Sistema de Edição-Subscrição

Pedro Fonseca
i29837@alunos.di.fc.ul.pt

Sérgio Ildefonso
i26615@alunos.di.fc.ul.pt

Bernardo Mendonça
i29325@alunos.di.fc.ul.pt

Abstract

A criação de aplicações distribuídas é, cada vez mais, um assunto de estudo interessante.

Ao se considerar um sistema distribuído fiável, deve-se ter em conta alguns factores relacionados com a confiança. Ou seja, é depositada confiança no sistema assumindo que este se irá comportar provavelmente de uma maneira esperada. A utilização de arquitecturas de sistemas distribuídos propicia a existência de alguns factores que, por vezes, põem em causa o funcionamento correcto do próprio sistema. Um sistema distribuído fiável deve ser capaz de lidar com as (possíveis) falhas de um ou mais dos seus constituintes. Quando um elemento apresenta falhas, normalmente apresenta um comportamento que é negligenciado pelos outros, uma vez que, em princípio, envia informações que são discordantes com as partilhadas pelos outros.

1. Introdução

O paradigma de edição-subscrição de informação permite que a comunicação seja distribuída e desligada de entre vários processos fazendo com que apenas a informação passe a ter o papel de elo de ligação entre os vários componentes de um sistema distribuído. Como os editores estão ligados indirectamente aos subscritores, podem nem sequer conhecer a sua existência.

O modelo de edição-subscrição permite a troca de mensagens através de um canal, que pode ser persistente[1] ou volátil. Num sistema de edição-subscrição existem processos que produzem mensagens para o canal, os *editores*, e processos que consomem mensagens nesse canal, os *subscritores*. Se considerarmos uma implementação com um servidor centralizado, o papel do editor consiste apenas em enviar mensagens para o servidor; já o subscritor pode optar por duas estratégias [5, Sec. 3.9.2], *push* ou *pull*. Na primeira os subscritores registam no servidor o tipo de mensagens que querem receber e este é responsável pela sua propagação para os subscritores interessados. No segundo caso, os subscritores devem contactar o servidor periódicamente para receber mensagens. Esta arquitectura baseada num servidor centralizado tem problemas de desempenho e tolerância a faltas.

No sistema que pretendemos desenvolver, a disseminação de eventos entre editores e subscritores não depende de um servidor centralizado, e pode recorrer a Qualidades de Serviço (QdS) diferentes, de acordo com o assunto. Os editores publicam informação sobre a forma de eventos que será recebida em tempo real por todos os subscritores interessados. Embora dependa necessáriamente de um servidor para manter a associação entre assuntos e QdS, este sistema apresenta uma tolerância a faltas claramente superior a uma aproximação baseada num servidor centralizado. Este sistema será concretizado com recurso aos serviços de comunicação em grupo fornecidos pelo *Appia*, descritos em [3].

O resto deste artigo apresenta-se organizado da seguinte forma:

A secção 2 discute o tipo de faltas que pretendemos suportar e como o fazer; a secção 3 refere as várias hipóteses para a replicação de servidores e qual a mais conveniente no âmbito deste projecto; na secção 4 apresentamos uma descrição geral do sistema; a secção 5 refere com algum detalhe que pilhas de protocolos serão usadas na implementação do sistema. As conclusões encontram-se em 6.

2. Modelo de Faltas

Um sistema computacional fiável deve ser capaz de lidar com as possíveis falhas de um ou mais dos seus constituintes. Quando um elemento apresenta falhas, normalmente apresenta um comportamento que é negligenciado pelos outros, uma vez que, em princípio, envia informações que são discordantes com as partilhadas pelos outros. Faltas arbitrárias e partições na rede não serão consideradas para este projecto. No entanto existem outros tipos de faltas que são consideradas e às quais tem que ser dada a devida importância e tratamento adequado. As faltas omissivas correspondem a faltas nas quais um elemento passa para um estado omisso, ou seja deixa de participar activamente no sistema (ou porque perdeu a ligação ao mesmo, ou porque

[1] e.g. Usenet

deixou de funcionar correctamente, ou ainda por um outro motivo qualquer não previsto). As faltas omissivas podem ser temporais (e neste caso um componente atrasa-se numa acção), por omissão (e neste caso uma das acções de um componente é omitida) ou por paragem (e neste caso o componente pára). As faltas assertivas correspondem às faltas nas quais um componente comporta-se de uma forma não especificada, mas não omissa. As faltas assertivas podem ser de dois tipos: faltas sintáticas (e neste caso as faltas são caracterizadas por um ou mais erros de sintaxe, como por exemplo, tipos de dados inesperados) e faltas semânticas (e neste caso são caracterizadas por erros de semântica, como por exemplo valores inválidos, mas de tipos válidos). Considerando que não acontecem partições na rede, as faltas omissivas podem ser detectadas pelo mecanismo de gestão de grupos já implementado no Appia. No caso das faltas assertivas:

- Sintácticas: as mensagens são descartadas pois não correspondem a dados válidos. No entanto, o estado dos componentes que as enviam é registado.
- Semânticas: as mensagens são aceites, mas ocorre sobre elas um mecanismo de validação, uma vez que o formato é correcto. Mesmo sendo aceites, durante o processo de validação podem ser desacreditadas, tendo em conta o contexto.

3. Modelo de Replicação

Dado o modelo de faltas proposto, existem três modelos [5, Sec. 7.6] possíveis para a resolução do problema: *replicação activa*, *replicação semi-activa* e *replicação passiva*.

