Relatório Final

Isabelle de A. Azevedo, Thales C. Castro

Curso de Bacharelado em Ciência da Computação
Universidade Federal de Pelotas (UFPel)

{idaazevedo, tccastro}@inf.ufpel.edu.br

Resumo. *Este artigo tem como objetivo descrever o desenvolvimento de uma arquitetura cliente-servidor para um jogo de Damas multiusuário implementado na linguagem Python. Abordam-se as fases e as ações da aplicação conforme o papel do dispositivo na rede, bem como os protocolos utilizados nas duas camadas superiores do modelo TCP/IP. Somando-se a isto, expõem-se os formatos das mensagens implantados na elaboração do projeto. Por fim, foram apontadas as dificuldades encontradas na construção.*

1. Introdução

O projeto desenvolvido consiste na elaboração do jogo de tabuleiro Damas por meio de uma arquitetura cliente-servidor. Esta utiliza conexão do tipo TCP (Transmission Control Protocol) na camada de transporte, para haver uma garantia de integridade das informações enviadas e recebidas entre cliente e servidor, ou seja, que propicie transferências confiáveis. Já em nível de aplicação, será definido um protocolo, assim como os formatos de mensagens, a fim de estabelecer a comunicação entre o cliente e o servidor.

Tanto as partidas quanto as funcionalidades voltadas à rede foram desenvolvidas na linguagem de programação Python, com auxílio das bibliotecas Socket e PyGame. Além disso, foi usado como base outro projeto que implementa o jogo em questão de forma exclusivamente local, do *link*: github.com/everestwitman/Pygame-Checkers.

O jogo implementado na aplicação é exclusivamente multiusuário, com dois jogadores simultâneos, cada um com uma conexão. Uma vez em funcionamento, o servidor aguarda que dois clientes se conectem para que seja autorizado o início da partida. Cada turno é computado integralmente na máquina cliente que, ao concluí-lo, envia o estado do tabuleiro para o servidor, o qual verifica se houve vitória. Em caso negativo, autoriza o começo da rodada do outro jogador. Já em caso positivo, envia o resultado e encerra as conexões.

1.1. Arquitetura Cliente-Servidor

O modelo cliente-servidor é definido como uma aplicação distribuída que divide o processamento da informação em módulos ou em processos distintos. Existem procedimentos responsáveis tanto pela manutenção da informação, quanto pela obtenção dos dados. Neste tipo de arquitetura, compreende-se o papel do servidor como um fornecedor de recursos ou de serviços, gerenciando e mantendo a informação. Já o cliente é o requerente, o qual objetiva obter os dados para uso no funcionamento da

aplicação. No trabalho a ser desenvolvido, os jogadores utilizam o *client-side* da arquitetura e um computador estará no *server-side*.

1.2. Jogo de Damas

Também conhecido apenas por Damas, é um jogo de tabuleiro notório em escala mundial. Necessita de dois jogadores e de um tabuleiro com 64 casas de cores alternadamente claras e escuras, dispondo de 12 peças para cada um. Por convenção, aquele que possui as brancas é quem começa o jogo e o objetivo é capturar todas as unidades do oponente.

A cada turno, uma peça movimenta-se em diagonal, um quadrado de cada vez e para a frente. Quando na casa adjacente houver uma pedra adversária, com uma vaga imediata no mesmo sentido, pode haver uma captura. Se após este movimento apresentarem-se as mesmas condições, realizar-se-á novamente, o que configura uma tomada em cadeia. É uma jogada obrigatória e quando há mais de um caminho possível, segue-se aquele com maior número de unidades.

Há também a possibilidade de se capturar uma peça movendo-se para trás. Todavia, para isto ser permitido, é necessário a promoção para "dama". Isto ocorre somente quando a pedra alcança a oitava casa do oponente. Este movimento é válido tanto para uma tomada em cadeia, quanto para uma jogada regular. A identificação desta unidade é feita sobrepondo-se uma sobre outra da mesma cor.

2. Manual de uso

Para utilizar a aplicação, faz-se necessário clonar o projeto em https://github.com/cineasthales/checkersnet. Além disso, é preciso certificar-se que o Python 2.7.16, o Pip e o PyGame estejam instalados no sistema operacional.

