STEdSub - Sistema Tolerante de Edição e Subscrição em grupos

Fernando André
i29040@alunos.di.fc.ul.pt

Ricardo Paiva
i29649@alunos.di.fc.ul.pt

Hugo Nunes
i29731@alunos.di.fc.ul.pt

30 de Outubro de 2006
Grupo TFD05

Resumo

As aplicações informáticas, com o decorrer dos anos, evoluiram de arquitecturas centralizadas para modelos de computação distribuída. Dada a actual conjectura dos sistemas de informação, cada vez mais distribuídos e exigentes, torna-se necessário adoptar alternativas, cada vez mais complexas, em que o conceito servidor e a dependência no seu funcionamento deixe de representar um papel central. Neste panorama é importante a utilização de técnicas de tolerância a faltas para garantir a disponibilidade e fiabilidade de um serviço que se quer prestar.
No âmbito da cadeira de Tolerância a Faltas Distribuída (TFD) foi-nos proposta a realização de um sistema que forneça serviços de filiação e comunicação em grupo com diferentes propriedades e tipos de ordenação.

1. Introdução

Os sistemas tolerantes a faltas distribuídas são bastante úteis quando é necessário garantir que quando um determinado componente falha, todos os outros se mantenham a funcionar e a prestar o serviço para o qual foram concebidos. Para além deste facto, é indispensável manter a coerência do sistema e evitar informações contraditórias.

Neste trabalho propomo-nos a conceber um sistema de edição-subscrição simplificado. Este sistema deve fornecer serviços para dois tipos de clientes, a saber: os **editores** e os **subscritores**. O objectivo é permitir que os editores publiquem informação sob a forma de eventos para que estes sejam recebidos em tempo-real por todos os subscritores interessados. Os artigos submetidos neste sistema tem associado um assunto que os identifica, sendo que os subscritores quando pretendem receber um conteúdo têm de indicar este mesmo assunto.

A cada evento está também associada uma qualidade de serviço (QoS) distinta, que é indicada pelos editores quando publicam um artigo. Assume-se que a cada assunto está sempre associada a mesma QoS.

Apesar da comunicação editor-subscritor ser feita directamente, o sistema a desenvolver baseia-se na existência de um conjunto de servidores que mantém a associação entre assuntos e respectivas QoS e nomes de grupos. Este conjunto de servidores vai ter um coordenador que irá ter responsabilidades acrescidas em determinadas tarefas.

Este sistema não irá ser persistente, isto é, quando se publica informação esta é apenas recebida pelos subscritores activos nesse momento, sendo que, por oposição, os subscritores que estiverem desligados não a recebem. Por esta razão não haverá necessidade de armazenar informação de modo permanente, para mais tarde ser entegue.

Sobre este projecto interessa ainda acrescentar que iremos utilizar a *framework* Appia para implementar a comunicação entre as diferententes componentes do sistema, não só pelas suas características, mas também pelo tipo de serviços que oferece.

Este artigo encontra-se organizado da seguinte forma. A secção 2 apresenta o problema e a ferramenta para o resolver, na secção 3 a relação entre clientes (editores e subscritores) e servidores responsáveis por manter o sistema operacional. Na secção 4 são descritos conceitos importantes para a resolução do problema, a secção 5 contém o modelo de faltas relativo a esta aplicação, na 6 apresenta-se a arquitectura projectada para resolver o problema e finalmente na secção 7 apresentam-se as conclusões deste trabalho.

2. Descrição do problema

Esta aplicação irá implementar um sistema de edição-subscrição, onde editores irão fazer a inserção e publicação de temas/assuntos e subscritores irao subscrever e/ou receber esses mesmos temas.

É importante referir ainda que o projecto seguirá o modelo cliente-servidor, tendo em conta a preocupação de tornar a aplicação tolerante a faltas, apesar de ser udada comunicação em grupo noutras áreas.

2.1. Principio de edição-subscrição

Apesar da existência de inúmeros sistemas baseados no principio de edição-subscrição, neste trabalho projectámos uma arquitectura para este tipo de sistema distribuído e tolerante a faltas usando como plataforma a *framework* Appia [5] [6] que possibilita o uso de um conjunto de protocolos que fornecem comunicação em grupo com diversas propriedades.

Pretendemos com este projecto demonstrar, por um lado as vantagens de uma aproximação distribuída ao problema e, por outro, a forma simples como se consegue atingir resultados bastante satisfatórios utilizando ferramentas já disponíveis.

