FTPS

A Fault Tolerant Publish Subscriber system

Alexandre Junqueira
31590
i31590@alunos.di.fc.ul.pt

David Reis
31601
i31601@alunos.di.fc.ul.pt

José Sousa
26666
i26666@alunos.di.fc.ul.pt

Resumo

O trabalho que a seguir se apresenta tem como objectivo criar uma nova solução para os sistemas de editor/subscritor (Publish/Subscribe). A solução proposta apresenta uma vertente inovadora que devido ao uso do Framework Appia [1], que fornece solução para alguns dos problemas de comunicação, vai permitir oferecer um serviço de alta qualidade aos utilizadores deste tipo de sistemas.

1 Introdução

Os sistemas de edição/subscrição[1] consistem na existência de um servidor onde os editores publicam mensagens e os subscritores registam-se para receber essas mensagens. Existem vários tipos de sistemas de edição-subscrição, entre os quais destacam-se: os que são baseados em tópicos e os baseados em conteúdos.

Num sistema baseado em tópicos as mensagens são publicadas e os subscritores associam-se aos tópicos que desejam receber. Os editores sempre que publicam algo, sobre esse tópico, é enviado a todos os subscritores nele registados. Num sistema edição-subscrição baseado no conteúdo os subscritores recebem todas as mensagens que tenham determinados atributos ou conteúdos definidos. Existem ainda os sistemas de edição-subscrição Híbridos em que os editores publicam as mensagens para um tópico e os subscritores registam-se para obter mensagens baseadas no conteúdo para um ou mais tópicos, as mensagens recebidas por estes serão um subconjunto filtrado com base no conteúdo.

O trabalho a realizar no âmbito da cadeira de Tolerância a Faltas Distribuídas consiste num sistema baseado em tópicos em que existem um conjunto de servidores que visam tornar o sistema tolerante a faltas.

[1] Do inglês Publish/Subscribe.

2 Trabalho Relacionado

Sobre o referido paradigma, Edição/Subscrição, existem várias soluções propostas, tal como esta de Kaiser [2], que usa uma resolução do problema com base diferente da que nos debrucamos. Contudo outra solução proposta onde contém uma das bases que usamos foi realizada pela equipa José Mocito, Liliana Rosa, Nuno Almeida, Hugo Mirana, Luís Rodrigues e Antónia Lopes [3]. O que nos motivou a escolher a framework Appia, foi um dos elementos dessa equipa, Luis Rodrigues, visto que por experiência própria deles, o Appia é uma ferramenta que nos ajudará a solucionar o problema. Também, ao longo do estudo e e pesquisa, observamos que nos vários tipos de soluções nenhuma apresenta uma solução idêntica á que nós pretendemos utilizar.

3 Descrição do Sistema

O sistema a concretizar irá oferecer serviços de publicação e subscrição. O sistema permitirá aos Editores anunciarem os assuntos que publicam e aos subscritores subscreverem a esses assuntos. Um processo coordenador estará encarregue de gerir os grupos, sendo toda a restante interação efecutada apenas entre Editores e Subscritores. O sistema funcionará de modo a que as mensagens publicadas sejam recebidas atempadamente, e com uma determinada QoS[2], pelos subscritores interessados nos assuntos. O coordenador será um processo servidor que encontra-se replicado por diversas máquinas com o intuito de tolerar faltas.

3.1 Appia

O Appia consiste numa framework de desenvolvimento e execução de protocolos constituido por uma

[2] **Q**uality **o**f **S**ervice

pilha que se encontra distribuída por camadas. A implementação do Appia oferece abstrações que implementão vários modelos de computação distribuída.

No nosso projecto iremos tirar uso de grande parte das ferramentas oferecidas por esta framework. Como é o caso dos protocolos de comunicação normais e comunicação em grupo [9] do Appia.

4 Operação entre componentes

4.1 Editor - Servidor Coordenador

Quando os editores pretendem publicar informação têm de se anunciar ao coordenador e indicar qual é o assunto e a qualidade de serviço dos eventos que o editor pretende publicar. O servidor em resposta atribui ao editor o nome do grupo a usar para esse fluxo. O editor pode ainda usar o método publicar que concretiza o envio de uma mensagem para o grupo em causa.