Primeiramente, executa-se o código do servidor em um janela do terminal (python2 servidor.py). Em seguida, abrir mais duas janelas (na mesma máquina ou em dispositivos diferentes na mesma rede) e abrir os programas dos clientes (python2 cliente.py endereco_ip). O "endereco_ip" é o do servidor, podendo ser localhost caso o(s) cliente(s) seja(m) testado(s) no mesmo sistema ou verificar previamente o IP daquela que executa o *server-side*.

O jogo seguirá as regras de Damas supramencionadas. Toda a vez que um jogador aguarda o seu turno, aparece uma mensagem em sua tela. Quando a partida termina, surge uma frase indicando o ganhador.

3. Protocolo da camada de aplicação

Esta seção detalha o protocolo implementado na camada de aplicação do projeto realizado. Contém as ilustrações das máquinas de estados e a descrição dos formatos de mensagens utilizados.

3.1. Máquinas de Estados

A máquina de estados do protocolo no cliente consiste em ilustrar as fases e as transições que a aplicação tem em cada dispositivo local que, para este projeto, encontra-se no papel de jogador. A ilustração está definida a seguir.

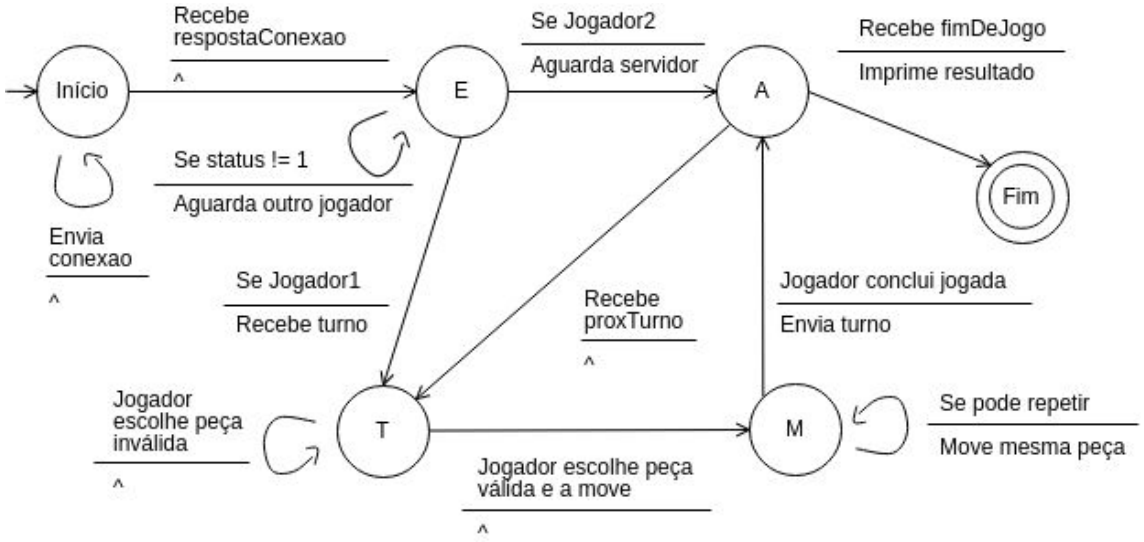

Figura 1. Máquina de estados no cliente.

Os rótulos dos estados tem os seguintes significados: Espera Início do Jogo (E), Turno (T), Movimento (M) e Aguarda Servidor (A).

Já a máquina de estados do protocolo no servidor apresenta as tarefas e as transições que o dispositivo remoto poderá executar nesta aplicação. A ilustração está definida abaixo.

