Tolerância a Faltas Distribuída - Artigo Preliminar do Projecto Comunicação em Grupo com Servidores Replicados

Helga Nunes Marques - i25292
hmarques@fc.ul.pt

Nuno E. V. Charrama - i25280
nunovc@yahoo.com

Pedro Manuel Santos - i25893
pmsantos@fc.ul.pt

October 20, 2003

Abstract

Para conferir uma maior fiabilidade a sistemas distribuídos, foram desenvolvidos mecanismos de tolerância a faltas. A construção de um jogo multi-utilizador recorrendo a servidores replicados, é uma ilustração possível de um destes mecanismos.
Este artigo descreve uma sua concretização.

1 Introdução

O modelo de computacao cliente / servidor (comunicação por RPCs), actualmente o mais utilizado, necessita do apoio de vários mecanismos de tolerância a faltas, de modo a melhorar o seu desempenho [2].
Como se pode imaginar, não existe um único mecanismo que resolva todos os problemas. É necessário encontrar um conjunto que se adeque aos requisitos do serviço e ao tipo de faltas que se pretende tolerar.
Para o jogo a implementar, pretende-se que o sistema sobreviva a falhas de clientes e/ou servidores (está subjacente que o jogo se mantém activo aquando da saída de um cliente ou crash de um dos servidores). Como meio para alcançar este objectivo, é usada uma plataforma de suporte de comunicação em grupo, o Appia. O grupo referido é composto por servidores replicados, que ordenam e difundem entre si os pedidos recebidos.
Este documento está estruturado da forma apresentada em seguida. Na secção 2, é apresentado o jogo de um modo geral, acompanhado das suas respectivas regras. A secção 3 mostra de uma forma abstracta a arquitectura do sistema. A secção 4 detalha a concretização das funcionalidades e objectivos do mesmo, entre elas a replicação, ordenação e reintegração. Na secção 5 encontra-se uma breve conclusão.

2 O Jogo - Pong

O objectivo do trabalho é criar um jogo multi-utilizador bastante simples. O jogo é jogado num campo quadrado com quatro paredes, uma por jogador. Existe uma bola que se move no campo, fazendo ricochete quando bate nas paredes que limitam o campo [5].

2.1 Regras

Cada jogador possui um cursor que se pode deslocar lateralmente de modo a tentar evitar que a bola bata na sua parede. Sempre que a bola bata numa parede (o cursor não interceptou a bola), é acrescentado um golo ao jogador que defende essa parede. Apesar do golo, o jogo não é interrompido, pelo que a bola mantém a sua trajectória.
O jogo termina no momento em que um dos jogadores atinja um número máximo de golos. Ganha quem tiver menos golos sofridos. Em caso de empate, é declarado mais do que um vencedor.

3 Arquitectura do Sistema

A infra-estrutura do jogo apresenta uma decomposição cliente / servidor.
Agindo como cliente temos os jogadores que tentam defender a sua baliza. Cada movimento efectuado é um evento que é transmitido a um servidor através de comunicação ponto-a-ponto. Um servidor que receba uma mensagem está encarregue de a difundir para todos os outros, usando comunicação em grupo, de modo a manter a coerência do estado de jogo entre eles.
O movimento da bola nos clientes é actualizado através de mensagens enviadas periodicamente pelos servidores (o estado actualizado do jogo é mantido nos servidores).
O sistema, além de tolerar o crash de servidores (através da filiação de grupos), suporta também a desconexão e (re)integração de clientes.

4 Concretização

A fim de implementar as funcionalidades requeridas, é necessário recorrer a várias técnicas de programação em sistemas distribuídos.
Para a concretização desta aplicação, o Appia [4] é a ferramenta escolhida. Este consiste numa pilha de camadas que definem micro-protocolos, que assistem a comunicação em grupo. Cada um dos protocolos é responsável por fornecer um serviço (exemplo: ordenação FIFO, UDP).