4.2 Servidor Coordenador- Servidor

A interacção entre servidores será fundamental para garantir a tolerância a faltas do sistema, os servidores terão de ter actualizada toda a informação sobre os grupos, os assuntos e a QoS associada. No inicio da sua comunicação, os servidores entram em acordo para seleccionar o servidor coordenador em que este será o responsável pela replicação das informações entre os restantes servidores.

4.3 Subscritor - Servidor Coordenador

De modo a conseguir subscrever-se a um assunto o subscritor envia uma mensagem ao coordenador através do método subscrever, este informa qual o assunto a que pretende receber notificações. O servidor responde com a QoS a utilizar e o nome do grupo. Com esta informação o subscritor irá criar uma camada na pilha do Appia com a QoS a utilizar e o grupo a que se irá juntar.

5 Conceitos relevantes

Alguns conceitos relevantes tomados em conta para a realização do projecto consistem na: Definição de grupo, Processo coordenador, Ordenação de mensagens, Qualidade de Serviço, Replicação.

5.1 Definição de grupo

Entende-se por grupo um sistema que oferece suporte a filiação, permite a inclusão e a exclusão de elementos no grupo e que oferece suporte a comunicação ou seja que permite a interacção entre elementos do grupo[3].

5.2 Processo coordenador

O processo coordenador entende-se como um processo que foi eleito e ao qual são atribuídas funções especiais como a prioridade de interacões com elementos externos ao grupo, a capacidade de decisão relativamente ao consenso, resolução de problemas de replicação, etc.

O uso de um processo coordenador no projecto torna-se necessário pois é a este que posteriormente se vão anunciar os editores e informar os subscritores.

5.3 Ordenação de mensagens

A ordenação de mensagens no projecto poderá ser necessária para determinar a ordem em que as mensagens foram enviadas para que desta seja possível estabeler uma relação de causa/efeito entre mensagens. Exitem vários modelos de ordenação de mensagens, dos quais podemos destacar: Ordem FIFO[4], Ordem Causal e Ordem Total. No projecto os modelos de ordenação de mensagens a adoptar são os disponibilizados pela Appia Framework, ou seja, Ordem Total(com sequênciador mais token), Ordem Total Optimista, Ordem Causal e Ordem Total Causal.

5.4 Qualidade de Serviço (QoS)

No trabalho será possivel utilizar duas qualidades de serviço, as qualidades de serviço que são disponibilizadas pelo appia, nomeadamente ordem total e sincronia na vista.

5.4.1 Sincronia virtual

Consiste num mecanismo que oferece suporte de filiação e comunicação por broadcast fiável. A sincronia virtual permite num determinado momento saber quantos e quais os elementos que participam num grupo[5].

Sempre que um elemento entra ou sai do grupo a sincronia virtual encarrega-se de entregar uma vista com os novos elementos ou com os elementos que actualmente estão no grupo.

[3] No trabalho em questão permite que um editor possa publicar um assunto para todos os que subscreveram esse assunto.

[4] **F**irst **I**n **F**irst **O**ut

[5] Elementos que pertencem à vista de grupo

A sincronia virtual oferece garantias que todas as mensagens que pertencem a uma vista anterior são entregues nessa vista antes que uma nova vista seja instalada.

5.4.2 Ordem Total

Utiliza o mecanismo da sincronia virtual e sobre este ainda oferece que as mensagens são entregues na ordem em que foram enviadas.

5.5 Replicação

Existem três tipos base de replicação:

1. Activa: em que os servidores estão constantemente a trocar informação sobre as listas que associam assunto ao grupo e a qualidade de serviço pretendida(QoS), este modo de replicação permite que todos os servidores possuam a "mesma lista". Possui ainda a vantagem de poder ser utilizado balanceamento de carga entre réplicas.
2. Semi-Activa: que é semelhante à anterior, pois continuam a trocar as listas, mas que o processo coordenador é responsável por indicar qual é a lista correcta. Perante a falha do coordenador é iniciado um processo de eleição onde é elegido um novo coordenador entre as restantes réplicas.
3. Passiva: O processo coordenador detêm a lista actualizada(assunto,grupo,QoS) e inicia o processo de replicação disseminando a sua lista pelas réplicas que aguardam actualizações periódicas para assim actualizarem as suas listas locais. Se o processo coordenador falhar é eleito um novo coordenador entre as réplicas restantes que tenham o seu estado correcto[6].

