TFD 2005/2006
Pong Multi-Utilizador Tolerante a Faltas

Nuno Gaspar
(gaspaxo@sapo.pt)

Rui Rodrigues
(rui_pedro_r@netcabo.pt)

October 25, 2005

Abstract

Este artigo tem como objectivo apresentar uma proposta para a solução dos vários problemas inerentes à replicação de servidores em sistemas distribuídos. Neste caso o sistema é uma implementação do popular jogo de vídeo Pong, numa versão multi-jogador, seguindo o modelo cliente-servidor, com replicação de servidores e tolerante a faltas. A sua implementação será apoiada na ferramenta Appia, ferramenta essa desenvolvida em Java, e que é fornecida pelo docente da cadeira. Esta ferramenta fornece numerosos mecanismos de comunicação, mas destes serão apenas referidos no corpo deste artigo aqueles aos quais recorreremos para concretizar a solução por nós idealizada.

1 Introdução

No universo da informática e com o alcance destes sistemas e suas aplicações a extender-se globalmente, o desenvolvimento de aplicações baseadas em sistemas distribuídos aparece como uma solução para o problema da disponibilidade da informação. Para isso recorre-se a um paradigma conhecido como replicação de informação cujo principio básico é copiar informação crítica para várias máquinas de modo a que, se uma delas falhar, não haja perda na continuidade do serviço.

Visto isto, e tendo como objectivo construir uma versão distribuída do jogo de vídeo Pong com suporte para vários jogadores, pretendemos desenvolver uma arquitectura baseada num sistema cliente-servidor, replicando este último. Nesta arquitectura os clientes apenas terão a função de desenhar o ambiente gráfico correspondente ao estado actual do jogo, enquanto que o servidor principal, designado de Coordenador, calculará o estado do jogo num determinado momento. A este estado chamaremos o estado actual de jogo e é esta a informação crítica que queremos replicar pelos diversos servidores existentes. Desta forma conseguimos a tolerância a faltas do lado do servidor porque, no caso do Coordenador falhar, um elemento do grupo de servidores replicados tomará o seu papel com a máxima transparência para os clientes.

2 O jogo

É geralmente aceite que o Pong foi o primeiro jogo de vídeo de que há conhecimento. Trata-se de uma jogo simples, em cuja vertente original dois jogadores se debatem num jogo de ténis virtual simplificado, em que o objectivo é defender o próprio lado do ecrã (baliza), tentando atingir o lado do adversário.

A vertente a implementar neste projecto será uma versão para quatro jogadores em simultâneo, jogando cada um na sua aplicação cliente, sendo o campo um quadrado cujos lados são defendidos por um jogador cada um. Assim, o jogador pontua se a bola colidir no lado do quadrado oposto ao seu, e sofre um "golo" se a bola bater no seu lado do quadrado. A implementação será, como o Appia[1], feita em Java, tendo como base um código "tem-

plate" (cujo cliente se encontra já quase todo implementado), disponibilizado pelo docente.

3 Problemas e Implementação

Existem vários problemas que podem surgir numa aplicação distribuída deste género. Procuraremos explicar os que consideramos principais neste ponto, apresentando, como implementação, as soluções que propomos.

Como se trata de uma aplicação multi-servidor, o conjunto de servidores será tratado como um grupo - para tal usaremos a camada do Appia para comunicação em grupo. Para manter a simplicidade, os eventos principais no jogo serão decididos por um servidor central, um Coordenador, que os propaga para os clientes e para os outros servidores. Estes eventos são: a bola colide em algum obstáculo e muda de direcção, a bola colide num dos lados do quadrado e conta como golo e um jogador moveu o seu paddle. Para além disto, será determinado um timeout ao fim do qual o Coordenador propagará pelas outras réplicas e pelos clientes o estado do jogo (que inclui as posições dos jogadores e da bola, a direcção desta e os pontos). Como tal, para o jogo correr sem inconsistências, é necessário manter as ordens causal e total dentro do grupo, para além de termos de garantir a entrega de todas as mensagens a todas as réplicas - o que é também conseguido graças a protocolos de *reliable broadcast*[3], implementados na prÃ³pria camada para comunicação em grupos do Appia, em cima de *TCP*[3]. Já a comunicação cliente-Coordenador mantém a consistência se a basearmos na camada *nakFIFO*[3] em *UDP*[3], para além de que este protocolo é relativamente eficiente para aplicações que requiram actualização frequente.

Caso o Coordenador falhe, existe, ainda na camada para grupos, uma aproximação ao "detector de falhas", o *suspect*[3], que o grupo usará para decidir quando o Coordenador falha. Aqui, a decisão é tomada sem recorrer a votação[2], de modo a diminuir o tráfego na rede e o tempo dispendido, através de um concenso com base na ordem de id de cada servidor: como todas as réplicas fazem parte do mesmo grupo, cada servidor sabe o id dos outros servidores. Assim, o Coordenador será sempre o servidor com o id mais baixo. Mas esta situação traz ainda o problema da ligação dos clientes ao Coordenador: como a ligação cliente-Coordenador é *point-to-point*[3] simples (os clientes não fazem parte do grupo), não sabem a que outros servidores se ligar. Este problema pode ser resolvido recorrendo à propagação de uma *remote view* simples pelos clientes quando algum servidor falha ou entra no grupo, de modo a que os primeiros saibam a que servidor se ligar caso o Coordenador actual falhe.

Appia - principais protocolos a usar[3]:

```
appia.protocols.updsimple
appia.protocols.tcpcomplete
appia.protocols.group:
                      bottom
                      inter
                      intra
                      leave
                      stable
                      suspect
                      vsync
appia.protocols.nakfifo.NakFifoLayers
appia.protocols.totalFastAb
appia.protocols.causal
```

4 Conclusão

Este artigo apresenta uma aproximação preliminar ao projecto proposto na cadeira, visando propor uma solução simples a alguns dos problemas específicos de uma aplicação deste tipo. Embora o Pong se trate de um jogo simples, com relativamente pouca informação inerente ao estado do jogo, é um bom ponto de partida para análise do paradigma da replicação de servidores em sistemas distribuídos. O plano de implementação sugerido é suficientemente flexível a alterações que possam surgir no decorrer da fase de implementação, sem no entanto prevermos grandes

alterações nos conceitos principais e no *modus operandi*.

5 Bibliografia

[1] Hugo Miranda Alexandre Pinto Nuno Carvalho Luís Rodrigues, *"Appia Protocol Development Manual"*, version 2.0

[2] Rachid Guerraoui and Luís Rodrigues, *"Introduction to Reliable Distributed Programming (Preliminary Draft)"*

[3] Appia Documentation: *http://appia.di.fc.ul.pt/docs/javadoc/*