Sistema de Edição/Subscrição Tolerante a Faltas

Mauro Lemos
i30340@alunos.di.fc.ul.pt

Nuno Salvador
i31569@alunos.di.fc.ul.pt

Tito Nobre
i31667@alunos.di.fc.ul.pt

TFD007
24 de Outubro de 2006

Abstract

Actualmente, existe uma procura crescente de Sistemas Distríbuidos de tempo-real fiáveis, havendo uma grande preocupação em dotá-los de mecanismos de tolerância a faltas para garantir a sua disponibilidade. A razão para este facto são as grandes vantagens de modularidade e extensibilidade que sistemas deste tipo apresentam.
Usando as potencialidades da framework do Appia, neste trabalho, apresentamos um modelo de edição/subscrição distribuído, tolerante a faltas, que oferece um serviço contínuo e transparente ao utilizador.

1. Introdução

Um sistema de edição/subscrição baseia-se na troca assíncrona de mensagens, conhecidas também como eventos. Existem um ou mais editores que publicam mensagens para um intermediário - *broker*[1] . Os subscritores que pretendam subscrever estas mensagens registam-se no *broker*, que gere as mensagens num sistema baseado em tópicos e as distribui pelos subscritores que tenham subscrito mensagens desse tópico.

Esta forma assíncrona[2] de comunicação por mensagens é bastante mais escalável do que alternativas basedas em filas de mensagens, dado que os emissores das mensagens apenas necessitam de se preocupar com a criação da mensagem, deixando a tarefa de entrega das mesmas para o serviço responsável. Na maior parte das vezes os emissores não têm conhecimento de quem subscreveu as suas mensagens.

[1]Um broker consiste num intermediário que faz a tradução da mensagem no protocolo formal do emissor para o protocolo formal do receptor num sistema de comunicação baseado na troca de mensagens formalmente definidas.

[2]Não se baseia num sistema bloqueante pedido/resposta, pois o emissor e o receptor são independentes e não estão sincronizados.

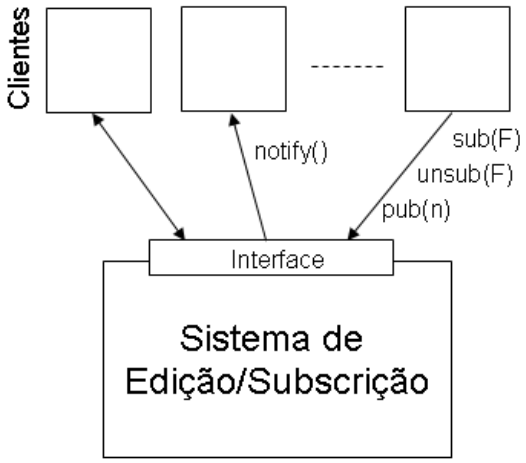

Figura 1. Vista Caixa Preta de um sistema edicao/subscricao [6]

A implementação de um serviço de edição/subscrição que use notificação de eventos pode seguir uma abordagem centralizada onde existe um único elemento responsável por manter o registo das subscrições dos utilizadores e pela propagação dos eventos gerados. Este tipo de implementação é bastante simples mas apresenta problemas de escalabilidade e tolerância a faltas, que se torna impraticável quando o número de clientes é grande.

Uma abordagem diferente baseia-se num modelo distribuído, que consiste num conjunto de servidores interligados, em que cada um é responsável por gerir serviços para uma parte dos utilizadores. A eficiência deste modelo depende da forma como a interligação dos servidores é feita, de como os eventos são propagados e de como é feito o processamento dos pedidos dos utilizadores.

Existe já diverso trabalho feito no que diz respeito a serviços de notificação, muitos deles já desenhados e im-

plementados, como é o caso do SIENA [3], JEDI [4] entre outros.

O restante artigo encontra-se organizado da seguinte forma:

Na secção 2 é descrito o problema apresentado, na secção 3 é feito um resumo das ferramentas e conceitos relevantes, na secção 4 é descrita a arquitectura do modelo proposto, como é feita a comunicação entre os elementos do sistema, a pilha protocolar e o modelo de faltas. Por fim, na secção 5 são apresentadas as conclusões sobre o artigo.

2. Problema Proposto [2]

Pretende-se implementar um sistema de edição/subscrição simplificado, que fornece serviços a dois tipos de clientes - *editores e subscritores*. Os editores devem poder publicar informação na forma de eventos de forma a que estes sejam recebidos em tempo real por todos os subscritores interessados. Como forma de simplificação, cada evento publicado é marcado com uma etiqueta que identifica o assunto. Assim o subscritor quando deseja receber eventos de um dado tipo apenas necessita de referir o assunto. Cada evento terá associada uma qualidade serviço, definida pelo editor do assunto correspondente.

A comunicação entre os editores e os subscritores é feita directamente, mas existe um servidor que mantém a associação entre assuntos e respectivas qualidades de serviço e nomes de grupos. Este servidor será replicado para oferecer tolerância a faltas.

