Sebnem Arslan

BOSPHORUS Unıversity

1.1.2015

**BACKGAMMON Protocol**

**Status of this Memo**

This memo defines an experimental game protocol to be used by BACKGAMMON clients (Player Client) and servers (Application Server).

**Abstract**

BACKGAMMON protocol is a text-based protocol that is to be built on top of the Transmission Control Protocol (TCP). The protocol has client-to-server.

The protocol features Unicode-encoded protocol messages that are text based and are framed as TCP segments. The header of the protocol message defines the type of the message, and every message has a number of expected responses.

**Revision history**

|  |  |  |
| --- | --- | --- |
| Version | Changed By | Date |
| 1 | Initial Draft. UML State and Sequence diagrams are added. | 1.1.2015 |
|  |  |  |
|  |  |  |

Table of Contents

[**Status of this Memo** 2](#_Toc407922192)

[**Abstract** 2](#_Toc407922193)

[1 INTRODUCTION 4](#_Toc407922194)

[2 UML Diagrams 5](#_Toc407922195)

[2.1 Sequence Diagram 5](#_Toc407922196)

[2.2 State Diagram 8](#_Toc407922197)

1. INTRODUCTION

This document gives detailed information about the software design. The document includes UML State and sequence diagrams to illustrate the process flow of the program.

The BACKGAMMON protocol is inspired by the Hypertext Transmission Protocol (HTTP, RFC2616) and Internet Relay Chat Standards (IRC, mainly RFC1459). The protocol adheres to the following general rules:

* The transmission between endpoints of the protocol complies with the request-response paradigm. That is, all traffic exchanged between two endpoints will be initiated (request) by one of the endpoints and will either be acknowledged or answered (response) by the other endpoint
* The BACKGAMMON protocol is Half Duplex, that is, a requesting client is blocked until a fitting response is received. This hints that a queueing mechanism will need to be set up for incoming messages.
* Only clients of the BACKGAMMON applications send out Requests. Application Servers are designed as passive elements that issue only responses.
* Application servers are stateless. That is, although they keep and maintain the state information of the clients, they do not have multiple states that necessitate a different execution path for the same incoming request.
* Player clients are state-full, in terms of the BACKGAMMON application. That is, they may have different states which call for different paths of execution such as being “Waiting for play”, “Playing”, “Watching a game”, etc.
* To ensure request to response matching on both sender and receiver, a sequence id "username" field shall be appended to bodies of each BACKGAMMON message. A BACKGAMMON message without a "username" field is invalid.
* In almost all cases, IP addresses are not explicitly communicated between elements of the system. It is both the client and the server's responsibility to extract, store and manage IP addresses of other system elements they are in connect with. i.e. It is the responsibility of the server to figure out which client sent a certain message by working out the IP address of the message and matching it to an internal IP table
* As a backgammon notation, Paul Magriel notation is used. <http://en.wikipedia.org/wiki/Backgammon_notation>

1. UML Diagrams
   1. Sequence Diagram





* 1. State Diagram

