Matchmaking num jogo online

Relatório do trabalho prático de Sistemas Distribuídos

Mestrado Integrado em Engenharia Informática

Grupo 35

Ana Paula Carvalho (A61855)
Daniel Militão (A74557)
João Pires Barreira (A73831)
Rafael Costa (A61799)

Janeiro 2018

Universidade do Minho
Escola de Engenharia

1 Introdução

No âmbito da Unidade Curricular de Sistemas Distribuídos foi-nos proposto, como tema para o projecto prático, a implementação do matchmaking de um jogo online, tendo como exemplo o *Overwatch*. Neste trabalho pretende-se incluir todos os conhecimentos estudados ao longo das aulas: criação de *threads*, controlo de concorrência e conceção de servidores e clientes.

Durante a extensão deste relatório iremos clarificar as nossas decisões e o nosso código de modo demonstrar a nossa solução para o problema apresentado.

2 Servidor

Do lado do servidor, a *main* fica à espera de uma nova conexão através de um *socket TCP*, criando uma nova *OverwatchThread* para cada cliente que efetue uma ligação. Esta classe possui as seguintes estruturas de dados que são criadas pela *main* do Servidor aquando da criação da nova *thread*:

- Socket – *socket TCP* relativa à ligação do cliente ao servidor;
- Player – jogador referente ao cliente que está conectado;
- Match – partida referente ao cliente que está conectado;
- Players – conjunto dos jogadores registados no servidor;
- Matchmaking – conjunto dos jogadores que se encontram à espera de uma partida (e outras estruturas/métodos auxiliares);
- Matches – conjunto das partidas a serem realizadas;
- Notifications – mapeamento entre os jogadores do servidor e respetivos *PrintWriter*s (para envio de mensagens);
- MatchThread – nova *thread* que será responsável por inserir o jogador no sistema de *matchmaking* e realizar uma partida.

2.1 Signup

Para efetuar o registo de um jogador no servidor, é necessário receber o seu *username* e a *password*. Com estes dados, a *OverwatchThread* relativa à conexão do cliente ao servidor insere o novo jogador na lista dos jogadores do servidor, bem como no mapeamento entre os jogadores e os seus *PrintWriter*s. Por fim, envia uma mensagem "OK".

Caso o jogador já se encontrasse registado no servidor, é lançada uma exceção (*InvalidAccountException*).

2.2 Login

Para que se efetue o *login* de um cliente no servidor, é chamado um método auxiliar que verifica se o *username* e a *password* correspondem a algum utilizador registado no servidor. Em caso afirmativo, este é adicionado ao mapeamento entre os jogadores e os seus *PrintWriter*s, sendo enviada uma mensagem do tipo: "OK:1:12"(em que 1 é o *ranking* do jogador e *12* o saldo de vitórias/derrotas).

Caso os dados recebidos não correspondam a nenhum jogador registado, é lançada uma exceção (*InvalidAccountException*).

2.3 Play

Caso um jogador pretenda iniciar uma partida, a *OverwatchThread* referente a esse jogador cria uma nova *MatchThread* que se encarregará de o inserir no sistema de *Matchmaking* e, eventualmente, de jogar uma partida.

Aquando da inserção do jogador no sistema de *Matchmaking*, é adquirido o *lock* deste sistema. Depois, procurar-se-á juntá-lo com 9 outros jogadores cuja diferença de *rankings* não seja maior do que 1. Assim, poderá resultar um jogo com jogadores apenas do seu *ranking*, do seu *ranking* e *ranking* inferior ou do seu *ranking* e *ranking* superior.

Caso não haja o número necessário de jogadores para iniciar uma partida, o jogador é colocado numa estrutura auxiliar referente ao jogadores em espera (i.e. *waitingPlayers*) e a sua *MatchThread* irá ser adormecida (com um *await()*).

Em alternativa, caso o jogador consiga encontrar 9 outros jogadores compatíveis, irá criar uma nova partida com os 10 jogadores em questão. Além disso, os jogadores que se encontravam adormecidos irão ser acordados com um *signal()* pelo jogador em questão e será libertado o *lock* do *Matchmaking*.

Por fim, irá ser enviada a cada jogado uma mensagem do tipo "MATCH: Jogador1:Jogador2:Jogador3:Jogador4:Jogador5|Jogador6:Jogador7:Jogador8: Jogador9:Jogador10"a todos os clientes referentes aos jogadores da nova partida que foi criada.

