```
CS594 Internet Protocols                                    Casper Rutz
Internet Draft                                Portland State University
Intended status: Networked Game Project Specification November 29, 2020
Expires: January 2021
```

Networked Game of Graveyard
draft-ng-pdx-cs594-00.txt

Status of this Memo

Copyright Notice

Abstract

This memo describes the communication protocol for a networked version of the two-player game Graveyard, which is an assignment for the Internetworking Protocols course at Portland State University.

Table of Contents

1.Introduction

This specification describes a protocol for a networked version of the two-person Banqi game Graveyard. Clients will be able to create a game with a specific name and password, and clients can join a game by inputting the correct name and password. Game updates are set from the client whose turn it is to the server; the other client requests an update from the server, and then sends its own update on its turn. This continues until one client “wins”.

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.

3.Basic Information

All communication described in this protocol takes place over TCP/IP, with the server listening for connections on port 6413. Clients connect to this port and maintain this persistent connection to the server. The client can send messages and requests to the server over this open channel, and the server can reply via the same.

This messaging protocol is inherently asynchronous - the client and server may send messages to each other at any time. However, messages that update game state MUST be sent sequentially, from client A to the server to client B and vice versa until a client “wins”.

As described in [4.2], both the server and the client may terminate the connection at any time for any reason. The client and the server SHOULD send an error message to the other party informing them for the reason for connection termination.

The server MUST allow only two clients to connect to any one game. The server MAY choose to allow only a finite number of games, 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 games currently hosted on the server.

4.Message Infrastructure

4.1.Generic Message Format

Messages are strings, where fields are separated by a colon.

Message structure is “header:payload”.

Header structure is “code:client-id:game-name:game-password”.

4.1.1.Field Definitions

o Header - contains the required fields of every message. These fields are used by the client and server to identify the type of message sent and to establish (on the server’s end) or verify (on the client’s end) which game is specified.

o code – specifies what kind of message is contained within the payload. May be either an operation code or an error code.

o client-id – clients starting a game assign themselves an id of 1, and clients joining a game assign themselves an id of 2.

The client MUST send this value to the server, which will use this value plus the game name and game password to identify which client it is communicating with. The server MAY send this value back to the server in its response to verify that it identified the client correctly, unless it responds. If either the client or the server detects that this value is incorrect, it SHOULD provide the error code err-invalid-credentials to the opposite party (see [4.2]) and MUST sever the connection.

o game-name - a 20-character string associated with a particular game. Used in tandem with the game password to identify a game. MUST follow identifier semantics (see [5]).

  The client MUST send this value to the server, which will use this value plus the client id and game password to identify which client it is communicating with. The server MAY send this value back to the server in its response, to verify that it identified the client correctly. If either the client or the server detects that this value is incorrect, it SHOULD provide the error code err-invalid-credentials to the opposite party (see [4.2]) and MUST sever the connection.

o game-password - a 20-character string associated with a particular game. Used in tandem with the game name to identify a game. MUST follow identifier semantics (see [5]).

  The client MUST send this value to the server, which will use this value plus the game name and client id to identify which client it is communicating with. The server MAY send this value back to the server in its response, to verify that it identified the client correctly. If either the client or the server detects that this value is incorrect, it SHOULD provide the error code err-invalid-credentials to the opposite party (see [4.2]) and MUST sever the connection.

o payload - variable length payload. Different fields are also separated by a colon. Not used by some messages.

4.1.2.Operation Codes (opcodes)

```
opcode-keepalive      = 0x41 (‘A’)
opcode-update?        = 0x42 (‘B’)
opcode-update         = 0x43 (‘C’)
opcode-forward-update = 0x44 (‘D’)
opcode-create         = 0x45 (‘E’)
opcode-game-created   = 0x46 (‘F’)
opcode-join           = 0x47 (‘G’)
opcode-forward-join   = 0x48 (‘H’)
opcode-forward-leave  = 0x4A (‘J’)
```

4.2.Error Messages

Error messages are strings, whose structure is “header”. Error messages sent from the server have a header structure of “error-code”.

Error messages sent from the client have a header structure of “error-code:client-id:game-name:game-password”.

4.2.1.Usage

SHOULD be sent by either the client or the server prior to closing the TCP connection to inform the other party why the connection is being closed. If either client or server receives this message, that entity SHOULD consider the game terminated. The server then SHOULD send the opcode-forward-leave message to the remaining client.