2.2. Appia

O Appia [5] é uma *framework* de suporte ao desenvolvimento e execução de protocolos por camadas, totalmente implementada em Java, que oferece abstracções que implementam inúmeros modelos de computação distribuída. No Appia a arquitectura das aplicações assenta numa pilha de microprotocolos, cuja implementação toma a forma de um canal [3] Appia. A informação circula neste canal sob a forma de eventos. No trabalho a implementar, pretende-se tomar partido das vantagens do Appia para desenvolver um sistema de comunicação em grupo para os servidores.

3. Operações entre componentes

3.1. Editor - Servidor

De modo a participar no STEdSub um editor (sendo este um cliente) tem de se conectar a um dos servidores do sistema. A interacção entre editores e servidores deve, principalmente incidir sobre a capacidade de inserir novos temas de conteúdos no sistema de forma coerente e eficiente. O editor deverá ainda indicar, para além do tema, a qualidade de serviço que quer oferecer. Para esta propriedade irá existir duas opções, sincronia na vista e ordem total [2].

Os editores comunicam ponto-a-ponto com um servidor. A informação é transmitida apenas para o servidor ao qual se encontra conectado, que terá por sua vez de criar um grupo que está afecto ao tema proposto para inserção. Caso o tema proposto já exista, o servidor junta este editor ao grupo correspondente.

3.2. Subscritor - Servidor

A interacção Subscritor-Servidor segue os mesmos moldes da interacção Editor-Servidor, sendo que a única diferença reside no facto de o subscritor não necessita indicar uma qualidade de serviço, ele apenas requisita um tema ao servidor, e deve estar preparado para receber conteudos com diferentes QoS (sincronia na vista e ordem total). A comunicação entre subscritor e servidor é igualmente ponto-a-ponto. Ao efectuar a subscrição de um tema, o servidor inscreve o subscritor no grupo do tema pretendido, se para este tema já existir um grupo, ou cria um novo grupo caso o tema nunca tenha sido solicitado.

3.3. Servidor - Servidor

A interacção entre servidores é essencial para garantir que o sistema é tolerante a faltas. Aos servidores compete assegurar, principalmente, garantias de continuidade de serviço aos clientes. O problema reside em conseguir oferecer estas garantias em detrimento de coerência e eficiência do sistema. Isto significa que para os clientes (editores e subscritores) o funcionamento replicado do servidor e a sua recuperação de falhas devem ser transparentes. Para poder oferecer estas propriedades, os servidores devem comunicar usando um sistema de comunicações fiáveis e em grupo, que no caso do Appia oferece garantias de qualidade de serviço, ordem total, e sincronia na vista.

3.4. Editor - Subscritor

A interacção Editor-Subscritor é em tudo semelhante à interacção Servidor-Servidor, com a excepção que o modelo de comunicação entre editor e subscritores não oferece serviços como sincronia na vista e ordem total. Aos editores compete apenas disseminar um conteúdo relacionado com um tema, pelo respectivo grupo associado e a fiabilidade de entrega de mensagens é assegurada pelo Appia.

4. Conceitos relevantes

4.1. Noção de grupo

A utilização de um sistema de comunicação de grupo [1] é vantajosa pois permite ter tolerância a faltas. Este sistema é conseguido dando uso aos protocolos fornecidos pelo Appia. Neste prisma, a noção de grupo surge como sendo um sistema que oferece as seguintes propriedades:

- Suporte de filiação para permitir acrescentar e/ou remover elementos de um grupo.
- Tornar possível a qualquer momento saber quais os elementos que pertencem a um determinado grupo.
- Oferecer um suporte de comunicação que permita a todos os elementos de um dado grupo comunicar entre si através de multicast [1] ou ponto-a-ponto.

4.2. Sincronia virtual

Por sincronia virtual [2][3] entende-se ser um mecanismo baseado em suporte de filiação e comunicação por broadcast fiável que permite saber, num determinado momento, quais os elementos pertencentes a um grupo - vista do grupo [5]. Sempre que a vista do grupo se altera, a sincronia virtual garante que todas as mensagens a enviar para a vista antiga são entregues, antes de instalar a nova vista.

4.3. Uniformidade

No contexto da comunicação em grupo, uniformidade [3] caracteriza-se pela seguinte garantia:

- Dada uma mensagem *m* enviada para o grupo, ou todos os elementos a recebem ou nenhum a recebe [2].

