Um Pong Tolerante a Faltas

André Luís
Departamento de Informática
andr3@alunos.di.fc.ul.pt

Ivo Guimarães
Departamento de Informática
i30348@alunos.di.fc.ul.pt

24 de Outubro de 2005

Abstract

A área dos Sistemas Distribuídos foi, nos últimos anos, das áreas da informática que mais evoluiu e fez evoluir a sociedade. Com o surgimento de sistemas cada vez mais complexos, em conjunto com uma necessidade cada vez maior de garantir a qualidade do serviço fornecido, as falhas nos sistemas informáticos começaram a ser tidas em conta no desenvolvimento de aplicações visto que as suas consequências são cada vez mais prejudiciais. Para minimizar ao máximo estas consequências surgiu uma disciplina dentro da área de Sistemas Distribuídos que visa o empreendimento de técnicas de tolerância a faltas. Este artigo destina-se a documentar as técnicas usadas para garantir que mesmo numa arquitectura cliente-servidor, na ocorrência de uma falha por parte do servidor, o serviço que este fornece sofre poucas ou nenhumas alterações.

1 Motivação

Na sociedade deste novo milénio há uma crescente necessidade de sistemas informáticos fiáveis, dado que cada vez mais vivemos numa sociedade de informação onde esta precisa de ser propagada com confiança. Para além disto, há sistemas informáticos de grande porte que necessitam de garantir fiabilidade sob pena de avultados prejuízos ou, em alguns casos extremos, perda de vidas humanas. Neste artigo vamos estudar um caso prático, em que um sistema recupera de falhas e garante uma qualidade de serviço (QoS)[1] aos seus clientes. Este estudo retrata uma situação de jogo, que irá tolerar falhas distribuídas e comportar-se de modo adequado. Para implementar os mecanismos necessárias, usaremos o ambiente de desenvolvimento de protocolos **APPIA**.

2 Pong4

2.1 O jogo reinventado

O objecto de estudo deste artigo, tem como base um jogo de computador que apesar de ter sido desenvolvido há algum tempo, continua actual nos paradigmas inerentes à sua concretização num ambiente multi-jogador, especialmente sobre uma rede de comunicação. Na sua versão original, era jogado por um ou dois jogadores, que tinham que impedir que a bola de jogo tocasse no seu lado do ecrã (esquerdo ou direito) dispondo para isso de uma pequena barra (*paddle*) que reflectia a bola no sentido oposto. Neste nosso estudo, temos quatro jogadores em simultâneo, cada um defendendo um lado do ecrã. Topo, esquerda, direita e base. Os jogadores

[1] QOS - Quality of Service

(clientes) ligam-se através da rede ao jogo (servidor) que reflecte todas as acções tomadas por cada um dos jogadores.

2.2 O problema

Esta arquitectura básica de comunicação em rede (cliente-servidor) sofre de um problema estrutural. O jogo é interrompido sempre que o servidor termina, voluntária ou involuntariamente. Para resolver este problema, vamos tentar criar um sistema que tolere este tipo de faltas de omissão e recupere a execução do jogo da forma mais imperceptível possível para os jogadores.

3 Appia

Para desenvolver aplicações fiáveis num ambiente distribuído, torna-se imperativo a utilização de ferramentas já existentes que facilitam a implementação de mecanismos conhecidos para a resolução de problemas recorrentes. Uma das componentes mais exigentes de qualquer aplicação distribuída é a comunicação. Nesta área surgiu uma ferramenta que facilita o desenho e implementação de camadas lógicas (protocolos) de comunicação de forma fléxivel e fiável. Esta ferramenta é conhecida como Appia, e tal como a primeira estrada romana (Via Appia[2]) promete facilitar - e muito - a comunicação duma forma pioneira. Vamos usar esta ferramenta para desenvolver um modelo de comunicação robusto e fiável, através de protocolos de comunicação de grupo, replicação de dados e gestão de eventos.

3.1 AppiaXML

Para especificar as várias camadas protocolares e respectivos parâmetros, usaremos o AppiaXML. Esta extensão do Appia permite a ordenação e configuração das camadas através de um ficheiro XML, o que simplifica significativamente o processo inicial de construção do sistema.

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

4 Arquitectura do Sistema

Tendo em conta o problema anteriormente descrito, chegámos a uma arquitectura em que temos de um a quatro jogadores (clientes) e um ou mais servidores. Os servidores do jogo estão integrados num grupo para poderem trocar mensagens entre si mantendo a ordem total permitindo assim mecanismos de replicação em cima de protocolos de políticas de grupo fornecidos pelo Appia. Há ainda um detector de falhas, implementado através de um servidor Gossip externo, que também está implementado no Appia.

5 Conceitos Importantes

5.1 Vistas no Appia

Um conceito importante que está inerente ao Appia e comunicação em grupo é o conceito de vistas. Neste ambiente, o de comunicação em conjunto, uma vista representa os membros que fazem parte desse mesmo conjunto. É isso que permite detectar mudanças de vistas, juntá-las e criar uma nova vista, com novos membros e/ou sem alguns dos membros anteriores. Quando um pseudo-membro inicia, não pode juntar-se imediatamente à vista actual. Cria uma nova vista com apenas um membro, ele próprio, e o sistema detecta a presença de uma vista concorrente no sistema, isto é, uma vista diferente da vista actual. O coordenador da vista junta as vistas e é assim que se procede à inscrição de um novo membro.