6 Modelo de Faltas

As faltas que possam ocorrer pertecem a dois grupos:

- Faltas omissivas: que pertecem ao dominio do tempo, ou seja, uma mensagem chega fora do tempo previsto ou nunca chega;
- Faltas afirmativas: que pertecem ao dominio do valor em que a mensagem enviada chega dentro do tempo previsto mas contém informação errada. As faltas afirmativas ainda podem ser subdivididas em dois subgrupos, nomeadamente:
  1. Semânticas;
  2. Sintáticas.

Não é difícil reconhecer que é praticamente impossível criar um sistema que tolere todo o tipo de faltas. No projecto será utilizado o sistema Appia que fornece formas de ultrapassar algumas das faltas que possam ocorrer, para todas as outras faltas é necessário desenvolver um modelo onde se especifique as faltas que se pretendem tolerar. No projecto identificou-se como necessário tolerar faltas afirmativas, ou seja, garantir que quem recebe as mensagens recebe-as correctamente.

7 Arquitectura Projecto

A seguir vamos apresentar a arquitectura concreta do nosso sistema e os seus principais pormenores.

- Será utilizado o serviço de Gossip da Appia Framework para a fornecer os serviços de grupo. No desenvolvimento do projecto é assumido por defeito que este serviço é tolerante a faltas.
- Todos os servidores fazem parte de um grupo implementado com a Appia framework. Um dos servidores é automaticamente eleito coordenador pela camada de comunicação em grupo.
- Um Editor pode ser Subscritor e vice versa, e têm ao seu dispor os serviços de anunciar, publicar e subscrever. Apenas os Subscritores receberão as notificações. Deverão ainda ter a possibilidade de conhecer os grupos existentes.
- Os Editores/Subscritores também passarão a pertencer a um grupo sempre que se tenham subscrito ou anunciem neste. Os Editores/Subscritores contactam o coordenador com o fim de obter os serviços fornecidos por este.
- A comunicação entre os Editores/Subscritores e coordenador é efectuada através de comunicação ponto-a-ponto fornecido pela Appia Framework.
- O serviço de Publish/Subscriber terá um máximo de grupos suportados. Quando este limite for atingido o coordenador irá reutilizar um grupo existente com a mesma QoS. Os Editores/Subscritores irão rejeitar as notificações para as quais não estão subscritos.
- Os Editores/Subscritores poderão falhar ou sair do grupo sem que isso afecte o funcionamento do sistema, ou elimine o grupo.

[6] Entende-se por estado correcto a posse de uma lista correctamente actualizada

- O Coordenador sempre que recebe novos dados, dissemina pelas suas réplicas a actualização. No caso do Coordenador falhar o serviço deverá continuar a funcionar enquanto existirem outras réplicas que possam assumir o seu lugar. Os Editores/Subscritores deverão passar a conhecer o novo Coordenador através de funcionalidades fornecidas pelo serviço de Gossip.
- No caso de um novo servidor se juntar ao grupo este irá inicialmente contactar o coordenador e obter todos os dados replicados de forma a ficar num estado coerente com os as restantes réplicas.

Todos os restantes pormenores e funcionalidades do sistema serão decididos em tempo de implementação pois ainda será necessario conhecer melhor as capacidades da Appia Framework.

7.1 Pilhas Protocolares

Depois de estudarmos as características do Appia e as camadas fornecidas por este optamos por criar duas pilhas protocolares [8].

No nosso sistema tanto os Servidores como os Editores/Subscritores necessitam de utilizar os serviços do *GossipServer* para criação dos grupos. Esses serviços são fornecidos pela camada *GossipOut Layer*. Partimos do principio também que o serviço de Gossip é tolerante a faltas.

