIN message out to the players to let the process start again.

When an END status message is received from the host, the game phase ends and the clean-up phase begins.

5.4. Clean-Up Phase

The player's clean-up phase is similar to the host's clean-up phase, as they each represent both sides of the same action. When the player receives the END message, it is an indication that someone has won the game, which will be shown on-screen, and that the connection with the host needs to be cleaned up.

As soon as the message is received, the program sends an OK message back to the host and then starts dismantling all of the setup for the game. The socket and port are closed off and the information on the

screen is cleared off, except for the winner of the game, for a few seconds. Once this time expires, the program returns entirely to the idle phase and waits for a new connection to be set up, either by connecting to a new lobby or setting up a lobby on the player's machine.

6. Security Considerations

Leaving ports open can create problems with foreign programs trying to connect to a machine. For this reason, each request is examined upon being made to the given port on the host's machine. If the request does not follow the correct message type or format, then the message is disregarded. Additionally, depending on the current state of the host program, requests can be ignored or sent back an error message to prevent them from connecting. Another method to mitigate this is by cleaning up connections as soon as possible. If a player should disconnect by failing three pings in a row, all associated resources for the player are cleaned up and the player is removed from the lobby. If there is an issue in setting up the host program and connecting the player to their own lobby, then the program automatically terminates to prevent any foreign connections from taking hold. As soon as the game is over, the host tells all players to end their connections with the host, at which point the host removes all resources for connecting players and then terminates itself to prevent any foreign programs from connecting to the host.

In order to prevent overuse of the host player's machine and to keep the network messages at a low rate, there is a maximum number of players. The other inherent reason behind this is the fact that a standard 52-card deck can only have so many players using the deck and still keeping the size of draw pile to a reasonable amount. With a limit on the number of players, it ensures that matches can happen in a much faster manner by not having to wait on as many message confirmations and that the amount of resources used for hosting is relatively low.

Author's Address

Samuel Fritz
George Washington University