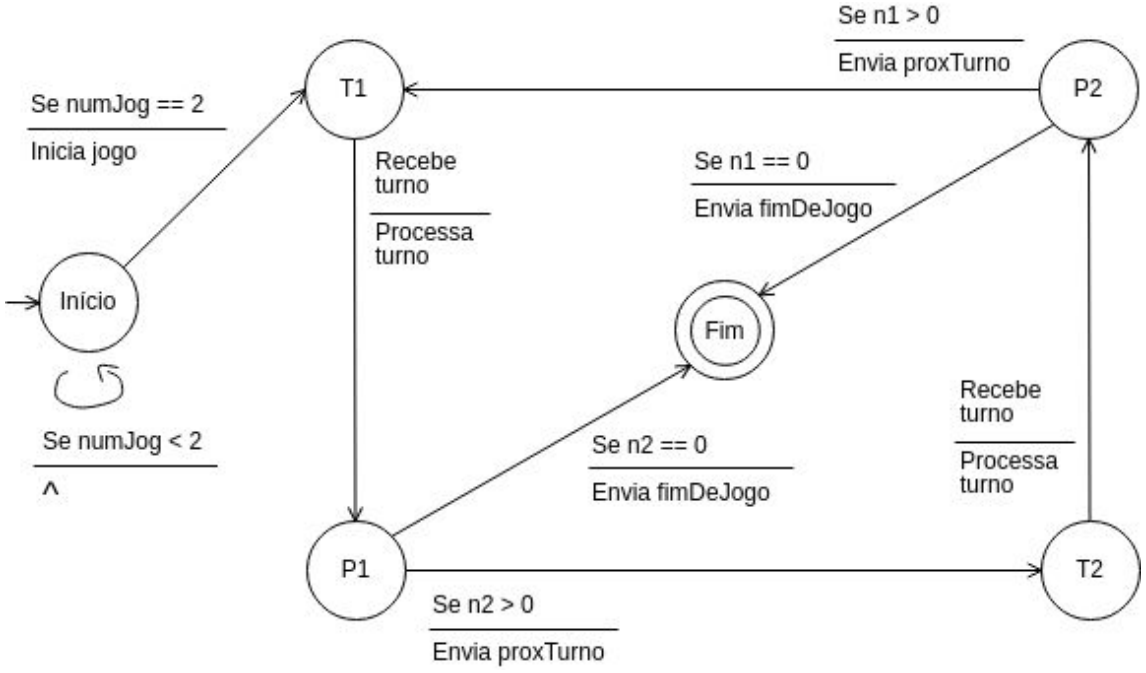

Figura 2. Máquina de estados no servidor.

Os rótulos dos estados tem os seguintes significados: Turno do Jogador 1 (T1), Processamento do Turno do Jogador 1 (J1), Turno do Jogador 2 (T2) e Processamento do Turno do Jogador 2 (J2).

3.2. Mensagens Cliente-Servidor

Inicialmente, é preciso explicar que tipos de mensagens cada cliente poderá enviar ao servidor. De forma inicial, haverá pedido para estabelecimento de conexão. Já durante a partida, há o encaminhamento do estado atual do tabuleiro e de quem realizou a jogada, cabendo aguardar o servidor transmitir as próximas informações. Ao receber o resultado final da partida, encerra-se a ligação.

Faz-se necessário explanar igualmente os modelos de mensagens que o servidor poderá transmitir. Em primeiro lugar, é preciso que confirme o estabelecimento de cada conexão e o início da partida. No desenvolvimento, após receber uma jogada, encaminha o estado atual do tabuleiro e o próximo jogador ou o resultado final, caso este exista. No último cenário, fecha a conexão.

3.2.1. Formato de Mensagens

O formato definido para cada tipo de mensagem enviada entre o cliente e o servidor e seus respectivos campos de cabeçalho, assim como os valores possíveis para estes dados podem ser visualizados na seguinte tabela.

Mensagem	Parâmetros	Aplicação
conexao	-	Cliente solicita ao servidor uma conexão.
turno	tabuleiro, jogador	Cliente informa ao servidor o estado do tabuleiro após o turno mais recente.

Figura 1. **Formato de mensagens do cliente para o servidor**

Mensagem	Parâmetros	Aplicação
conexaoAceita	-	Informa ao cliente que a conexão foi bem-sucedida.
proxTurno	tabuleiro, proxJogador	Informa estado atual do tabuleiro ao jogador do próximo turno.
fimDeJogo	ganhador	Informa fim de jogo, bem como quem ganhou.

Figura 2. **Formato de mensagens do servidor para o cliente**

4. Considerações Finais

Como este projeto baseou-se em outro que já implementa o jogo de Damas, o principal objetivo era realizar a adaptação para uma arquitetura cliente-servidor em rede local. Primeiramente, foi necessária uma pesquisa para entender como a biblioteca Socket deveria ser configurada e quais as peculiaridades sintáticas do Python2. Houve dificuldade em como efetuar as sincronizações necessárias para a realização das trocas de mensagens, contudo, foram superadas.

Referências

Bezerra. Disponível em: https://www.estudokids.com.br/como-jogar-damas/. Acesso: maio/2019.

Kurose, J. F.; Ross, K. W. Redes de Computadores e a Internet. 5. ed. São Paulo: Pearson Addison Wesley, 2010.

Python Software Foundation. Socket - Low-Level Network Interface. Disponível em: https://docs.python.org/2.7/library/socket.html. Acesso: junho/2019.