# Servidores Tolerantes a Faltas para o jogo 4Pong

Pedro Nunes
Departamento de Informática
Faculdade de Ciências de Lisboa
i29155@alunos.di.fc.ul.pt

**Grupo TFD016**

Fernando André
Departamento de Informática
Faculdade de Ciências de Lisboa
i29040@alunos.di.fc.ul.pt

## Abstract

*O objectivo deste projecto é criar um jogo multi-jogador, que possuirá no máximo quatro jogadores e que correrá num ambiente distribuído. A ideia é que existindo um ou mais servidores do mesmo, o sistema seja tolerante a faltas; ocorrendo uma falha num servidor o jogo deverá continuar a correr sem qualquer tipo de problema para os clientes (jogadores). Uma possível concretização do mesmo é apresentada neste artigo.*

## 1. Introdução

Um Sistema Distribuído definido por Tanenbaum é uma colecção de computadores independentes que se apresenta ao usuário como um sistema único e consistente; outra definição possível, seria uma colecção de computadores autónomos interligados através de uma rede de computadores e equipados com software que permita o compartilhamento dos recursos do sistema: hardware, software e dados.

Para que os recursos existentes num sistema distribuído estejam sempre disponíveis, foram desenvolvidas várias técnicas que permitem o sistema continuar operacional após uma falha. Entre elas, a que mais se destaca para a realização o projecto, é o modelo de replicação. Este modelo consiste em uma máquina que por qualquer motivo fica impossibilitada de fornecer um dado serviço, ser substituída por outra que também o faz. É um modelo transparente para os utilizadores pois estes podem estar ligados a máquinas que nem sabem que existem.

No projecto a realizar, o jogo Pong, existirão quatro jogadores no máximo, onde cada um terá ao seu controlo um paddle com o qual deve evitar que a bola embata na sua parede. A bola circula dentro do tabuleiro de jogo de modo linear, apenas alterando a sua rota quando embate no paddle ou na parede de um jogador. Os jogadores que não conseguem evitar a colisão, vão somando pontos e no final do jogo ganha quem tiver menos pontuação.

Este jogo embora simples, tem aqui como objectivo por em prática a replicação de servidores de modo ao clientes que se encontrem a jogar, não interrompam o jogo caso algum destes servidores deixe de responder. O artigo encontra-se dividido nas seguintes secções; 2-A Comunicação; 2.1-APPIA; 2.2-Eleição do Coordenador; 3-Conclusão; 3.1-Trabalho Futuro; 4-Bibliografia.

## 2. Comunicação

A comunicação entre o cliente e o servidor será realizada através do protocolo TCP, podendo assim usufruir das vantagens do mesmo em relação a UDP. Existirá assim garantia da ordem das mensagens enviadas assim como da sua entrega. Será mantida uma lista das mensagens enviadas para a eventualidade do servidor a que um cliente se encontra ligado falhar, para ser assim possível a outro servidor (uma réplica existente) continuar o serviço ao mesmo cliente sem este sequer notar algum tipo de alteração. Na situação de um cliente falhar, o servidor manterá o jogo activo para os restantes jogadores, imobilizando apenas o paddle do jogador e dando baixa deste.

Os servidores comunicarão entre eles através da comunicação em grupo, utilizando o protocolo UDP que não garante ordem nem fiabilidade na entrega de mensagens. Esta opção tem como principal vantagem uma menor sobrecarga da rede visto os servidores em grupo terem de actualizar constantemente as suas vistas. Aqui reside o núcleo da tolerância a faltas do tipo replicação, onde um servidor está replicado por um ou mais servidores. A actualização das vistas serve precisamente para o caso de um servidor falhar, uma réplica poder substitui-lo e continuar o seu trabalho precisamente a partir do ponto onde este falhou.

### 2.1. APPIA

O grupo dos servidores replicados será gerido por uma camada desenvolvida através da framework de comunicação por camadas implementada em java, o APPIA e inserida na pilha de protocolos inicial fornecida com

o jogo. Inicialemente existem apenas duas camadas para o cliente e para o servidor; PongClient utilizada pelo cliente para tratar as mensagens recebidas por um servidor e pela interface e a camada TcpCompleteLayer para enviar as mensagens para o servidor com garantia de entrega. Do lado do servidor existe a camada PongServerLayer para tratar das mensagens dos clientes e a camada TcpCompleteLayer para enviar as mensagens para os clientes.

As camadas a desenvolver irão tornar o sistema tolerante a faltas, tratando de eventos como a mudança de cliente para outra réplica no caso do cliente e no caso do servidor possibilitar a sincronia em grupo, gestão de membros do grupo e do próprio grupo e também detectar e gerir falhas.

### 2.2. Eleição do Coordenador

A eleição de um coordenador é necessária quando o existente por qualquer motivo deixa de poder continuar a servir os clientes. O modelo de faltas que se considera, é o 'fail-stop' onde um servidor ao falhar não recupera e a falha é detectada. Para facilitar a sua implementação o coordenador escolhido será o próximo existente na lista de réplicas. As regras de eleição deste poderiam ser mais complexas do que a solução apresentada, tendo em conta muitos aspectos importantes num sistema distribuído (carga das réplicas, distância das mesmas, etc.), o que daria um sistema mais equilibrado, mas por não ser um ponto fulcral nesta situação, optou-se pela solução anterior.

## 3. Conclusão

Através do uso de replicação, é possível optimizar um sistema distribuído, tornando-o tolerante a faltas e dar uma boa qualidade de serviço, neste caso, à arquitectura cliente-servidor vastamente usada hoje em dia em vários jogos e aplicações. O principal objectivo é manter um dado serviço sempre acessível, mesmo que o seu servidor falhe, recorrendo a réplicas deste. Este jogo pretende por em prática este tipo de tolerância a faltas, de modo a que os jogadores possam sempre continuar a jogar mesmo que o servidor a que se encontram ligados falhe. Nesta situação se isso acontecesse não existiriam problemas de maior relevância do que que simplesmente os jogadores não poderem terminar o jogo, mas existem situações em que a falha de um servidor pode trazer danos irreversíveis. Pretende-se apenas mostrar a importância dos sistemas tolerantes a faltas de um modo global.

### 3.1. Trabalho Futuro

Como já referido anteriormente, na parte de escolha de um coordenador, o projecto não tem como objectivo a melhor solução mas sim uma solução. Isto reflecte-se um pouco ao longo de todo o processo de implementação porque dadas as circunstâncias que em este irá ser realizado, devido sobretudo à falta de tempo e também a alguns conceitos importantes deixados de lado por ser um projecto pequeno, este não está preparado para lidar com alguns aspectos importantes existentes num sitema distribuído de dimensão considerável. Por exemplo, se viesse a ser utilizado em redes públicas não teríamos apenas falhas internas mas estaríamos também que lidar com intrusões externas ao sistema. Futuramente e com uma maior capacidade de dedicação a um sistema deste tipo, poderiam então ser implementadas novas funções para que este sistema podesse ser utlizado num ambiente em que realmente não poderiam existir qualquer tipo de falhas.

## 4. Bibliografia

[1] Veríssimo, P. Rodrigues, L. (2001) Distributed Systems for System Architects

[2] URL: http://appia.di.fc.ul.pt

[3] Pinto, Alexandre Appia Group Communication Manual, 2001