2.4 Hero

Como a escolha de heróis só deve ser possível durante os primeiros 30 segundos, sempre que se cria uma partida é também criada uma *thread* do tipo *Timer*. Esta *thread* após esperar 30 segundos (através do método *sleep()*), coloca a *flag closed* da partida a verdadeiro, indicando que terminou o período disponível para a escolha dos heróis. Cada jogador fica com o último herói que escolheu e, caso não tenha escolhido nenhum, é-lhe atribuído um herói que não tenha sido escolhido por nenhum dos seus companheiros de equipa.

Depois, o *Timer* encarrega-se de gerar um resultado aleatório para a partida (vitória de uma das equipas e derrota da outra), informando os jogadores deste mesmo resultado.

Por fim, a partida e o Timer são removidos das estruturas de dados onde figuravam (*matches* e *matchThreads*, respetivamente).

Em alternativa, caso uma partida seja abortada, esta é eliminada das estruturas (bem como o *Timer* respetivo) a qualquer momento da execução (através do tratamento da exceção *InterruptedException*).

2.5 Logout

Caso um cliente efetue *logout*, é chamado um método que é responsável por limpar todos os dados referentes à sessão do jogador que agora foi terminada. Assim sendo, caso o utilizador se encontrasse a meio de uma partida (ou estivesse à procura de uma), este método interrompe a *MatchThread*, abortando a partida

e notificando os restantes utilizadores da mesma com uma mensagem do tipo "ABORTED".

Além disso, remove o jogador das estruturas relativas ao processo de *Matchmaking* e do mapeamento entre os jogadores e os seus *PrintWriter*s.

3 Cliente

Todos os clientes fazem uso da classe *OverwatchStub* que se encarrega de fazer a ligação ao servidor através de *sockets TCP*, possuindo a codificação de todos os métodos referentes às ações que os clientes podem efetuar.

3.1 Signup

É enviada uma mensagem do tipo "signup:joaquim:123"(em que "joaquim"é o *username* e "123"a *password* do jogador).

Caso não seja recebida uma resposta do tipo "OK", é lançada uma exceção *InvalidAccountException* que indica que já existe no servidor um jogador registado com os dados indicados.

3.2 Login

É enviada uma mensagem do tipo "login:joaquim:123"(em que "joaquim"é o *username* e "123"a *password* do jogador).

Caso não seja recebida uma resposta do tipo "OK", é lançada uma exceção *InvalidAccountException* que indica que não existe no servidor um jogador registado com os dados indicados.

3.3 Play

Caso um jogador pretenda iniciar uma partida, o método *play()* de *OverwatchStub* encarrega-se de enviar ao servidor uma mensagem "play", esperando receber uma indicação sob a forma de uma mensagem do tipo "MATCH:Jogador1:Jogador2:Jogador3:Jogador4:Jogador5|Jogador6:Jogador7:Jogador8:Jogador9:Jogador10"com as informações da partida.

Após este momento (e por 30 segundos) passará a ser possível ao cliente escolher o seu herói através do método *hero()* de *OverwatchStub*. Passado esse período de tempo (e caso não hajam desconexões) é recebida uma mensagem contendo "VICTORY!"ou "DEFEAT..."com o resultado da partida (gerado aleatoriamente).

3.4 Hero

Para escolher um herói é necessário enviar uma mensagem ao servidor contendo "hero:Junkrat"(em que "Junkrat"é o nome de um dos 26 heróis existentes no Overwatch). De seguida, é recebida uma mensagem contendo "OK:Joaquim:

Junkrat"caso não haja nenhum elemento da equipa com o mesmo herói e contendo "ERROR:Joaquim:Junkrat"caso contrário.

No entanto, é necessário que o jogo não tenha sido terminado devido a alguma desconexão nem que tenham passado os 30 segundos iniciais da partida para a escolha dos heróis.

3.5 Logout

Neste caso é apenas enviada uma mensagem contendo "logout"ao servidor através do *socket TCP*.

4 Conclusão

Com o desenvolvimento deste trabalho prático conseguimos ter um conhecimento mais sólido e uma maior destreza na utilização de *threads*, a capacidade de entender claramente como gerir concorrência e acesso a dados e uma visão geral de como as ligações entre um servidor e os vários clientes que a ele se querem conectar funcionam na vida real.

Do ponto de vista crítico, podemos referir que o grupo optou por implementar uma interface gráfica sem termos muito tempo para o fazer pelo que o código relativo a esta parte não ficou como queríamos. No entanto, achamos que no geral, fizemos um bom trabalho a todos os níveis, cumprindo a totalidade dos requisitos apresentados.

Assim sendo, tendo possibilidade de melhorar o trabalho, o grupo tentaria melhorar a interface gráfica, bem como comentar o código com mais exaustão.