Mecanismos de tolerância a faltas em jogos de computador multi-utilizador

Pedro Magrito
i25988@alunos.di.fc.ul.pt

Rui Pacheco
i25278@alunos.di.fc.ul.pt

Rui Miranda
i26610@alunos.di.fc.ul.pt

Abstract

Com a crescente expansão das novas tecnologias e comunicação via internet, o mercado dos jogos multi-utilizadores atingiu um nível astronómico. A fiabilidade e confiabilidade são cada vez mais pontos decisivos na escolha de um produto. Sentiu-se necessidade de recorrer a mecanismos que minimizassem os vários tipos de faltas que podem ocorrer e corrigi-las de forma a que o utilizador não dê conta da existência delas. O artigo que se segue descreve detalhes de concepção de um sistema multi-utilizador que sirva de suporte ao famoso jogo do "Pong".

1. Introdução

O objectivo deste texto é introduzir mecanismos de tolerância a faltas numa aplicação multi-utilizador. Uma das formas de concretizar este tipo de aplicações é utilizando chamadas a procedimentos remotos (RPC). Este modelo, tambem conhecido por cliente/servidor, funciona enviando um pedido pela rede a um servidor, que o executa e envia a resposta ao cliente. Este modelo tem problemas relacionados com a falha do servidor, se o servidor sofre algum problema, todo o sistema pode falhar.

"A distributed system is the one that prevents you from working because of the failure of a machine that you never heard of."

Tendo em conta a famosa citação de Leslie Lamport e com base na experiência adquirida, sabemos que todos os programas podem falhar devido a diversas circunstâncias que podem ou não ser previstas. Somente com um servidor centralizado o problema não tem resolução, uma vez que todo o sistema se encontra dependente da informação retida num ponto. Com o recurso à replicação da informação e conferindo a possibilidade de qualquer elemento do grupo se comportar como servidor, podemos obter um sistema tolerante a faltas pelo que este não será afectado pela perda de uma máquina ou de um serviço. No resto deste artigo descreve-se o modelo de replicação e respectivas primitivas de comunicação(secção 2); a arquitetura do sistema com replicação e entidades intervenientes, (secção 3); as camadas desenvolvidas (secção 4); e a conclusão (secção 5).

2. Modelo

Nesta secção descrevemos o modelo que pensamos concretizar para obter tolerância a faltas no nosso projecto. O nosso sistema é composto com uma componente, dividida em 2 sub-componentes: a do servidor que é implementada por um cliente coordenador; e a do cliente, que trata os eventos de um cliente normal. O cliente coordenador é ao mesmo tempo considerado um cliente normal. Em seguida descrevemos os componentes em que se baseia o nosso sistema.

2.1. Primitivas de comunicação

Comunicação ponto-a-ponto Permite a comunicação entre clientes e servidores usando tcp.

Comunicação em Grupo Permite a comunicação entre os elementos do grupo usando Broadcast ou Multicast.

Sincronia Virtual Uma vista representa o conjunto de processos que fazem parte de um grupo. Quando existe uma mudança na filiação do grupo é entregue uma nova vista a todos os elementos. A sincronia virtual garante que numa certa vista todos os processos "correctos"recebem o mesmo conjunto de mensagens.

Ordem Total Propriedade que garante a entrega das mensagens a todos os membros "correctos"do grupo pela mesma ordem.

2.2. Replicação passiva

Neste modelo apenas existe uma copia do estado. Um dos elementos do grupo é eleito como coordenador que fica

encarregue de efectuar as operações e actualizar o novo estado em todas as réplicas. As réplicas têm a responsabilidade de actualizar o seu estado quando recebem mensagens do coordenador. Este tipo de replicação necessita de mais largura de banda da parte do coordenador do que com a replicação activa, mas, no entanto, minimiza o numero total de mensagens a circular na rede. Tendo em conta que este tipo de aplicação não exige muitos recursos optámos pelo uso da replicação passiva.