4.1 Replicação

A replicação é conseguida através da criação de um grupo de servidores, cujo objectivo é tolerar a falta de elementos individuais sem prejudicar o funcionamento normal da aplicação (excepto no caso em que não existe nenhum servidor disponível).
Para os clientes, a existência de um grupo de servidores é transparente, pois ele só necessita de contactar um deles (não necessariamente sempre o mesmo).

4.1.1 Replicação Activa vs. Passiva

Existem duas formas de replicação [1], cada uma delas com as suas vantagens e desvantagens. No sistema que se pretende desenvolver, foi considerado que a mais vantajosa é a **replicação activa**, em detrimento da **replicação passiva**. Como explicação para esta decisão, apontamos o facto de que na replicação passiva existe a distinção entre um servidor primário e os servidores secundários. Somente o servidor primário tem a responsabilidade de manter o estado actual do sistema, actualizando-o nos servidores secundários periodicamente, e responder aos pedidos dos clientes. Quando o servidor primário falha, torna-se necessário eleger um novo. Este processo faz com que o sistema tenha uma ligeira paragem que poderá ser perceptível para os utilizadores. Pode até acontecer que o sistema tenha um retrocesso de estado, caso o último estado disponível no servidor eleito não seja igual ao que o servidor primário mantinha aquando da falha.
Na replicação activa, não existe distinção entre servidores, pois todos eles devem manter o mesmo estado e estar disponíveis para a recepção e resolução de pedidos do cliente (a partir do momento em o cliente recebe uma resposta, descarta todas as repetidas). Deste modo, não existe o perigo de um bloqueio momentâneo da aplicação. Como inconveniente, o número de mensagens na rede é maior, mas nunca o suficiente para provocar atrasos na entrega destas.

4.2 Ordenação

O método de replicação escolhido introduz mais uma particularidade(complicação?) ao sistema pois torna-se necessário, para que todos os servidores mantenham o mesmo estado, que recebam as mesmas mensagens em ordem idêntica, garantindo que o sistema evolui da mesma forma em cada um deles. Como tal, a ordenação total de mensagens é essencial ao bom funcionamento. Esta ordenação de mensagens é conseguida recorrendo a camadas especializadas do Appia.

4.3 Reintegração

Tal como é possível um servidor sofrer um crash, é normal que ele recupere e pretenda reintegrar-se de volta no grupo, de modo a retomar a sua execução.
À partida, os serviços de comunicação em grupo concretizados pelo Appia facilitam a reintegração de servidores. Tendo em conta a aplicação em causa, quando um servidor é reintegrado, é entregue uma nova vista

[3]. Para facilitar a integração de um servidor, é inutilizada toda a informação que este poderá possuir. Desta forma, não há necessidade de distinguir entre integração e reintegração, sendo apenas necessário entregar-lhe o estado actual do jogo.
Quando um cliente sofre um crash, o jogo mantém-se com menos um jogador. No que diz respeito à sua reintegração, toda a informação anterior desaparece, sendo considerado que entrou um novo jogador (a pontuação vem a zero).

5 Conclusão

Este artigo teve como objectivo explicitar a implementação de um jogo multi-utilizador em ambiente de computação distribuída, utilizando replicação activa de servidores para atingir tolerância a faltas. Referimos também que para realizar esta forma de replicação, é necessário trabalhar sob um protocolo de ordenação total de mensagens.

References

[1] Paulo Veríssimo and Luís Rodrigues. Distributed System for System Architects. Kluwer Academic Publishers, 2001. ISBN 0-7923-7266-2

[2] Rachid Guerraoui and Luís Rodrigues. Abstractions for Distributed Programming. Capítulos 1, 2, 3.

[3] Pedro Vicente e João Martins. Arquitectura de um Sistema de Chamadas a Procedimentos Remotos a Servidores Replicados.

[4] http://appia.di.fc.ul.pt

[5] http://www.di.fc.ul.pt/ ler/docencia/tfd