3. Ferramentas e Conceitos

3.1. Appia

O Appia [1] é uma *framework* desenvolvida em JAVA, que oferece suporte ao desenvolvimento e execução de protocolos por camadas.

Esta framework é constituída por um núcleo que permite a composição de pilhas de protocolos e por um conjunto de protocolos que oferecem diversas propriedades como comunicação em grupo, garantias de ordenação, difusão atómica entre outras.

A grande vantagem do Appia em relação a outros sistemas similares é permitir a criação de pilhas de protocolos mais flexíveis com estruturas complexas.

3.2. Comunicação em Grupo [5]

Um sistema que oferece comunicação em grupo, deverá dispôr de suporte de filiação, para permitir adicionar/retirar elementos do grupo e saber a qualquer momento definir a constituição (vista) do grupo; deverá dispôr também de suporte à comunicação entre os elementos do grupo.

A comunicação entre servidores é fundamental para garantir um correcto funcionamento de um sistema distribuído. Assim, torna-se vantajoso utilizar o serviço anteriormente descrito, já fornecido com a *framework* do Appia.

3.3. Elemento Coordenador

Existem dois tipos de abordagens possíveis no que toca à comunicação entre os utilizadores e o sistema de subscrição, a saber:

- com elemento coordenador, em que é definido um servidor do sistema como o elemento coordenador, servidor este ao qual se ligam todos os utilizadores quando desejam fazer subscrições ou publicações de mensagens. Este servidor é responsável por propagar a informação para o resto dos servidores de sistema de modo a garantir coerência e tolerância a faltas caso o servidor coordenador falhe.

- fazendo balanceamento de carga, isto é, cada servidor ficaria responsável por um grupo de utilizadores de sistema. Caso ocorra uma falha num servidor, os utilizadores ligados ao mesmo terão que se ligar a outro servidor activo, o que obriga a haver coerência entre a informação de todos os servidores do sistema, para não ser perdida informação.

As opções tomadas em relação a este aspecto serão explicadas na secção 4 deste artigo.

3.4. Replicação

A replicação [7] é uma técnica bastante utilizada para obter tolerância a faltas, sendo uma das técnicas mais importantes para a sua concretização a difusão fiável com ordem total, também designada por ordem atómica.

Garante-se, assim, que todas as réplicas correctas estejam coerentes e que disponibilizem os mesmos arquivos.

Existem dois tipos de replicação:

- Passiva: na replicação passiva existe sempre um servidor principal com que os clientes interactuam. O segundo servidor fica de reserva, de forma a que, quando detecta que o servidor primário falhou, torna-se o primário.

- Activa: na replicação activa não existe controlo centralizado, todos os servidores recebem pela mesma ordem os pedidos dos clientes, efectuam a operação, determinam qual o resultado correcto por votação, chegando a um consenso, e devolvem a resposta ao cliente.

O método a implementar neste trabalho é discutido posteriormente na secção 4.

3.5. Uniformidade

A uniformidade num sistema deste tipo é essencial, pois, o facto de apenas alguns dos elementos do grupo receberem as mensagens enviadas tornaria o sistema incoerente caso fosse necessário trocar de servidor coordenador. Assim, tomando partido da camada *Uniform* da distribuição do Appia é garantido que todos os servidores correctos recebem a mensagem enviada ou nenhum a recebe. Este aspecto será novamente referido na secção 4.

4. Arquitectura

O modelo de sistema para resolver o problema proposto será um modelo distribuído, baseado na existência de um coordenador de modo a simplificar a implementação. Este coordenador será um dos elementos do grupo de servidores que fazem parte do sistema de subscrição. Será utilizada replicação passiva, em que o coordenador é responsável por actualizar as suas réplicas num intervalo de tempo pré-definido, ou em casos que seja realmente necessário propagar nova informação, como por exemplo aquando de uma nova publicação por parte de um editor.

Em suma, existe um servidor coordenador ao qual os clientes se ligam quando desejam receber ou publicar novas mensagens. Este servidor é responsável por disponibilizar a interface para os clientes e por manter as réplicas actualizadas para o caso de ser necessário mudar de coordenador (falha do coordenador actual).

4.1. Pilha de Protocolos

No Cliente a pilha de protocolos [Figura 2] apresenta apenas duas camadas, a camada "Aplicação" e a camada "TcpCompleteLayer". A camada "Aplicação" representa a aplicação própriamente dita onde é disponiblizada a interface para os utilizadores do sistema; a camada "TcpCompleteLayer" encarrega-se da comunicação com o servidor coordenador do serviço de edição/subscrição.

