                   Turn-Based Multiplayer Network Games

Abstract

   This project is an application-level communication system for turn-
   based multiplayer games across the internet. Mutliplayer games have
   been around for hundreds of years in many different implementations,
   from various board games to card games. In a digital age containg
   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.

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 7 April 2022.

Table of Contents

1. Introduction

   This project is an application-level communication system for turn-
   based multiplayer games across the internet. This 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.

2. Games

   Due to the high specificity of every game, no specific game rules
   will be mentioned outside of this section. This ensures that the
   overall description can remain relevant for any turn-based game to be
   implemented. For the sake of this project, Rummy 500 will be the
   primary focus of development and a secondary focus on Go Fish.

   Rummy 500 consists of a standard deck of cards and multiple rounds,
   where each round consists of trying to score as many points as
   possible by putting down sets of the same card type or in sequences
   of 3 or 4 cards of the same card suit. Turns consist of choosing to
   either take from the top of the deck or taking any card from the
   discard pile along with all cards above it. Each round ends when one
   player runs out of cards and the rounds continue until one player
   reaches 500 points. Each number card is worth its face value and
   face cards are each worth 10 points. If any cards are remaining in
   hand, then their point value is subtracted from the total point value
   for the round.

   Go Fish consists of a standard deck of cards, with the goal of
   putting down as many groups of 4 of the same card type as possible.
   Each turn gives the ability to prompt another player to give up a
   specific card type, if able. If unable, the player draws from the
   top of the deck; if they receive the card type they wanted, they can
   continue playing until they are unsuccessful in getting a card type.
   The game ends when all card type groups have been put down and the
   player with most card type groups wins.

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, 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, 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 type is the first part of the message, with two newline characters indicating the end of the message type. 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.

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.

   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.

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 >

   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.

   If the host does not approve their request, an ERR message will be
   sent back to indicate that the desired turn is invalid.

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, 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.

   The message body will differ a little bit depending on the game, as
   resources need to be set up globally for all players to see, but need
   to be randomized for each game (like a discard pile that starts with
   a card in it or what each hand starts with). First player: < player
   alias > Setup info: < game, things player needs to know >

   The host ensures that each player responds with an OK message and
   will rebroadcast if an ERR message is sent in reply, 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.

   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 acknowledgement
   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. - 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
   is currently unavailable, or the game they are trying to connect to
   is already in progress and cannot be joined after it starts. - 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. - 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 >

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, but they are unable to connect
   to another host lobby while their hosting protocol is active.

   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 as being legit, the program takes
   in the information about the requester and decides whether or not to
   allow them enter the lobby. There are three main conditions for
   denying entry: the IP address is already in use, the alias is in use,
   or the IP address requesting has been banned by this host in a
   previous lobby. 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, 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 just
being 'localhost' and takes in a specified username for the hosting
player. In the event that this player cannot connect to the host
program, the program is terminated before any other connections can
be made active.

For the lobby phase to come to an end, the hosting player will
manually hit a button 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 and
indicates that each player's program should display the starting
state of the game. The host ensures that all players respond with an
OK message before allowing the next phase to begin or before
processing any new messages.

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. If the turn is not valid, 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.

   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.

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, the host will send the END message again to the player.

   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, as

they both handle setting up the connection before the game begins.
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.

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.

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 either the game
phase.

5.3. Game Phase

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. 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 against the rules
   of the game 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, 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 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.

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, 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.

Author's Address

   Samuel Fritz