Com o objectivo de alcançar esta propriedade na implementação do enunciado proposto, tomou-se partido da camada *uniform* [5] fornecida pela distribuição da *framework* Appia. De notar que, para uma arquitectura como a especificada neste artigo, seria ideal uma camada uniforme que esperasse resposta de todos os elementos vivos do grupo um determinado momento. Mas, devido à imprecisão inerente à camada *Suspect* [5] do Appia (responsável pela detecção de falhas de membros do grupo), a camada *uniform*, a ser utilizada, necessita saber *à priori* o número mínimo de servidores necessários para garantir a uniformidade. Logo, se esse número mínimo não for igual ou superior a uma maioria dos servidores do grupo, a uniformidade pode não ser garantida. Se, pelo contrário, o número de servidores do grupo for inferior a esse valor estipulado *à priori*, todas as operações que necessitem de uniformidade deixam de estar operacionais.

4.4. Noção de coordenador

Frequentemente, de modo a simplificar as interacções entre elementos de um grupo, é eleito um coordenador [2] do grupo ao qual poderão ser atribuídas funcionalidades exclusivas como, por exemplo: prioridade de interacção com entidades exteriores ao grupo, poder de decisão em problemas de consenso, poder de decisão em problemas de replicação, etc.
A utilização de um coordenador na implementação do STEdSub está dependente do modelo arquitectural escolhido para a resolução do problema, nomeadamente o modelo de replicação.

4.5. Ordenação de mensagens

A ordenação de mensagens assume, principalmente, duas facetas:

- Determinar *à posteriori* a ordem em que as mensagens foram enviadas de modo a assumir ou excluir relações de causa-efeito entre elas.
- Garantir que as mensagens obedecem a critérios de ordenação estabelecidos por políticas definidas *à priori*.

Existem uma série de modelos de ordenação que permitem aplicar uma ou ambas das facetas referidas: ordem FIFO, ordem causal, ordem total, entre outros. Dada a natureza da arquitectura adoptada para resolver o enunciado proposto, apenas a noção de ordenação FIFO [2] interessará para este artigo.

4.6. Modelo de replicação

Para além de garantir a continuidade de serviço, o grupo de servidores tem de assegurar a coerência do mesmo através da replicação do estado do STEdSub pelos elementos (réplicas) do grupo de servidores. Consideramos de seguida três modelos de replicação possíveis para a resolução deste problema:

- Replicação Activa
- Replicação Semi-Activa
- Replicação Passiva

No modelo de replicação activa todos os servidores recebem actualizações de estado dos clientes e mantém um estado do sistema idêntico de servidor para servidor. A disseminação de estados para o cliente é feita por todos os servidores em simultâneo. A replicação activa tem algumas vantagens, como por exemplo, a possibilidade de efectuar

balanceamento de carga entre as réplicas, e a detecção de erros nas mensagens para o cliente através de um processo de maioria.
O modelo de replicação semi-activa assemelha-se ao modelo de replicação activa pois todos os servidores continuam a receber os estados do sistema, cabendo a um dos servidores - o coordenador - actualizar as réplicas do estado considerado correcto. Perante a falha do servidor coordenador um novo coordenador será eleito de entre os restantes elementos.
No modelo de replicação passiva cabe apenas a um servidor coordenador receber actualizações de estado vindas dos clientes; as restantes réplicas encontram-se em espera de actualizações periódicas - checkpoints - vindas do servidor coordenador. Perante a falha do servidor coordenador um novo coordenador é eleito de entre os restantes elementos do grupo.

4.7. Tipos de faltas [1][3]

Num sistema distribuído é imperativo ter especial cuidado com o tipo de faltas que o sistema tolera. Existem dois grupos essenciais de faltas: assertivas e omissivas. Na classe de faltas assertivas incluem-se as interacções entre componentes em que os valores transmitidos estão errados. Existem duas subclasses de faltas assertivas: semânticas e sintácticas: Uma falta semântica caracteriza-se pelo envio de um valor errado mas possível; uma falta sintáctica pelo envio de um valor impossível.
A detecção de faltas sintácticas é, pela própria natureza da linguagem Java, normalmente fácil, visto que o envio de um tipo de dados inesperados resultará no levantamento de uma excepção, tornando essa interacção inválida. O envio de um valor errado mas possível, torna a sua detecção virtualmente impraticável na arquitectura proposta (Uma maneira possível de detectar estas falhas semânticas seria através do uso de replicação activa e redundância de operação, onde um pedido a um componente originaria *n* respostas, através das quais se poderia deduzir um valor correcto com maior probabilidade).
As faltas omissivas situam-se no domínio do tempo, e dividem-se em três subcategorias: faltas por omissão, paragem e temporais.
Quando um componente falha em realizar uma determinada interacção por exemplo enviar um ACK considera-se uma falta por omissão, ou seja, um componente omite uma interacção potencialmente crítica para o funcionamento do sistema.
Uma falta por paragem acontece quando um componente falha e, por consequência, deixa de interagir com os restantes componentes.
Uma falta temporal é, essencialmente, um atraso numa comunicação, introduzindo latência na computação do sistema. As faltas de paragem são portanto falhas temporais com latência superior a um timeout do sistema.