No servidor optamos por uma pilha em que vamos apenas utilizar um canal de comunicação:

PubSubServer Layer
VSync Layer
Leave Layer
Stable Layer
Heal Layer
Inter Layer
Intra Layer
Suspect Layer
GossipOut Layer
Bottom Layer
TcpComplete Layer

Os micro-protocolos da pilha Appia escolhidos foram os que fornecem comunicação em grupo, e na ligação *TcpComplete*. Decidimos usar comunicação em grupo fornecida pela Appia Framework pois esta fornece uma noção de grupo e de comunicação em grupo. A comunicação em grupo ainda nos fornece sincronia na vista que pensamos ser suficiente para as nossas necessidades pois utilizamos um servidor coordenador. Optamos ainda por utilizar para ligações o protocolo *TcpComplete*, porque pensamos ser o mais ideal para o nosso sistema. A camada de *PubSubServer* irá tratar as operações efectuadas pelos Editores/Subscritores e Coordenador caso este seja o caso. Esta camada faz uso dos protocolos inferiores, recebendo os eventos destes e passando a estes quando necessario.

Para os Editores/Subscritores optamos por utilizar uma pilha que fornece 2 canais de comunicação:

Publisher Layer	
Subscriber Layer	
	Uniform Layer
VSync Layer	
Leave Layer	
Stable Layer	
Heal Layer	
Inter Layer	
Intra Layer	
Suspect Layer	
GossipOut Layer	
RemoteView Layer	
Bottom Layer	
TcpComplete Layer	

Para o caso dos Editores/Subscritores optamos por escolher esta pilha protocolar com 2 canais pois necessitamos de fornecer 2 QoS, num canal utilizaremos sincronia na vista e noutro ordem total. A camada responsável por fornecer o serviço de ordem total é a *Uniform Layer*, o serviço de sincronia na vista já é fornecido por defeito pela pilha de comunicação em grupo. A camada de comunicação em grupo será necessária para efectuar as notificações dentro dos diversos grupos. Optamos também por utilizar o protocolo de ligação *TcpComplete*. Utilizamos a camada *RemoteView* para que seja possível aos Editores/Subscritores contactarem um servidor. A pilha do Editor/Subscritor irá conter ainda duas camadas aplicação a *Publisher* e a *Subscriber*. A camada *Publisher* será responsável por tratar as operações do Editor e a camada *Subscriber* por tratar as operações do Subscritor. Ambas terão ainda interface com o utilizador.

8 Conclusão

Sabendo que um sistema distribuído deve oferecer varias propriedades, tais como: fiabiliade, confiabilidade e disponibilidade. No desenvolvimento do trabalho, criação de um sistema de Editor/Subscritor (Publish/Subscribe), e com vista a obter uma solução tolerante a faltas, foi dado especial destaque às propriedades enunciadas anteriormente. A solução proposta utiliza a Framework Appia que oferece solução para alguns

dos problemas de comunicação, e que facilita a compreensão desta. A solução apresentada, desenvolveu-se por um caminho diferente do utilizado nas soluções préviamente existentes de outros autores que se debruçaram sobre este mesmo paradigma.

Referências

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

[2] J.Kaiser, M.Mock, Implementing the Real-Time Publisher/Subscriber Model on the Controller Area Network (CAN) http://ieeexplore.ieee.org/iel5/6308/16866/00776373.pdf

[3] Context Adaptation of the Communication Stack, http://micas.di.fc.ul.pt/ICDCS.pdf

[4] http://www.javaworld.com/javaworld/jw-11-2001/jw-1109-subscriber.html

[5] http://en.wikipedia.org/wiki/Publish/subscribe

[6] http://www.vico.org/pages/PatronsDisseny/Pattern%20Publisher%20Subscriber/index.html

[7] http://portal.acm.org/citation.cfm?id=1028625

[8] http://appia.di.fc.ul.pt/docs/appia-pdm-2-1.pdf

[9] http://appia.di.fc.ul.pt/docs/groupman.pdf