3. Arquitectura

O projecto proposto vai ser desenvolvido sobre o Appia, um sistema de comunicação baseado em micro-protocolos desenvolvido na Faculdade de Ciências. O sistema está definido de modo a que qualquer cliente possa ser coordenador. Mesmo em alturas de falha crítica do coordenador o sistema continua a funcionar normalmente, sem qualquer diferença para os outros utilizadores.

3.1. Cliente/Coordenador

No nosso sistema todos os clientes têm oportunidade de serem coordenadores. O primeiro cliente a filiar-se no grupo é eleito coordenador. O coordenador tem como tarefa receber os pedidos dos clientes e enviar uma actualização do estado a todos. Recebe os pedidos através de uma ligação ponto-a-ponto estabelecida pelo cliente e responde com a actualização usando comunicação em grupo.O coordenador também tem como tarefa a gestão das filiações. Todos os clientes mantém uma réplica actualizada do estado do servidor de modo a que o sistema possa recuperar das falhas mais critícas do coordenador. Em caso de falha do coordenador, um elemento do grupo assumirá essa responsabilidade, usando como critério a antiguidade no grupo.

4. Camadas de protocolos

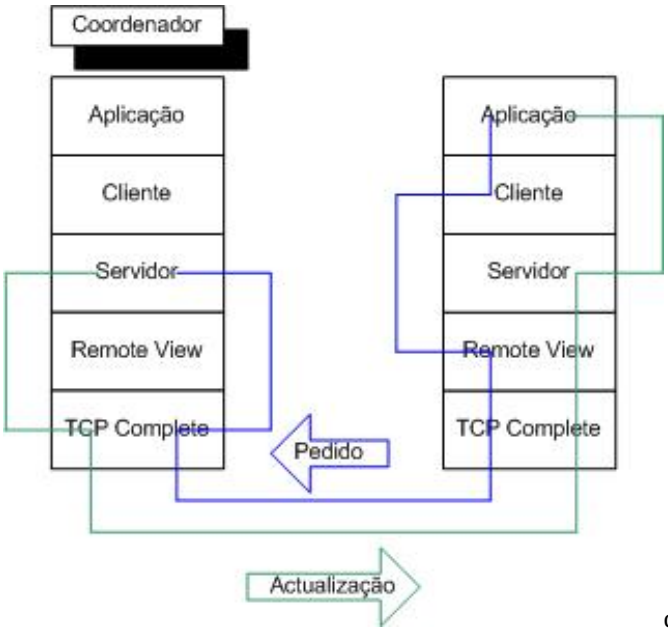
Nesta secção descrevemos todas as camadas alteradas ou desenvolvidas durante a replicação do jogo.

4.1. Camada Aplicação

Esta camada corresponde à interface gráfica e tem como responsabilidade informar a camada cliente das alterações efectuadas pelo utilizador. Recebe as actualizações da camada cliente e actualiza as animações.

4.2. Camada Cliente

A camada Cliente recebe os pedidos da camada aplicação e trata os mesmos para serem enviados ao coordenador. A cada pedido é atribuido um identificador único,



dth=0.5]s

que contém também a identificação do cliente responsável pelo pedido. O cliente é também responsável pelos pedidos de entrada e saída do grupo.

4.3. Camada Servidor

Esta camada é utilizada somente pelo processo coordenador e tem como função receber os pedidos dos clientes e propagar para todo o grupo as actualizações. Nos restantes clientes esta camada é utilizada para replicação do servidor.

5. Conclusões

Neste artigo descrevemos a arquitectura de um sistema multi-utilizador, com replicação passiva. Após algum tempo de consideração considerámos esta a melhor forma para a resolução do problema proposto. Este é um sistema fiável que suporta faltas até 3 máquinas (75Este sistema deverá ser implementado em linguagem JAVA e com recurso ao sistema de comunicação em grupo Appia.

Referências

[1] P. Vicente, J. Martins: *Arquitectura de um Sistema de Chamadas a Procedimentos Remotos a Servidores Replicados*

[2] P. Veríssimo and L. Rodrigues: *Distributed Systems for Systems Architecs.* Kluwer Academics Publishers, 2001.

[3] H. Miranda, A.Pinto and L. Rodrigues: *Appia, a flexible protocol supporting multiple coordinated channels.*