Sebnem Arslan

Bosphorus University

21.12.2014

**SWE 544 BACKGAMMON Application: System-Wide Requirements**

**Status of This Memo**

This memo defines an experimental protocol and requirements for client and server applications of BACKGAMMON: a client-server architecture application.

**Abstract**

The BACKGAMMON application requirements and protocol were developed as part of the BACKGAMMON term project effort, in partial fulfillment of the SWE544 Internet Programming course requirements at Bogazici University.

The application includes a server (i.e. Game Application Server) and a client (i.e. Player Client) implementation as well as the definition of a protocol for client-to-server communications. The backgammon system has features such as:

* Connect to System
* Play a game (Join, throw dice, send move, notify wrong move alert)
* Watch a game
* Disconnect the System

**Revision history**

|  |  |  |
| --- | --- | --- |
| Version | Changed By | Date |
| 1 | Initial Draft | 21.12.2014 |
| 2 | Corrected typos and protocol port. Modified heart beat logic as to direction from server to client | 1.1.2015 |
|  |  |  |

Table of Contents

[**Status of This Memo** 1](#_Toc407145603)

[**Abstract** 1](#_Toc407145604)

[1 INTRODUCTION 3](#_Toc407145605)

[1.1 Definitions 3](#_Toc407145606)

[2 SYSTEM-WIDE REQUIREMENTS 4](#_Toc407145607)

[2.1 Functional System Requirements 4](#_Toc407145608)

[2.2 Non-Functional System Requirements 6](#_Toc407145609)

[3 BACKGAMMON Application Server Requirements 6](#_Toc407145610)

[3.1 Application Server Functional Requirements 6](#_Toc407145611)

[3.2 Application Server Non-Functional Requirements 7](#_Toc407145612)

[4 BACKGAMMON Client Requirements 7](#_Toc407145613)

[4.1 Player Client Functional Requirements 7](#_Toc407145614)

[4.2 Player Client Non-Functional Requirements 7](#_Toc407145615)

1. INTRODUCTION

Backgammon is a game played between two players. The playing pieces are moved according to the roll of dice, and a player wins by removing all of his pieces from the board before his opponent.

BACKGAMMON system is a game application and accompanying protocol for the Internet. The application features a server (i.e. Application Server, here forth "App Server") that forms the backbone of the game system, and is responsible for keeping track of the status of participating clients, and coordinating data about the clients and server.

The application also features clients (i.e. Player Client, here forth "PC"), which are Python based client applications that communicate with GC to connect, request to start playing, watching a game and playing a game.

The two key actors on the system are the User and the System, which can further be broken down into Receiving and Sending stone movements for game scenarios, and the client and server-side components of the backgammon game system as a whole.

This document is intended for the use of AS and PC Design & Development Teams as well as testers in the BACKGAMMON ecosystem. It contains the elicited system-wide requirements and an analysis of requirements for the client and server side components of the system. The protocol to be used, between the client and the server however, is defined in a separate document.

1.1 Definitions

This document uses the following definitions:

**Server**, **application server**, **AS**: is the server-side component of the BACKGAMMON system that uses the **PC-AS** protocol of messaging and keeps track of the active game client sessions, along with IP addresses and usernames.

**Client**, **player client**, **PC**: is the client-side component of the BACKGAMMON system that allows for retrieving a list of players from the server, and messaging with opponent clients connected to the system throughout application server. The client presents a Graphical User Interface to the User.

**User**: The user is any real person using the BACKGAMMON client application.

**Protocol**: The text-based messaging protocol, based on TCP, used for the communication of PC and AS components. The Protocol is defined in a separate RFC document (RFC2).

**System**: refers to the BACKGAMMON system as a whole, incorporating both server and client side code

1. SYSTEM-WIDE REQUIREMENTS
   1. Functional System Requirements

A User, of the system is any human actor who interacts with the BACKGAMMON system through use of the client. The users of the system shall be able to:

