Projecto Recurso

Licenciatura em Engenharia Informática - Computação Distribuída

Distributed Black Jack

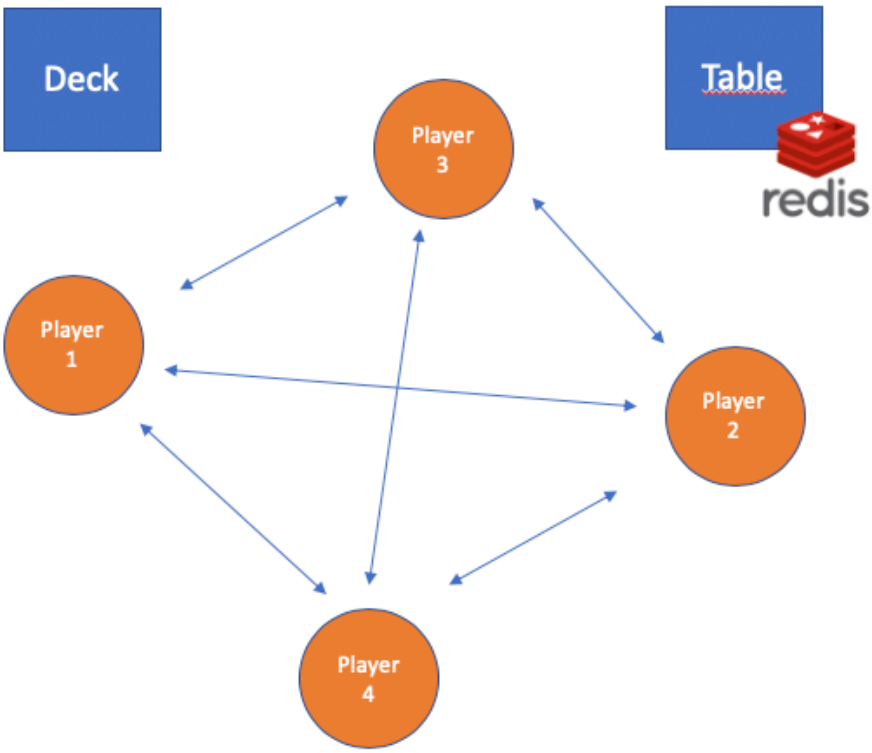
Docentes:
- Diogo Gomes (dgomes@ua.pt)
- Nuno Lau (nunolau@ua.pt)

Prazo:
23 de Julho – 24h00

Conceitos a abordar:

- Sockets
- Marshalling
- P2P
- Fault Tolerance

Introdução



Uns amigos em confinamento resolveram desenvolver um jogo P2P de black jack. Como estão em confinamento o jogo é 100% distribuído e não há casa/dealer, existe apenas um servidor central com os baralhos de cartas (infinitos) que atende um jogador de cada vez. Os jogadores competem entre si.

Os jogadores de black jack começam por tirar 2 cartas. Posteriormente por turnos tiram uma carta até se aproximarem do valor 21, sendo que o "Às" pode valer 1 ou 11, as figuras valem 10 e as restantes cartas o valor de rosto. Quando um jogador se encontra perto de 21 sem ultrapassar pode passar a sua vez a outro jogador. Um jogador que tenha 21 ganha imediatamente e um jogador que tenha mais do que 21 perde automaticamente ("bust"). O jogo pressupõe que as cartas de cada um não são conhecidas pelos adversários e que os jogadores devem informar os demais ganharam/perderam. O primeiro a fazer 21 ou o último a fazer "bust" é o vencedor do jogo.

Para esta tarefa é fornecido este guião e um código seminal disponível em https://github.com/detiuaveiro/CD_black_jack. Devem criar o vosso projecto no GitHub Classroom usando https://classroom.github.com/a/NTw48r4V.

Objectivos

Este é um trabalho aberto em que terão que desenvolver um **protocolo P2P** que permita aos amigos jogar uma partida de black jack. No início do jogo os jogadores devem acordar uma ordem e cada um à vez seguindo essa ordem pode contactar o "deck" para obter uma carta.

Cada jogador deve informar os demais que: "jogou", "ganhou", "perdeu".

Todos jogadores devem chegar a um consenso sobre quem foi o vencedor da partida. Para chegar a este consenso os jogadores devem apenas neste momento partilhar as suas cartas com os restantes jogadores.

Deve desenvolver o ficheiro "player.py" para criar o agente "bem-comportado"

Há sempre a possibilidade de um jogador fazer batota e mentir sobre as suas cartas. Deve desenvolver um agente "malcomportado" "bad_player.py" com base no código de "player.py" que tem as seguintes opções:

- Tirar uma carta a mais
- Mentir sobre ter ganho e/ou o valor das suas cartas.
- Mentir sobre o valor de hash obtido do deck

Para ajudar na tarefa de encontrar batoteiros o "deck" fornece um md5sum da lista de cartas distribuídas até ao momento. Esta é, no entanto, uma operação com custo elevado e apenas pode ser executada 2 vezes por jogo.

O código do "deck.py" não pode ser alterado, e obedece ao seguinte protocolo request-response em plain text:

"GC" -> devolve 2 chars e um \n correspondendo ao valor de cara de uma carta

"HC" -> devolve 32 chars e um \n com o md5sum da lista de strings com valor de cara das cartas já distribuídas.

Depois de removerem uma carta os jogadores devem colocar a carta na mesa. A mesa é neste projecto implementada através de um servidor REDIS que vai guardar para cada jogador (chave é o seu ID/porto) a lista de cartas.

Desenrolar do jogo:

- Todos jogadores sabem quantos jogadores existem e as suas portas (passadas por argumento)
- Inicia o jogo o jogador com porta inferior, mas o vosso protocolo pode encontrar outra ordem.
- Na primeira jogado todos devem tirar 2 cartas
- Cada jogador tira uma carta no seu turno (ou dá a vez)
- Quando um jogador atingir ou exceder 21, deve avisar todos os demais

- Ganha o primeiro a atingir 21 ou o último em jogo

Averiguação do vencedor:

No final do jogo os jogadores devem eleger 2 jogadores entre si para irem ao "deck.py" pedirem a Hash das cartas em jogo. Com base nessa informação deverão poder chegar a um consenso sobre o resultado final do jogo (se houve batota ou não)

Avaliação

A avaliação deste trabalho será feita através da submissão de código na plataforma GitHub classroom e de um relatório em formato PDF com não mais de 5 páginas colocado no mesmo repositório junto com o código e com o nome `relatorio.pdf`.

Está em avaliação o protocolo definido e documentado, assim como as *features* implementadas de acordo com os objectivos incrementais:

- player.py consegue jogar jogo solitário
- 2 player.py conseguem jogar um jogo
- 3 player.py conseguem jogar um jogo
- "bad_player" implementado no mínimo com as opções anteriormente mencionadas
- Restantes jogadores são capazes de detectar que houve batota no jogo.

Referências

[1] https://redis-py.readthedocs.io/en/stable/