Primeiro Relatório de Tolerância a Faltas
Grupo TFD01

João Costa
Número 30329

Luís Fura
Número 31591

Sónia Alves
Número 31641

Sumário

Actualmente, a maioria dos sistemas de edição-subscrição já são tolerantes a faltas.
Neste relatório iremos propor uma possível solução para um sistema de edição-subscrição simplificado. Com o auxílio das funcionalidades do Appia iremos construir um sistema fiável e tolerante a faltas.

1. Introdução

Neste artigo iremos explicar as características do nosso sistema de edição-susbcrição simplificado e adiante iremos dar uma noção do conceito de edição-subscrição e de como é feita a comunicação entre os editores, subscritores e o servidor.

O nosso sistema será constituido por editores, subscritores e um servidor, que por sua vez se encontrará replicado em várias máquinas. Tentaremos dar uma visão geral do trabalho que já foi feito nesta área, explicando alguns desses sistemas e também como os subscritores se juntarão a um grupo para a partir daí receberem os assuntos que lhes interessem. Também iremos explicar quais as funções que os servidores irão ter no nosso sistema, o modo como a informação é transferida pelo editor para o subscritor.

Os protocolos do Appia [1] serão um importante auxílio na execução de todo este sistema porque estes irão garantir a fiabilidade e a tolerância a faltas do mesmo.

Na secção 2 será apresentado o trabalho relacionado, no qual apresentamos informação sobre trabalhos já existentes.

Na secção 3 apresentaremos o tema do artigo, onde explicaremos os objectivos do nosso trabalho.

Na secção 4 será possível encontrar a nossa proposta de implementação do problema edição-subscrição.

Na secção 5 iremos dar ideias para um possível trabalho futuro.

Na secção 6 serão apresentadas as nossas conclusões.

Finalmente, na última secção iremos apontar as referências usadas para a concretização da proposta apresentada neste trabalho.

1.1. Contexto

Um sistema de comunicação edição-subscrição é baseado na troca assíncrona de mensagens, conhecidas como eventos, o que permitem uma maior escalabilidade e uma maior topologia de rede dinâmica. Edição-subscrição é semelhante ao paradigma da fila de mensagens[17]. O JMS [2, 3], por exemplo suporta tanto edição-subscrição como modelos de filas de mensagens.

Este sistema consiste num conjunto de clientes que publicam eventos (editores), os quais são encaminhados para clientes que registaram interesse em recebê-los (subscritores).

Num sistema baseado nos tópicos, as mensagens são publicadas por tópico ou pelo nome dos canais que estão no ”broker”. O Appia[4] será o nosso ”broker”, o ”broker” é o programa intermediário que efectua a tradução das mensagens enviadas para um formato de mensagens do protocolo Appia, desta forma o editor e o subscritor comunicam com o mesmo formato de mensagem.

Os subscritores irão receber todas as mensagens publicadas correspondentes ao(s) tópico(s) subscrito(s) e todos os subscritores irão receber as mesmas mensagens.

Num sistema baseado no conteúdo, as mensagens são apenas entregues aos subscritores se o conteúdo das mensagens corresponder aos parâmetros definidos por uma ou mais subscrições desses subscritores.

Vantagens:

- *Anonimato*: Não é necessário os editores conhecerem os subscritores.
- *Escalabilidade*: O sistema tem o mesmo comportamento quer sejam muitos ou poucos clientes no sistema.

Desvantagens:

- *Sobrecarregamento da rede*: Devido às mensagens serem enviadas em broadcast ou multicast inunda a rede com várias mensagens desnecessárias.

1.2. Tema do Trabalho

O tema do nosso trabalho tem como pontos fortes usar o sistema Appia[4] para garantir as qualidades de serviços propostas: ordem total e sincronia na vista. O Appia[4] também oferece comunicação em broadcast e ponto-a-ponto fiável o que garante a entrega de todas as mensagens enviadas.

O nosso servidor encontra-se replicado em várias máquinas o que garante que o nosso servidor seja tolerante a faltas, pois se um dos servidores for dado como ”morto” haverá outros que continuarão a oferecer os serviços aos editores e subscritores. As mensagens são enviadas pelos editores e pelos subscritores em broadcast para os servidores, o que garante que haja sempre algum servidor que responda.

O nosso sistema não é persistente, ou seja, quando a informação é enviada só os subscritores activos recebem as mensagens, os que ”morreram” nunca mais voltam à ”vida”, isto é uma vantagem, pois não temos de guardar em memória permanente as mensagens anteriormente enviadas.

O subscritor irá guardar em memória quais os seus assuntos que subscreveu, assim quando receber mensagens que não são dos assuntos pretendidos então essas mensagens irão ser descartadas.

2. Trabalho Relacionado

Para termos uma ideia de como são construídos os sistemas de edição-subscrição fizemos um estudo prévio de alguns trabalhos já efectuados.

Existem vários sistemas como: Gryphon[5], SIFT[6, 7], Siena[8], Le Subscribe[12], Herald[9], XMLBlaster[10], Elvin4[14], Keryx[11] entre outros. De todos estes sistemas iremos falar do Gryphon e do Siena.