5. Modelo de Faltas

É impraticável tentar desenvolver um sistema que se considere tolerante em todo o tipo de faltas mencionadas anteriormente. É com este pressuposto que surge a necessidade de definir um modelo de faltas onde se especifique que faltas se pretende tolerar.
A aproximação ao problema por nós escolhida privilegia a tolerância a faltas omissivas, considerando-se que faltas afirmativas não deverão ocorrer e que faltas maliciosas não são consideradas dado o contexto do projecto. Nomeadamente pretende-se tolerar faltas de atraso na entrega de mensagens, falha de servidores e garantir que estas falhas não prejudicam o normal funcionamento do STEdSub.

6. Arquitectura

Depois de uma pesquisa sobre arquitecturas deste género, Editor-Subscritor, tentaremos implementar uma arquitectura baseada num conjunto de servidores em que um deles actua como coordenador e a replicação do estado do coordenador será efectuada sobre TCP sem balanceamento de carga [4].
No modelo por nós projectado os clientes ligar-se-ão a um único servidor - o coordenador - encarregue de actualizar o estado do sistema. Os elementos do grupo de servidores são actualizados pelo coordenador periodicamente - *checkpointing* [1][4] - ou em situações que a sua actualização seja imediatamente necessária. Dada as características da replicação passiva, e de modo a reduzir a complexidade da solução não é feito nenhum balanceamento de carga dos servidores. Ou seja, o coordenador é responsável pela manutenção de um estado coerente em todas as réplicas e por disponibilizar aos clientes uma plataforma de edição-subscrição fiável.

6.1. Detalhes de Implementação

Nesta arquitectura, por nós projectada, valoriza-se a simplicidade, deixando de lado abordagens ao problema mais complexas.
A arquitectura deste sistema tem um funcionamento baseado nos seguintes critérios:

1. Todos os servidores do sistema fazem parte de um grupo implementado com a tecnologia da *framework* Appia. Com este sistema um dos servidores é automaticamente eleito coordenador do grupo.

2. Todos os editores e subscritores ligam-se ao sistema, mais concretamente ao coordenador.

3. Se o coordenador falhar os clientes (editores e subscritores) ligar-se-ão a um novo coordenador. Este novo coordenador é eleito de entre as réplicas activas no grupo de servidores. Se a eleição de um novo coordenador for possível, o sistema deixará de funcionar.

4. Durante o funcionamento do STEdSub, o estado deste, que se encontra no coordenador, é disseminado pelas réplicas, o que garante que caso ocorra uma falha no coordenador, nalgum momento os clientes (editores e subscritores) continuarão a funcionar e a ter acesso ao sistema enquanto existir um coordenador no grupo de servidores.

5. Em situações críticas será necessário garantir a uniformidade da informação entre as réplicas, antes de devolver uma resposta a um pedido.

No que toca à disseminação do estado dos sistema entre o coordenador e as réplica, esta será realizada periodicamente, ou quando se observarem as seguintes situações:

1. Quando um editor manifestar interesse em publicar um novo conteúdo.

2. Quando um subscritor manifestar interesse em inscrever-se num novo grupo.

3. Quando um subscritor manifestar interesse em sair de um grupo do qual é membro.

A disseminação é feita obrigatoriamente nestes casos para garantir que, após um editor ou um subscritor juntar-se ao sistema, se o servidor coordenador falhar o estado com a informação de um dos clientes terá sido enviado para as réplicas. Adicionalmente de modo a garantir a coerência, esta informação não pode ser enviada do coordenador aos clientes antes que o coordenador se certifique que as restantes réplicas recebam a informação: se tal não se verificasse, poderia ocorrer um fenómeno de *rollback* [4] onde o estado do sistema andaria para trás. Para garantir esta última condição poderá tomar-se partido da camada uniform fornecida pela *framework* Appia. Sempre que uma nova vista for criada o suporte de sincronia de sincronia virtual bloqueará todas as réplicas através do evento BlockOk. Neste tipo de acontecimento, é necessário garantir que todas as mensagens criticas serão guardadas para serem enviadas assim que o servidor seja desbloqueado.