5.2 Replicação

Um dos pontos principais de tolerância a faltas desta arquitectura é a replicação de dados nos vários servidores do grupo. Os dados que têm de ser replicados são:

- Estado global do jogo (posição da bola e do paddle de cada jogador)
- Fila de Coordenadores/Servidores

5.3 Propagação

O coordenador activo é o único elemento que efectua propagação no sistema. Fá-lo sempre que houver alteração na lista de jogadores ligados, propagando os eventos para todos os membros do grupo.

5.4 Comunicação em grupo

Para haver comunicação em grupo, o Appia fornece-nos camadas que garantem sincronismo virtual.[3] Com este paradigma, garante-se que todos os intervenientes sabem quem pertence à vista actual. Para além disso, todas as mudanças de vista ocorrem na mesma ordem para todos os membros, mantendo assim a ordem total destes eventos. Uma condição desta propriedade é que todas as mensagens devem ser entregues na mesma vista em que foram enviadas. Antes de ser instalada uma nova vista, o grupo é bloqueado, as mensagens em trânsito entregues e só depois se faz a troca.

6 Intervenientes e interacções

6.1 Coordenador e Servidor Activo

Quando o primeiro servidor se liga e é integrado no grupo, este auto-elege-se como o **coordenador** e **servidor activo**. O coordenador tem ainda a função de criar uma **fila de coordenadores** e de a propagar por todos os servidores do grupo. Trata-se de uma fila normal, onde o elemento à cabeça da mesma é o coordenador actual. Sempre que novos membros se juntarem ao grupo, estes devem ser adicionados à cauda. Esta fila será actualizada em cada mudança de vista e enviada a todos os membros do grupo. Para além disso, sempre que a instalação de uma nova vista implica a substituição do coordenador, o membro à cabeça da fila é retirado, e o novo coordenador assume funções, isto é, o elemento que se encontrava em segundo lugar na fila. Esse coordenador deve notificar todos os clientes do seu ip para que estes se liguem. Para tal, o coordenador deve ir notificando todos os membros do grupo sempre que um jogador entre ou saia do jogo. Como há uma camada do Appia inferior que garante a comunicação fiável entre os membros, sabemos que todos terão sempre uma lista de jogadores actualizada.

6.2 Jogadores (clientes)

Um cliente, para se ligar a um servidor precisa de apenas ter conhecimento do IP multicast associado ao grupo. Antes de iniciar a ligação ao servidor, este envia um pedido UDP para o endereço multicast do grupo a perguntar qual o ip do **servidor activo**. O coordenador actual deve responder com o seu ip para o cliente efectuar a ligação e posterior arranque do jogo. Durante o jogo, o cliente deve estar preparado para receber um ip de um novo servidor ao qual se deve ligar imediatamente. Isto acontece quando o servidor activo actual falhou.

6.3 Servidor Gossip

Este servidor é um elemento fora do grupo de servidores que serve para detectar vistas concorrentes, isto é, vistas diferentes dentro do mesmo uni-

[3] A. Pinto - *Appia Group Communication*, Outubro 2005

verso. Isto significa que houve uma alteração à lista de membros da vista. Há membros novos ou houve membros que já não fazem parte do grupo. A comunicação entre o servidor Gossip e as camadas principais de comunicação é feita por um segundo canal do Appia.

7 Opções de Implementação

Para tomarmos estas decisões houve uma discussão de opções alternativas de implementação, no entanto, estas foram as que nos pareceram mais viáveis.

Considerámos colocar os clientes a ligarem-se a diferentes servidores do mesmo grupo e manter o estado através de propagação de eventos aos outros membros do grupo. Esta opção foi desde logo excluída, já que o volume de comunicações na rede seria muito superior, já que no máximo quatro servidores - visto que há até quatro jogadores - iriam estar a enviar o estado do jogo para todos os membros da rede. Da forma que escolhemos, apenas há um servidor a enviar o estado, o que diminiu o envio de pacotes para a rede.

Para além disto, considerámos utilizar um serviço de nomes para fazer a tradução de um domínio no ip do coordenador/servidor actual. No entanto, isto iria forçar uma comunicação por parte do cliente a um servidor, um custo adicional de recursos para ter um servidor exterior e manter uma tabela com associações nomes/ips, mas a principal razão é que estavamos a criar mais um ponto falível no sistema, que precisaria de implementar tolerança a faltas. Da forma que adoptámos, temos a certeza que se houver um servidor ligado, este irá responder ao cliente com o seu ip.

8 Conclusão

O Appia é uma ferramenta poderosa, bem implementada, com grandes possibilidade de suporte e de destacar, é a recente adição da extensão do AppiaXML[4]. Desta forma pode-se concentrar mais no desenho das camadas protocolares sem ter que perder muito tempo com o código [Java] necessário para criar lógicamente a estrutura do sistema. Com os protocolos fornecidos de comunicação em grupo, construir uma aplicação deste género em cima do Appia é muito mais compensador, já que não se perde tempo a implementar soluções para problemas recorrentes e pode-se dedicar esse tempo ganho a outros aspectos do sistema, que acaba por ganhar noutros aspectos.

9 Leitura Recomendada

- **[1]** Paulo Verissimo e Luís Rodrigues, ”Distributed Systems for System Architects”, Kluwer Academic Publishers ISBN 0-7923-7266-2
- **[2]** L. R. Hugo Miranda, Alexandre Pinto, Application program interface specification of *Appia* version 1.2. Technical report, Universidade de Lisboa, Julho 2001.
- **[3]** A. Pinto, *Appia Group Communication*, Outubro 2005

[4] J. Mocito, L. Rosa, N. Almeida, L. Rodrigues - *AppiaXML - A Brief Tutorial*, 24 de Setembro, 2004