O Gryphon[5] foi concebido pela IBM em 1997 sendo usado na Internet para resultado de desportos em tempo-real eventos de Grand Slam em ténis, Ryder Cup e reportou as estatísticas dos Jogos Olímpicos de Sydney. Este sistema foi implementado em Java Message Service(JMS)[2].

A troca de mensagens entre o editor e subscritor são anónimas, o Gryphon dá particular atenção a 3 propriedades: a escalabilidade, a disponibilidade e a segurança do sistema. Para garantir estas propriedades usa ”Server Farms” em vários pontos do Mundo o que garante a escalabilidade, redirecciona o tráfego se houver alguma falha na rede, este processo é automático e não é necessário a intervenção de um administrador, o que garante a disponibilidade e por fim usa SSL na troca das mensagens entre editor e subscritor, o que garante a segurança na comunicação.

O SIENA[8] é um sistema de notificação de eventos que adopta uma estrutura de servidores ligados entre si que propagam as notificações e servem de ponto de acesso, através dos quais os clientes podem registar os seus interesses e receberem as notificações de eventos. Esses clientes são divididos em duas categorias, os objectos de interesse, que são as fontes de eventos (editores), e as partes interessadas, que ”consomem” as notificações(subscritores).

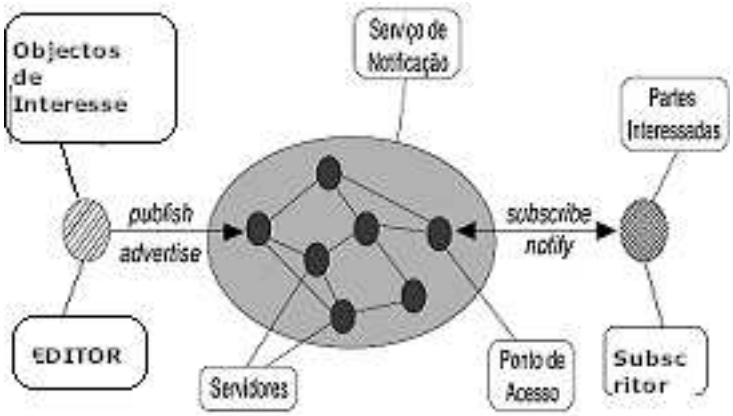

Figura 1: Sistema SIENA

3. Tema do Artigo

Como já foi referido, o sistema a desenvolver é constituído por editores (que submetem assuntos), subscritores (que subscrevem esses assuntos) e os servidores (que gerem a comunicação entre os editores e subscritores).

Cada assunto tem uma qualidade de serviço associada que é indicada ao sistema pelos editores. Neste caso, existem duas qualidades de serviço: sincronia na vista e ordem total, ambas são armazenadas na pilha do Appia, que presta essa qualidade.

O Appia[4] permite comunicação em grupo[15], pelo que neste trabalho toda a comunicação vai ser em grupo, em que cada um terá um nome correspondente e é escolhido pelo sistema de edição-subscrição.

Os editores podem efectuar duas operações: editar e publicar. Na primeira, o editor indica o assunto e a qualidade de serviço ao sistema e este atribui o nome do grupo. Na segunda, é apenas enviada uma mensagem para o grupo em causa. Os subscritores so podem efectuar a operação de subscrição, que é feita através do envio duma mensagem com o assunto que este pretende subscrever, e em resposta o sistema envia-lhe a qualidade de serviço e o nome do grupo. Os servidores replicados fazem a associação dos assuntos, qualidades de serviço e os nomes dos grupos, e estão configurados para não atribuir mais do que N nomes de grupos(que deve ser superior ao número de qualidades de serviço disponíveis). Caso isto não aconteça, um grupo pode conter vários assuntos.

4. Proposta para uma possível concretização

4.1. Sistema

Este sistema possui login para editores e subscritores, para os podermos distinguir.

Cada assunto tem a sua vista, se o número de assuntos for menor que N (N é o número de grupos possíveis como já foi referido no tema do artigo). O N terá um valor de 2 para efectuar testes, pois temos apenas duas qualidades de serviço.

Se o número de assuntos não for menor que N, então o assunto vai ser associado a uma vista já existente. Associamos à vista com menos elementos, porque desta forma nao é preciso enviar informação a tantos subscritores que não estejam interessados nesse assunto, ou se possível, associar o assunto a uma vista onde o editor que publica já esteja contido.

4.2. Comunicação

Os editores e subscritores comunicam com os servidores em broadcast, recebendo as mensagens de retorno com comunicação ponto-a-ponto. Os editores enviam para a vista a que está associada o assunto que querem enviar, em broadcast para todos os elementos dessa vista.

As mensagens podem ser enviadas com duas qualidades de serviço: ordem total e sincronia na vista.

O nosso sistema é não persistente, isto é, se um editor ou subscritor falharem, entao nunca poderão voltar à vida, o mesmo acontece com os servidores. Assim, não é necessário armazenar a informação editada em memória persistente.

Toda a comunicação trocada no Appia é efectuada em XML[16].