|  |  |
| --- | --- |
| Requirement # | Definition |
| 1.1.1 | Connect to the BACKGAMMON system, through their client software, by providing a valid username. User name is unique within the current connected users. After a user disconnected the system, same username of disconnected user can be used by next user.  If a User loses connection to the server, any open game sessions will be lost and the User shall need to re-connect using the client. |
| 1.1.2 | After user connected, the below welcome message is shown:  **“Hi Ali,**  **You are connected to <<Application Server IP>>, <<Connected Port>>”**  If username already exists in the system, the below system message is shown  **“Ali is already exists, Choose another username.”** |
| 1.1.3 | After user successfully connected, Two action link is shown to user on Welcome screen.   * **“I want to play”** * **“I want to watch”** |
| 1.1.4 | I want to play: Once a user requests game, application server receives the request and match the request randomly to an opponent. The opponent user shall check the assigned game by clicking on I want to play.  Once application server matches user to an opponent, the below message shall be displayed to both users on their client screens:  **“Your opponent is Mehmet (Ali is shown in opponent’s screen)”**  -**“Start playing”** link is active.  User can only play to one opponent at a time. Application Server shall check if client is playing. While user playing with another user, application server shall automatically eliminate the user to match for another game. Game requests will be handled between client and server. |
| 1.1.5 | If application server cannot find an active user who is waiting for entitlement for a game, the user shall receive below notification message:  **“No active user to play”**  -**“Return**” link is active  Return link shall allow user to return back to previous screen giving options to start playing or watching a game. |
| 1.1.6 | “Start playing”: Once user clicks on “Start playing”, the initial board shall be displayed to both users.  The below action links shall be displayed to both users   * **Throw Dice** * **Send Move** * **Bear Off (Remove from board)** * **Wrong Move Alert** |
| 1.1.7 | Throw Dice: Once user clicks on “**Throw Dice**”, the both user shall see the dice numbers changed on client screen. |
| 1.1.8 | Send Move: Once user clicks on “Send Move”, the playing pieces are moved according to the roll of dice. The new look of stone table is displayed to both players. |
| 1.1.9 | Wrong Move Alert: Each player has right to send Wrong Move alert right after opponent’s move.  If opponent’s move is wrong, the stone table will revert back to previous state.  If player clicked on Throw Dice or Send Move, Wrong move alert will not be enabled to the user.  Only after player clicks on “Send Move”, the opponent is able to click on “Wrong Move alert” |
| 1.1.10 | I want to watch: Once user clicks on “I want to watch” action link, the first active game session will be displayed to the user.  The stone and dice movements are send to watching user as well as players. |

* 1. Non-Functional System Requirements

|  |  |
| --- | --- |
| Requirement# | Definition |
| 2.2.1 | The system shall implement the BACKGAMMON protocol as defined in RFC2. |
| 2.2.2 | The system shall be robust, that is, especially to erroneous user input. That is, both the server and client (and especially the server), are expected to graciously handle unexpected protocol messages, protocol message fields, message field contents, unrecognized character encodings, etc. |
| 2.2.3 | The BACKGAMMON system is a client-server game system that works on the internet. |
| 2.2.4 | The system utilizes the Transmission Control Protocol (TCP) |
| 2.2.5 | The system assumes all system components are able to listen to and receive traffic from fixed port number (namely 9898). Seeing as many system components will be behind NAT-enabled routers, the testing and use of the system shall comprise port-forwarding on the routers in order to be able to listen effectively to fixed port numbers. For the purposes of the BACKGAMMON system, no other port numbers than those specified shall be used, and this set of requirements **assumes each endpoint of the system is available on a public IP and the NAT layer is fully transparent to the BACKGAMMON system** |
| 2.2.6 | In order to test requirement 2.1.4 in a stable environment, the software shall be tested on a LAN, or similar virtual private network environment. |

1. BACKGAMMON Application Server Requirements
   1. Application Server Functional Requirements

|  |  |
| --- | --- |
| Requirement # | Definition |
| 3.1.1 | The server shall listen to BACKGAMMON PC-AS protocol port for incoming traffic (TCP port 9898) |
| 3.1.2 | The server shall support clients to connect to the server |
| 3.1.3 | The server shall keep a record of all game requested users (clients) currently connected to it in a queue. |
| 3.1.4 | The server shall check if clients are alive periodically. If periodical heartbeats are not received from the client (3 heartbeats back to back), the server shall switch the status of the client to offline. In which case, the Client will be required to reconnect to start another session with the server. |
| 3.1.5 | The server shall keep record of waiting player list, active game playing sessions and connected users. |
| 3.1.6 | The server shall listen to BACKGAMMON PC-AS protocol port for incoming traffic (TCP port 9898) |
| 3.1.7 | The server shall support clients to connect to the server |
| 3.1.8 | The server shall keep a record of all game requested users (clients) currently connected to it in a queue. |

* 1. Application Server Non-Functional Requirements

|  |  |
| --- | --- |
| Requirement # | Definition |
| 3.2.1 | The server shall be available through the BACKGAMMON PC-AS protocol as defined in RFC2 |
| 3.2.2 | The system shall support up to 100 concurrent users. |
| 3.2.3 | The server shall be written in the Python programming language |

1. BACKGAMMON Client Requirements
   1. Player Client Functional Requirements

|  |  |
| --- | --- |
| Requirement # | Definition |
| 4.1.1 | The client shall use the BACKGAMMON/PC-AS protocol for traffic with application servers (TCP port 9898) |
| 4.1.2 | The client shall provide a general game playing environment to the User. |
| 4.1.3 | The client shall provide an interface for connecting to the game, playing or watching a game. |
| 4.1.4 | The client shall be able to receive response from application server corresponds to game request. |
| 4.1.5 | A Disconnected client shall close playing interface except the Welcome interface (thus blocking the user from trying to interact with the server) |

* 1. Player Client Non-Functional Requirements

|  |  |
| --- | --- |
| Requirement # | Definition |
| 4.2.1 | The client shall be available through the BACKGAMMON/PC-GS protocol as defined in RFC2 |
| 4.2.2 | The client shall be written in the Python programming language. |