4.2.2.Error Codes

```
err-invalid-message    = 0x4D (‘M’)
err-wrong-credentials  = 0x50 (‘P’)
err-invalid-name       = 0x51 (‘Q’)
err-3s-a-crowd         = 0x52 (‘R’)
err-too-many-games     = 0x54 (’T’)
```

4.2.3.Invalid Messages and Incorrect Credentials

err-invalid-message is sent by the client or server in response to an incorrectly formatted message.

err-wrong-credentials is sent by the client or server in response to a message containing an unexpected client-id, or an incorrect game-name and game-password.

The other error codes are covered in the following sections.

4.3.Keepalive Messages

Header of client’s message: “A:client-id:game-name:game-password”
Header of server’s response: “A”

4.3.1.Usage

Provides an application-layer notification of a disconnected peer.

MUST be sent at least once every 3 seconds by a client to notify the server that they are still active.

Both client and server MUST watch for these keepalive messages and consider the other party inactive if more than some set time period has elapsed. This time MUST equal or exceed 10 seconds.

5.Identifier Semantics

Creating and joining games involves sending and receiving identifiers. All identifier rules are the same, and MUST be validated as follows:

- o Must consist entirely of lowercase letters with ASCII character values between 0x61 and 0x7A.
- o Must be at least 1 character, and at most 20 characters.
- o If less than 20 characters, the first character following the message MUST be a null character (0x00). Remaining characters MAY also be null characters.
- o If any of these rules are broken, the receiver MUST return the err-invalid-name error code and MAY terminate the connection.

6.Client Messages

6.1.Creating a Game

Header: “E:client-id:game-name:game-password”
Payload: “pieces:piece-player”

6.1.1.Usage

Sent by the client to create a game, identified by the game name and password.

The server MUST assign an identifier to the client, in order to associate the client with the socket connection of the user. This message SHOULD be sent only once; if the server receives the message more than once, the server MAY either ignore the message or terminate the client’s connection.

6.1.2.Field Definitions

- o client-id - identifies which player is sending the message. MUST be 1.
- o game-name – identifies the game. MUST follow identifier semantics.

  game-name cannot be the same name provided by any other currently connected client. If the name already exists, the server MUST return the error err-invalid-name. The client can then attempt to reconnect with a different name.

- o game-password – used by clients to connect to this game, in order to add a small layer of security. MUST follow identifier semantics.

- o pieces - a string of letters, where each letter represents a tile piece on the board. Tile pieces are listed in increasing row order.

- o piece-player - a string of letters, where each letter represents which player that tile piece belongs to. Tile pieces are listed in increasing row order.

6.1.3. Response

Server MUST return a response with the opcode-game-created message. If this is not returned, the client MUST terminate the connection and consider the game over.

6.2.Joining a Game

Header: “G:client-id:game-name:game-password”

6.2.1.Usage

Sent by the client to join a game. The name and password specified by the client MUST both match a game currently active on the server. If this condition is not met, the server MUST return the error code err-invalid-name and MUST terminate the connection. This game MUST have exactly 1 other client connected to it. If this condition is not met, the server MUST return the error message err-3s-a-crowd and MUST terminate the connection.

This message SHOULD be sent only once; if the server receives the message more than once, the server MAY either ignore the message or terminate the client’s connection.

Upon accepting this message, the server MUST respond with an opcode-forward-join message.

6.2.2.Field Definitions

- o client-id - identifies which player is sending the message. MUST be 2.

- o game-name – identifies the game the client wishes to join. MUST follow identifier semantics.

- o game-password – the password of the game the client wishes to join. MUST follow identifier semantics.

6.2.3. Response

Server MUST respond with the opcode-forward-join message. If this is not returned, the client that sent this message MUST terminate the connection and consider the game over.

6.3.Has A Second Player Joined?

Header: “K:client-id:game-name:game-password”

6.3.1.Usage

Sent by a client every 2 seconds to see if another client has joined the game.

If a second client has not joined yet, the server MUST respond with a keepalive message to indicate that a client has not joined, but there are no errors preventing the client from continuing the game.

If a second client has joined and disconnected from the game, the server MUST respond with an opcode-forward-leave message. Both client and server should consider the game terminated.

If a second client has joined and is active, the server MUST respond with an opcode-forward-join message.

6.3.2.Field Definitions

- o client-id - identifies which player is sending the message.

- o game-name – identifies the game the client wishes to join. MUST follow identifier semantics.

  This MUST be the name of a game currently running on the server with 1 and only 1 currently active client.

- o game-password – the password of the game the client wishes to join. MUST follow identifier semantics.

  This MUST be the password of the game specified by the game-name.

6.4.Updating Game State

```
Header: “C:client-id:game-name:game-password”
Payload: “move-from:move-to”
```

6.4.1.Usage

Sent by a client to update the server on a new move.

The server MUST save move-from and move-to as the last move, along with which client sent this message. The server does not check if this move is valid; it is the responsibility of the clients to only send valid moves.