4.3. Editor

Quando um editor quiser publicar tem de perguntar previamente ao servidor se esse assunto já existe. Se existir, o servidor retorna o nome do grupo, ao qual o assunto está associado. Assim o editor junta-se à vista e publica, através dos protocolos do Appia[1].

Quando o editor quer anunciar envia para o servidor o assunto e a qualidade de serviço dos eventos. O servidor tem em memória todos os assuntos existentes. De seguida recebe o nome do grupo ao qual se deve associar e então o editor junta-se(join) à vista. Por fim o editor envia para a vista o assunto correspondente.

O editor não necessita saber o número de subscritores.

4.4. Subscritor

Quando um subscritor quiser subscrever um assunto, adiciona ao seu array o assunto que quer receber. Assim quando receber assuntos não desejados, irá descartá-los e comunicar com o servidor para saber a que vista se deve juntar. O subscritor filtra os assuntos que não lhe interessam, porque quando os recebe vai ao seu array de assuntos verificar se é um assunto que subscreveu.

4.5. Servidor

Como já foi referido, o servidor vai ser replicado em várias máquinas, as quais guardam as qualidades de serviço, nome dos grupos e a associação entre estes, e a lista dos assuntos.

Quando um subscritor pede para subscrever um assunto, o servidor retorna o nome do grupo ao qual o subscritor se deve juntar.

Quando um editor quer publicar, o servidor retorna o nome do grupo ao qual o editor se vai juntar. O servidor tem de associar o id único do grupo ao nome do grupo, isto faz com que após o servidor devolva o nome do grupo, o editor retorne o id do grupo para o servidor, para fazer tradução entre o id do grupo e o nome do grupo.

4.6. Tolerância a Faltas

Como o servidor se encontra replicado, se houver uma falha existe sempre outro servidor que preste serviço.

Se um dos editores ou subscritores falhar o Appia garante que as mensagens são enviadas/entregues.

4.7. Appia

O Appia[4] oferece protocolos[1] que garantem:

- Difusão fiável
- Comunicação Ponto-a-Ponto fiável
- Ordem total nas mensagens (é oferecida pela classe TotalOrderEvents)
- Sincronia na vista na comunicação em grupo[15]

Os servidores correm a partir dos protocolos Gossip (GossipGroupEvent)[13].

O Appia garante tolerância a faltas.

5. Trabalho Futuro

Futuramente esperamos conseguir implementar esta proposta de solução do problema de modo a alcançar todos os objectivos traçados, usando como principais ferramentas os protocolos do Appia, que irão ser muito úteis neste

tipo de comunicação, porque garante todos os requisitos de tolerância a faltas.

Sem o sistema concebido torna-se difícil decidir o que podia ser adicionado no trabalho, no relatório final já teremos uma visão mais futura de como poderia ser elaborado um trabalho futuro.

6. Conclusões

Neste artigo explicamos as características do nosso sistema de edição-susbcrição simplificado constituído por editores, subscritores e servidor, este servidor foi replicado por várias máquinas para o sistema ser tolerante a faltas.
A comunicação que existe entre editores, subscritores e os vários servidores também foi resumida, bem como a distribuição dos vários assuntos pelos grupos existentes quando já não existem grupos disponíveis. Também explicamos como é possível conceber um sistema tolerante a faltas com o auxílio dos protocolos Appia[1] que fornecem todas as funcionalidade necessárias e ainda demos uma ideia do que é realmente edição-subscrição.

Descrevemos os serviços que o Appia nos oferece para a elaboração do nosso sistema edição-subscrição.

Mostramos que já existem vários sistemas edição-subscrição concebidos com sucesso, ou seja, tolerante a faltas,com grande capacidade de escabilidade e seguros.

Propusemos um possível trabalho futuro, que iremos com certeza implementar.

7. Bibliografia

References

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

[2] http://java.sun.com/products/jms/

[3] http://pt.wikipedia.org/wiki/JMS

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

[5] http://www.research.ibm.com/distributedmessaging/gryphon.html

[6] SIFT- A Tool for Wide-Area Information Dissemination Yan T., 1999

[7] SIFT Information Dissemination Hector Garcia-Molina, Tak W. Yan, 1999

[8] ”Serviços de Notificação de Eventos Baseados em Publish/Subscribe”, Bruno Oliveira Silvestre, 2005

[9] http://research.microsoft.com/sn/herald

[10] http://www.xmlblaster.org

[11] http://keryxsoft.hpl.hp.com

[12] http://www-caravel.inria.fr/ pubsub

[13] A Gossip Protocol for Subgroup Multicast, Kate Jenkins, Ken Hopkinson and Ken Birman

[14] Awareness and Agility for Autonomic Distributed Systems: Platform-Independent Publish-Subscribe Event-Based Communication for Mobile Agents,Amir Padovitz, Arkady Zaslavsky, Seng Wai Loke

[15] Appia Group Communication, Alexandre Pinto, 2005

[16] AppiaXML,A Brief Tutorial,José Mocito,Liliana Rosa,Luís Rodrigues and Nuno Almeida, 2004

[17] http://en.wikipedia.org/wiki/Message_queue