Relatório - O Grande Dalmuti

Pedro H. Kochinski
GRR20206144
phk20@inf.ufpr.br

Vinicius Mioto
GRR20203931
vm20@inf.ufpr.br

***Resumo*—Esse relatório tem como objetivo apresentar as principais escolhas dos autores para desenvolver o jogo de cartas chamado "The Great Dalmuti", o qual deve ser executado em uma rede em anel, devido ao caráter multijogador.**

***Palavras-chave*—rede em anel, sockets, jogo de carta.**

I. Introdução

Nesse trabalho implementamos o jogo de cartas "O Grande Dalmuti" em linguagem Python, utilizando a biblioteca socket para estabelecer a comunicação entre as máquinas. Para a execução, será necessário no mínimo 4 máquinas (jogadores) de acordo com as regras do jogo. [1]

II. Funcionamento da Rede

O jogo utiliza uma rede em forma de anel, onde os dispositivos estão conectados e usam um "bastão" (ou *token*) para coordenar a transmissão de dados. O bastão é passado sequencialmente, garantindo que apenas uma máquina possa transmitir dados de cada vez. Quando um dispositivo deseja transmitir dados, ele aguarda a chegada do bastão, assume o controle da rede, transmite os dados e, em seguida, passa o bastão para o próximo na sequência (ver figura 1).

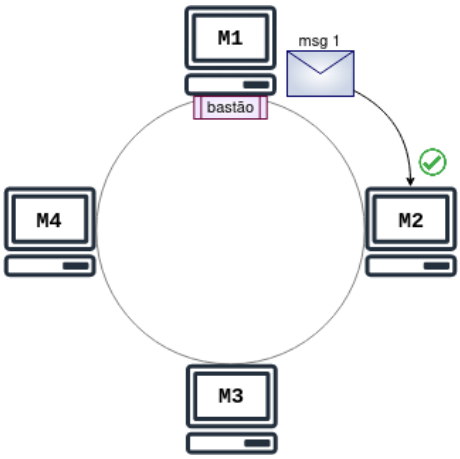

Fig. 1. Passagem de Bastão

O fluxo de execução das jogadas inicia-se com a criação da mensagem pela máquina que possui o bastão, neste caso, M1. A mensagem é enviada de M1 para M2, posteriormente de M2 para M3, e assim sucessivamente, seguindo a ordem do anel. Com isso, todos os jogadores ficam cientes da jogada executada pelo jogador 1. A última máquina deve enviar a mensagem de volta para a origem, fechando o ciclo.

Quando a máquina que realizou o envio receber a sua própria mensagem, saberemos que a jogada passou por todos os jogadores. Nesse momento, M1 deve passar o bastão para a próxima máquina, que criará uma nova mensagem com a respectiva jogada (ver figura 2).

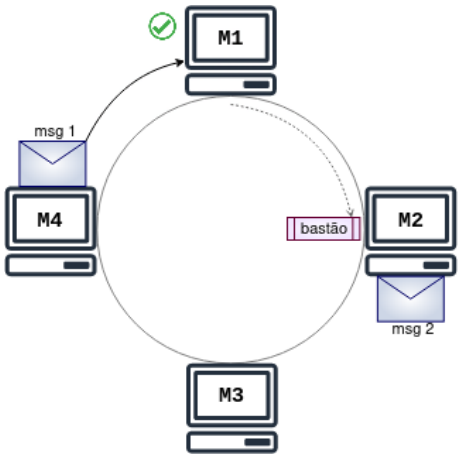

Fig. 2. Passagem de Bastão

Na implementação da rede para o jogo, não foram adicionados controles de perda de bastão ou verificações de erro nas mensagens. Assumiu-se que as mensagens foram enviadas por uma máquina deram a volta no anel com sucesso.

III. Instruções

Para iniciar o jogo, deve-se preencher as informações do arquivo `config.txt`, primeiramente com o número de jogadores e posteriormente as informações de cada máquina.

- `ADDRESS`: Endereço IP da máquina
- `SEND_ADDRESS`: IP da máquina destino
- `SEND_PORT`: Porta de envio de mensagem
- `RECV_PORT`: Porta de recebimento de mensagem
- `CLASS`: Classe do jogador

O funcionamento correto exige que a porta de envio da origem deve coincidir com a porta de recebimento do destino. Para as classes, as opções são: *Greater Dalmuti* (`GD`); *Lesser Dalmuti* (`LD`); *Lesser Peon* (`LP`); *Greater Peon* (`GP`). No caso de existirem mais n jogadores, eles deverão ocupar a classe intermediária *Merchant* e deverá preencher com `N`$\{1...n\}$.

[1] regras do jogo: The Great Dalmuti

IV. Estruturas de Dados

A comunição das máquinas exige o envio de objetos definidos pela classe `Mensagem`. Os atributos de cada novo objeto são definidos no momento da criação da mensagem, com exceção dos marcadores de início e fim, pois esses são fixos e estão definidos no método construtor. A Tabela I descreve a classe em questão.

Tabela I
Atributos da Classe Mensagem

Campo	Descrição
`init_marker`	Marcador de início: `01110`
`origin`	Endereço de origem da mensagem
`number_from`	Número da máquina que enviou a mensagem
`destiny`	Endereço de destino da mensagem
`move_info`	Informação sobre o tipo da mensagem
`receive_confirm`	Campo de confirmação de recebimento da mensagem
`end_marker`	Marcador de fim: `10001`

V. Conclusão

O projeto foi testado com os computadores dos laboratórios do Departamento de Informática da UFPR. O código fonte e o arquivo de exemplo para configuração das máquinas está em um repositório público no GitHub.