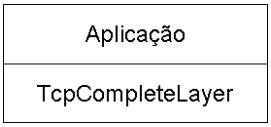

Figura 2. Pilha de Protocolos do Cliente

De modo a oferecer duas qualidades de serviço distintas, o servidor necessita de ter dois canais distintos. Na [Figura 3] pode-se observar o canal 1, que oferece sincronia na vista [5] e o canal 2 que oferece ordenação total através da camada *TotalAbcastLayer*. Além disto oferece

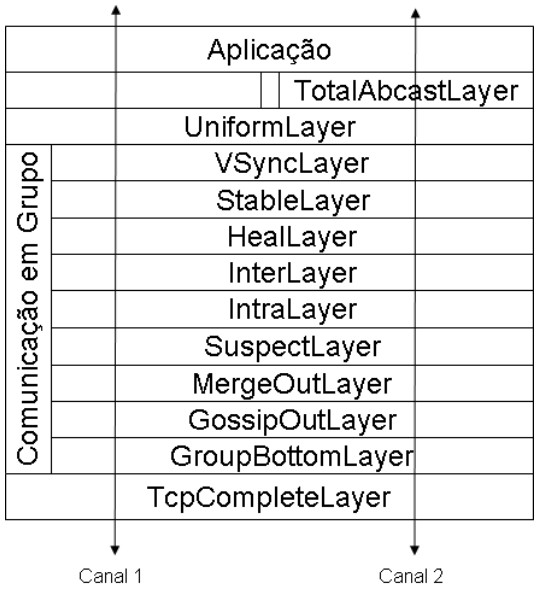

Figura 3. Pilha de Protocolos do Servidor

também uniformidade através da camada *Uniform* para evitar incoerências entre réplicas de servidores (membros do grupo). As restantes camadas da comunicação em grupo podem ser vistas em mais pormenor em [5]. Todas estas camadas referidas anteriormente assentam sobre a camada *TcpCompleteLayer* que é responsável pela comunicação usando o protocolo TCP.

4.2. Interacções entre Elementos

- Cliente - Servidor: para que o utilizador consiga receber os eventos tem que se ligar a um servidor, o coordenador. Os utilizadores comunicam ponto-a-ponto com o coordenador quando querem receber ou publicar conteúdo. O servidor encarrega-se depois de devolver os conteúdos aos utilizadores que os subscreveram.
- Servidor - Servidor: a interacção entre os vários servidores é fundamental para garantir um sistema tolerante a faltas, garantindo, assim, que todos os servidores apresentam os mesmos eventos quando contactados por um utilizador, havendo uma coerência e eficiência. A replicação do servidor deve ser transparente para o utilizador.

4.3. Detecção e Recuperação de Falhas

De modo a detectar o caso em que o servidor coordenador falhe a aplicação dos utilizadores necessitam de ter acesso à lista de servidores activos para se poderem ligar a outro. Através da sincronia na vista, oferecida pela framework do Appia, é possível detectar quando um membro do

grupo falha. Esse elemento é retirado da vista e a nova vista passa a ser a actual (com o conhecimento de todos os membros activos). No caso desse elemento que falhou ser o coordenador é eleito um novo coordenador (o primeiro elemento da nova vista). Torna-se assim necessário informar os utilizadores do novo coordenador. Cada utilizador mantém uma lista dos servidores do sistema. Quando um novo coordenador é eleito este envia a vista actual do grupo para os utilizadores activos de modo a poderem actualizar a sua lista e ligarem-se ao novo coordenador.

4.4. Actualização das Réplicas

A actualização das réplicas, por parte do servidor coordenador, ocorre em intervalos de 200ms ou em situações específicas, no caso de ser publicada nova informação por parte de um editor. Assim, garante-se que o estado entre as réplicas é coerente e não há perda de informação no sistema, no caso de ser necessário mudar de servidor coordenador. Esta garantia é dada pela camada *Uniform* da pilha protocolar do servidor.

5. Conclusão

Neste artigo foi apresentada uma possível solução para um sistema de edição/subscrição distribuído tolerante a faltas em que é utilizada a comunicação em grupo para replicar o servidor de maneira a que os utilizadores possam continuar, de forma transparente, a editar e subscrever mensagens mesmo que o servidor a que estão ligados falhe.

Referências

[1] Appia toolkit homepage. http://appia.di.fc.ul.pt.

[2] Enunciado do projecto de tfd, 2006/2007. http://www.di.fc.ul.pt/ ler/docencia/tfd/trab.html.

[3] A. Carzaniga, D. S. Rorenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service, 2001.

[4] G. Cugola, E. D. Nitto, and A. Fuggetta. The jedi event-based infrastructure and its application to the development of the opss wfms, 2001.

[5] A. Pinto. Appia group communication, 2005.

[6] G. uhl. *Large-Scale Content-Based Publish /Subscribe Systems*. PhD thesis, Darmstadt University of Technology, 2002.

[7] P. Vicente, H. Miranda, and L. Rodrigues. Protocolo híbrido de ordem total uniforme com entrega optimista, 2001. TR-01-14.