The server MUST send an opcode-forward-update message in response.

6.4.2.Field Definitions

- o client-id - identifies which player is sending the message.
- o game-name – identifies the game the client wishes to join. MUST follow identifier semantics.
- o game-password – the password of the game the client wishes to join. MUST follow identifier semantics.
- o move-from – The coordinates of the tile, pre-move. Consists of a column (0-7) and a row (0-3).
- o move-to – The coordinates of the tile, post-move. Consists of a column (0-7) and a row (0-3).

6.5.Polling for Updates

```
Header: “B:client-id:game-name:game-password”
```

6.5.1.Usage

Sent by a client every 2 seconds to see if the server has been sent a game update from the other client.

There has been an update if client-id in this message does not match the id of the client who sent the last update. In this case, the server MUST respond to this message with an opcode-forward-update message.

There has not been an update if the client-id in this message matches the id of the client who sent the last update. In this case, the server MUST respond to this message with a keepalive response.

6.5.2. Field Definitions

- o client-id - identifies which player is sending the message.
- o game-name – identifies the game the client wishes to join. MUST follow identifier semantics.
- o game-password – the password of the game the client wishes to join. MUST follow identifier semantics.

7.Server Response Messages

7.1.Game Created

Header: “F:client-id:game-name:game-password”

7.1.1.Usage

Sent by the server to a client in response to the opcode-create message. The game-name and game-password MUST match the game the client created, and the client-id MUST match the id of the client that sent the opcode-create message (which should be 1). If any of these are false, the client SHOULD send the server an err-wrong-credentials message and MUST consider the game terminated.

7.1.2.Field Definitions

- o client-id - ID of the client which sent the opcode-create message. MUST be 1.
- o game-name – Name of the game that was created.
- o game-password – Password to the game that was created.

7.2.Player Joined

Header: “H:client-id:game-name:game-password”
Payload: “pieces:piece-player”

7.2.1.Usage

Sent by the server to a client in response to an opcode-join message or opcode-joined? message to notify them that a second player has joined the game and the client with the first move is free to send an opcode-update message. This message also serves to send an encoding of the board to the client who joined the game, who then must create it before being able to start the game.

The name and password specified by the server MUST both match the game both clients believe themselves to be in; if not, the clients SHOULD send an error-invalid-code message (see [4.2]) to the server and MUST terminate the connection.

7.2.2.Field Definitions

- client-id - ID of the client who sent the initial message.
- game-name – Name of the game that the client joined.
- game-password – Password to the game that the client joined.
- pieces - a string of letters, where each letter represents a tile piece on the board. Tile pieces are listed in increasing row order.
- piece-player - a string of letters, where each letter represents which player that tile piece belongs to. Tile pieces are listed in increasing row order.

7.3.Player Left

Header: “J”

7.3.1.Usage

Sent by the server to the client who created a game to notify them that the second player has joined and already left the game. The server and client MUST terminate the connection and consider the game over upon sending or receiving this message.

A client leaving the game has no need to send a message notifying the server, as the server will consider the game terminated when a keepalive message has not been sent within 10 seconds.

7.4.Forwarding Updates

Header: “D:client-id:game-name:game-password”

Payload: “move-from:move-to:who-last-updated”

7.4.1.Usage

Sent by the server to a client in response to an opcode-update message or opcode-update? message.

The client should then execute the piece movement indicated by the indexes. This move is assumed to be valid, as clients should not send updates for invalid moves.

7.4.2.Field Definitions

- o client-id - ID of the client who sent the initial message.
- o game-name – Name of the game that the client joined.
- o game-password – Password to the game that the client joined.
- o move-from – The coordinates of the tile, pre-move. Consists of a column (0-7) and a row (0-3).
- o move-to – The coordinates of the tile, post-move. Consists of a column (0-7) and a row (0-3).
- o who-last-updated - the id of the client who sent the last update message.

8.Error Handling

Both server and client MUST terminate a game upon receiving or sending an error code, or when the other party has timed out due to not receiving a heartbeat message.

As stated previously, one party SHOULD notify the other in the event of an error.

9.Security Considerations

There is no protection against inspection, interception, or tampering by third parties. Users wishing to secure this system should use or implement their own encryption protocol.

10.IANA Considerations

None

11.Conclusions

This specification provides a generic message passing framework for multiple clients to communicate with each other via a central forwarding server.

12.References

12.1.Normative References

1.   Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

13.Acknowledgments

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

Author's Address

Casper Rutz
Portland State University
1930 SW Fourth Avenue, Suite 500
Portland, OR 97201

Email: catrutz@pdx.edu