CS594 Portland State University
Internet Draft November 1, 2023
Intended status: RFC Class Project Specification
Expires: June 2024
Team Members: Chethana Muppalam , Venkata Tejaswi Battula

Network Game - Block Race Multiplayer Game

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on Fail 1, 0000.

Copyright Notice

Abstract

This memo describes the communication protocol for a Multiplayer Network Game that Involves client/server system for the Internetworking Protocols class project at Portland State University.

Table of Contents

1. Introduction:

This specification describes a fun online multiplayer game where multiple users race their blocks towards reaching the goal. An in-game chat feature is implemented for players to create/join groups and communicate with other players in the game network. The game and chat feature are implemented using TCP and client server socket programming in Python.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT","SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying significance described in RFC 2119.

In this document, the characters ">>" preceding an indented line(s)indicates a statement using the key words listed above. This convention aids reviewers in quickly identifying or finding the portions of this RFC covered by these keywords.

3. Basic Information:

The server can either run locally or on the cloud, and the client will establish a connection to the server through the terminal, passing commands as needed. All the described communication occurs over TCP in a client-server architecture. Multiple clients can connect to the server and maintain a persistent connection. Each client connection will spawn a new thread responsible for handling incoming commands and messages.

Both the server and client may terminate the connection at any time for any reason. They MAY choose to send an error message to the other party informing them of the reason for connection termination.

The server may choose to allow only a finite number of users and game rooms, depending on the implementation and resources of the host system. Error codes are available to notify connecting clients

that there is currently a high volume of users or groups accessing the server.

4. Game Infrastructure
    1. Multiple clients can connect to the server.
    2. Color Codes:
        - ➢ Blue Block: Initial state of the opponent player.
        - ➢ Red Block: Current player's move.
        - ➢ Yellow Block: Opponent player's move.
        - ➢ Green Block: Winner.
        - ➢ Black Block: Loser.

    Note: When a new player joins the network as an opponent, their block color changes from blue to yellow on the initial player's screen.
    4. Players can move their blocks on their canvas.
    5. Winner: The first player to reach the goal with their block wins.
    6. Chat:
        - ➢ Players can create a new room or join existing chat rooms
        - ➢ Players can send messages in the chat room
        - ➢ Players can leave/quit the chat room
        - ➢ Players can list the users in the chat room

    Other Features:

    1. Clients can gracefully handle server crashes and exceptions
    2. Server can gracefully handle unresponsive/crashed clients

5. Label Semantics

- o Move Validation: Any player's move must comply with the game's rules.
- o Player Disconnection Handling: If a player faces network interruption during their turn, the game should pause, offering a reconnection window. If unsuccessful the client will be disconnected from the game.
- o State Synchronization: The game server should ensure that all connected players have a synchronized view of the game state, minimizing discrepancies and conflicts.
- o Game Over Conditions: Detecting and declaring the end of the game, either based on specific conditions.

- Game Restart/Rematch: If players opt for a rematch or a new game after completion, the game server should handle this request appropriately, resetting the game state for all players involved.

6. Client Messages

Client messaging involves the communication and interaction between the client-side game implementation and the server.

This messaging mechanism facilitates the exchange of data related to player actions, positions, and game state updates between the client and the server. The client sends messages containing player positions, colors, and other relevant game information to the server, which processes this data and responds accordingly, enabling real-time updates and synchronization between different clients connected to the game.

Additionally, the client-side code implements handling mechanisms to manage the sending and receiving of messages, ensuring a coherent flow of data between the client and the server for seamless multiplayer gameplay.

The following messages are issued by client:

- Server didn't respond within the timeout - If the server doesn’t reply or send packets of data to client within a specified amount of time.
- Connection with the server was reset - If the Server connection was lost with the client for any reason, it is informed to the client.
- Unable to establish connection please try again later - If the server is unavailable in the network to serve the client.

7. Server Messages

Server messaging encompasses the handling of incoming connections from multiple clients using socket-based communication. This server-side messaging mechanism manages the reception of data from connected clients, interprets and processes the received messages, and subsequently responds to the clients based on the received information.

Each client connection is managed in a separate thread function, allowing concurrent communication with multiple clients. The server listens for incoming messages from clients, interprets the received data to determine client IDs and positional information, updates the positional data accordingly, and sends relevant responses back to the clients.

The server ensures continuous data exchange, synchronization, and coordination among multiple connected clients within the context of the established server-client architecture.

The following messages are issued by the server:

- Client didn't send data within the timeout - If the client is unresponsive.
- Connection Closed - If the client is disconnected from the server.

8. Error Handling

Both server and client are detecting when the socket connection linking them is terminated, either when actively sending traffic or by keeping track of the heartbeat messages. If the server detects that the client connection has been lost, the server removes the client from the game. If the client detects that the connection to the server has been lost, it waits until a certain time and tries a certain number of times before disconnecting. As stated previously, it is optional for one party to notify the other in the event of an error.

9. "Extra" Features Supported

In-Game Chat Feature: Players can connect with other users in the game server which supports the following features and is implemented faithfully using TCP.

- Players can create a new room or join existing chat rooms
- Players can send messages in the chat room
- Players can leave/quit the chat room
- Players can list the number of users in the chat room

Cloud-Connected Server: Utilize cloud services to make the server more scalable and reliable.

File Sharing: Allow clients to share files with each other within game rooms.

10. Conclusion & Future Work

This specification provides a generic game playing framework for multiple clients to connect with each other via a central server.

Without any modifications to this specification, it is possible for clients to devise their own protocols that rely on the text-passing system described here. For example, transfer of arbitrary binary data can be achieved through transcoding to base64. Such infrastructure could be used to transfer arbitrarily large files, or to establish secure connections using cryptographic transport protocols such as Transport Layer Security (TLS).

In future, the game can be made robust by managing a large number of Game rooms and AI Features to predict the next best move/winning strategy.

11. Security Considerations

Messages sent using this system have no protection against inspection, tampering or outright forgery. The server sees all messages that are sent through the use of this service. 'Private' messaging may be easily intercepted by a 3rd party that is able to capture network traffic. Users wishing to use this system for secure communication should use/implement their own user-to-user encryption protocol.

12. IANA Considerations

None

12.1. Normative References

[1] Socket Programming in Python
https://realpython.com/python-sockets/

[2] Network games
https://gafferongames.com/post/what_every_programmer_needs_to_know_about_game_networking/

[3] Socket Python Reference Documentation
https://www.chilkatsoft.com/refdoc/pythonSocketRef.html

13. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.