Relatório T2 Redes

Bruno Aquiles de Lima
Luan Marko Kujavski

1. Introdução

O objetivo desse trabalho é implementar o jogo chamado "foda-se" ou "fodinha" em uma rede em anel com quatro máquinas.

2. Estruturas de dados

Para implementarmos o jogo, utilizamos a linguagem Python. O nome do arquivo principal é game.py. Você pode executar o programa com a seguinte linha:

I. python3 game.py <ip_prox>

<ip_prox> é o endereço IP da máquina para a qual mensagens serão enviadas.

Após executado, é mostrado no terminal a opção de quatro jogadores. Cada jogador deve escolher um número entre 1 e 4. Não é permitido dois jogadores escolherem o mesmo número.

O jogo começa quando todos os quatro jogadores se conectam. Para isso, o jogador 1 manda continuamente uma mensagem e, enquanto não a recebe, isto é, não passou pela rede em anel, o jogo não inicia. A função em questão é a estabilish_connection().

Para facilitar a implementação do jogo, escolhemos sempre algum dos players para ser o *manager* da rodada. Quem ganha a rodada torna-se um *manager*. Se o player que se tornará *manager* morrer, passa para o próximo em sequência. É sempre o manager que começa a jogar. No início, cada player cria uma estrutura de dados do tipo RoundManager e atualiza conforme recebe informações, mas somente um player usa esta classe em cada rodada para gerenciá-la.

Além disso, cada player cria sua própria estrutura de dados Player, que será alimentada de informações conforme todos vão jogando. Ademais, foram escolhidas as portas de 1917 até 1920.

Também foi criada uma classe Deck, que guarda informações do baralho da rodada. Cada carta pertence à classe Card.

Para gerenciar os pacotes, foi criada a estrutura de dados SocketHandler que, a partir do arquivo packets.py, é responsável por mandar e receber mensagens.

3. Laço principal

O laço principal está na função main_loop(). Cada player, em sua classe Player, tem uma fila de mensagens que deseja mandar. As iterações de main_loop() retiram e colocam

elementos nessas filas. Quando um player recebe o *manager*, ele restaura sua fila para que o jogo continue.

4. Passagem do bastão

Quando a fila de mensagens de um player está vazia, ele deve passar o bastão. para isso, cada player recebe duas flags:

I. passing_baston
II. baston

Quando um player está passando o bastão, ele coloca passing_baton como True. O player que receberá o bastão, ao receber a mensagem, coloca seu baston como True e reenvia a mensagem. Somente após receber a própria mensagem, o primeiro player coloca baston como False e passing_baston como False. Isso evita que o bastão se perca caso a mensagem não chegue ao destino.

5. Broadcasting

Mensagens que servem para informar os usuários são passadas em forma de Broadcasting. Isso é feito, por exemplo, para informar uma carta jogada por um jogador ou uma carta virada para sabermos qual é a manilha.

6. Filas

Do lado esquerdo está a pilha que o manager recebe inicialmente para começar uma nova série de rodadas. Do lado direito é a pilha que o manager recebe para começar uma nova rodada. Quando uma fila acaba, o manager é trocado.