6.2. Pilhas Protocolares

No Appia as propriedades suportadas pelo canal de comunicação são definidas com base nos protocolos utilizados e denomina-se Qualidade de Serviço (QoS). Neste projecto iremos ter dois tipos de QoS, sincronia na vista e ordem total, que são as opções de escolha dos editores para publicarem um conteúdo. No fundo iremos ter dois canais paralelos no appia, em que um atravessa uma camada que oferece ordem total e outro canal em que esta propriedade não é oferecida.

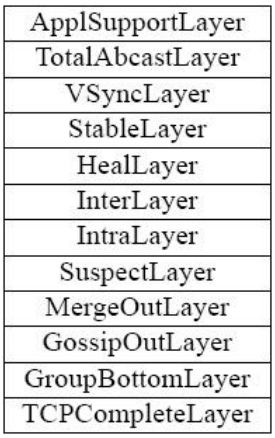
No nosso sistema definimos uma pilha protocolar que irá ter dois canais e que tem as propriedades habituais de um sistema com sincronismo virtual:

- Todos os membros de um grupo observam o seu grupo sobre a forma de vistas.
- Todos os membros de um grupo observam a mudança de vista pela mesma ordem. Existe uma ordem total das vistas.
- Todas as mensagens enviadas durante uma determinada vista são entregues nessa mesma vista.
- Se uma mudança de vista é necessária, o envio de mensagens é bloqueado até a formação de uma nova vista e até à replicação desta por todos os elementos. Todas as mensagens anteriores ao 'bloqueio' são entregues antes da mudança da vista.

Outra propriedade importante é a oferecida pela ordenação total de mensagens que, como já foi referido, garante que as mensagens são entregues pela mesma ordem em todos os processos intervenientes.

Com estas propriedades garantimos que os estados dos vários processos são sempre actuais e sincronizados, e garantimos também a tolerância a faltas provocadas por falhas de processos assim como também a sua reintegração no sistema.

Assim, a pilha protocolar definida, com as camadas partilhadas será:



A primeira camada da pilha, *TCPCompleteLayer*, é responsável pela comunicação real entre os processos usando

o protocolo TCP. As nove camadas acima desta executam diversas tarefas relacionadas com a comunicação em grupo e são elas que actuando em conjunto oferecem as garantias habituais de um sistema de comunicação com sincronismo virtual. A descrição detalhada do que cada uma destas camadas faz pode ser encontrada em [6]. A camada *TotalAbcastLayer* oferece a garantia de ordem total na entrega de mensagens à camada imediatamente acima, denominada *ApplSupportLayer* que corresponde à camada de suporte à interface do sistema e consiste na implementação da grande maioria das funcionalidades descritas neste artigo.

7. Trabalho Futuro

Iremos proceder à implementação do sistema descrito anteriormente recorrendo à plataforma Appia.

8. Conclusão

Ao optarmos pela criação de um sistema distribuído asseguramos à partida determinadas características ao funcionamento do sistema como por exemplo, maior disponibilidade, maior fiabilidade e eventualmente melhor desempenho em relação aos sistemas centralizados. Com o crescimento deste tipo de sistemas e dada a complexidade associada é necessário construi-los de forma a serem tolerantes a faltas, que neste trabalho específico passa pela utilização de réplicas que irão assegurar a coerência do sistema. Neste artigo encontra-se uma possível solução para um sistema de publish-subscribe tolerante a faltas que será implementado tirando partido de todas as facilidades oferecidas pela *framework* do Appia. A comunicação é feita directamente entre os clientes mas para que todo o processo de edição-subscrição corra dentro da normalidade foi necessário recorrer a um conjunto de servidores que de alguma forma coordena todo este processo.

References

[1] Paulo Veríssimo, Luís Rodrigues, 2001. *Distributed Systems for System Architects*, Kluwer Academic Publishers.

[2] Rachid Guerraoui, Luís Rodrigues, 2005. *Introduction to Reliable Distributed Programming*, Preliminary Draft, Springer-Verlag.

[3] Artigo apresentado pelos alunos José Côrte-Real, Leonel Duarte eMiguel Figueiredo no ano lectivo de 2005/2006 no âmbito da cadeira de TFD com o titulo: "'*FTP - Fault-Tolerant Pong*"'.

[4] *http://en.wikipedia.com* - Artigos sobre os conceitos.

[5] *http://appia.di.fc.ul.pt*

[6] *http://appia.di.fc.ul.pt/wiki/index.php?title=Documentation*