

## Sistemas Distribuídos

Exame (Época Especial)<sup>1</sup>

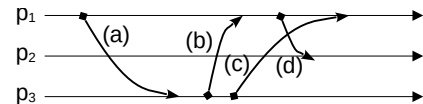
16 de julho de 2024

Duração: 2h00m

### I

1 Explique porque é que o algoritmo de exclusão mútua de *filtros* usa um número preciso de camadas dependente do número de *threads*. O que aconteceria se usasse mais? Ou menos?

2 Um programa distribuído que usa relógios lógicos tem um *bug*. Recolheu-se o registo de mensagens trocadas mostrado ao lado em que os valores trocados nas mensagens são: (a) 2; (b) 5; (c) 4; (d) 5. Explique qual é o problema.



3 Explique como é que o algoritmo de 2PC (*two-phase commit*) lida com a falha de um dos participantes durante a primeira fase.

4 Considere um sistema de domótica com um conjunto de dispositivos (sensores, luzes, interruptores, ...) ligados a uma rede WiFi e que podem ser consultados e atuados através de um serviço remoto na *cloud*. De forma a poupar energia, seria conveniente que só um dos dispositivos mantenha ligação à *cloud*, servindo de intermediário para os restantes. Identifique aquele que lhe parece ser o problema de sistemas distribuídos mais importante neste caso e proponha, justificando, uma solução típica.

### II

Considere um sistema cliente/servidor para permitir jogar o jogo *pedra, papel e tesoura*.

1 Apresente uma classe Java (para ser usada no servidor) que implemente a interface abaixo que representa uma partida deste jogo. Tenha em conta que os seus métodos serão invocados por duas *threads*, representando os dois jogadores (A ou B).

```
interface PPT {
    String jogaA(String jogada) throws InterruptedException;
    String jogaB(String jogada) throws InterruptedException;
    void desisteA();
    void desisteB();
}
```

Os métodos *joga\** devem bloquear até ambos os jogadores terem feito uma jogada, invocando estes métodos. Cada jogada pode ser Pedra, Papel, ou Tesoura. Devem retornar a cada jogador o resultado da jogada, que será *Empate*, se ambos jogaram o mesmo, ou *Ganhou/Perdeu* caso contrário, segundo as regras: pedra ganha a tesoura, tesoura ganha a papel, e papel ganha a pedra. Um método *joga\** pode ainda retornar, imediatamente, *Ganhou* se o outro jogador invocar *desiste\**.

**Simplificação:** (max. 60% da cotação) Considere apenas uma jogada.

2 Considere um servidor onde jogadores se ligam por TCP, sendo emparelhados para jogarem uma partida. Cada cliente poderá enviar jogadas (Pedra, Papel, ou Tesoura), e deverá receber do servidor a resposta adequada. Se a resposta for *Empate*, os clientes continuam a fazer mais jogadas, até a resposta ser *Ganhou* ou *Perdeu*. Um cliente pode ainda desistir a meio, em vez de fazer mais uma jogada, enviando *Desisto*. Fechar a conexão conta como uma desistência.

Implemente só o programa servidor usando *threads*, *sockets* TCP, e a classe desenvolvida na pergunta anterior. Use um protocolo o mais simples possível, por exemplo, baseado em linhas de texto.

**Simplificação:** (max. 80% da cotação) Considere apenas dois jogadores.

<sup>1</sup>Cotação — (2, 5 × 4) + (6 + 4)