No modelo de replicação activa não é necessário que um servidor conheça as suas réplicas. Por outro lado, implica a existência de um mecanismo de consolidação nos clientes, que só é trivial enquanto considerarmos faltas omissivas, e um canal de difusão fiável que suporte ordenação total das mensagens. O modelo de replicação semi-activa dispensa a ordenação total das mensagens, já que existe um coordenador, mas todas as réplicas continuam a executar comandos. Já no modelo de replicação passiva o estado das réplicas só é actualizado periódicamente pelo coordenador.

Neste caso concreto, consideramos que é mais vantajoso usar um modelo semi-passivo ou passivo, já que a complexidade adicional no cliente, e também o custo de suportar ordenação total na comunicação com os servidores, não compensa a ausência de comunicação entre réplicas. Sendo assim, os servidores replicados terão de decidir quem é o coordenador e comunicar entre si. Para eleger o coordenador podemos usar a antiguidade como critério. Assim, o primeiro servidor a juntar-se à vista será sempre o coordenador; caso este falhe, será eleito o servidor seguinte. Em alternativa, o novo coordenador também pode ser escolhido aleatóriamente.

4. Arquitectura

4.1. Comunicação

Existem várias primitivas de comunicação. A comunicação ponto-a-ponto permite a comunicação entre os servidores e os clientes utilizando uma camada na qual as mensagens de um processo são recebidas pela ordem que são enviadas - FIFO. A comunicação em grupo é um modo de comunicação que permite a comunicação entre servidores e clientes através de multicast ou broadcast. A sincronia virtual é um paradigma de comunicação em grupo em que um conjunto de processos são organizados numa vista. Quando existe uma alteração nos elementos do grupo é enviada a todos os elementos do grupo uma vista nova. A sincronia virtual garante que numa vista todos os processos correctos recebem o mesmo conjunto de mensagens. A difusão atómica garante a entrega de mensagens a todos os processos do grupo que estão correctos.

Como já foi referido anteriormente, os editores publicam eventos que são consumidos pelos subscritores interessados, sem recorrer a um servidor. Esta publicação de eventos é concretizada pelo método *publicar* dos editores. Os editores também usam o método *anunciar* para indicar ao servidor que atribui nomes de grupos o assunto e a Qualidade de Serviço (QdS) desejada. Um subscritor obtém o nome de grupo a usar indicando ao servidor o assunto sobre o qual está interessado em receber informação, através do método *subscrever*.

Para ser possível ”descobrir”os servidores existentes, tanto editores e subscritores como os servidores de nomes de grupos têm de contactar um serviço externo, num endereço conhecido. Este serviço já está concretizado no Appia em *GossipServer*, e proporciona um mecanismo de *pseudo-difusão*, que retransmite mensagens para todos os processos conhecidos [3, Sec. 2.4]

4.2. Notificações Não Solicitadas

Se existirem mais assuntos do que grupos, vários assuntos podem necessitar de ser disseminados usando o mesmo grupo. Nesse caso, do lado do cliente, será necessário filtrar as notificações não solicitadas. Para limitar o número de notificações não solicitadas que são recebidas (mas filtradas) nos subscritores. Para isso é necessária a criação de uma nova camada intermédia entre a aplicação e uma das camadas de entrega de notificações.

4.3. Configuração de Qualidade de Serviço

A configuração das pilhas do Appia que serão usadas, dependendo da QdS, vai ser facilitada pelas funcionalidades do AppiaXML [2].

Os servidores que suportam o sistema não devem atribuir mais do que *N* nomes de grupos. Tal como os grupos de editores e subscritores usam XML para configurar a pilha de protocolos do Appia, os servidores também podem usar XML para configurar a sua própria pilha. O parâmetro de configuração *N* pode ser definido no ficheiro XML com a configuração da pilha dos servidores.

5. Pilhas Protocolares do Appia

Para cada Qualidade de Serviço (QdS) oferecida pelo sistema será necessário uma pilha de protocolos diferente. Os servidores, que vão ser replicados, também terão uma pilha protocolar distinta.

5.1. Sincronia na Vista

A pilha que oferece uma QdS que assegure sincronia na vista será constituída por uma camada com a lógica da aplicação, o conjunto de protocolos do Appia que oferecem suporte a comunicação em grupos com sincronia na vista [3] e por um protocolo que ofereça FIFO fiável, *TcpCompleteLayer*.

A camada *FilterLayer* é responsável por filtrar notificações não solicitadas, quando vários assuntos forem disseminados usando o mesmo grupo.

A camada *RemoteViewLayer* é necessária para obter um Grupo que não seja o próprio [1], ou seja, para um grupo de editores e subscritores descobrirem os servidores que mantêm as associações entre nomes de grupos e QdS.

5.2. Ordem Total

Para oferecer ordem total, será necessário um protocolo de ordem, que vai usar também os protocolos de sincronia virtual.

5.3. Servidor

O servidor também vai precisar dos protocolos que oferecem sincronia na vista, para garantir que o coordenador actualiza correctamente o estado de todas as réplicas. Também vai tirar partido do detector de falhas do Appia, concretizado em *SuspectLayer*.

AppLayer
Filter Layer
VsyncLayer
StableLayer
HealLayer
InterLayer
IntraLayer
SuspectLayer
MergeOutLayer
GossipOutExtLayer
BottomLayer
RemoteViewLayer
TcpCompleteLayer

Tabela 1. Pilha do Appia para editores e subscritores, com ordem total

AppLayer
Filter Layer
TotalOrderLayer
VsyncLayer
StableLayer
HealLayer
InterLayer
IntraLayer
SuspectLayer
MergeOutLayer
GossipOutExtLayer
BottomLayer
RemoteViewLayer
TcpCompleteLayer

Tabela 2. Pilha do Appia para editores e subscritores, com sincronia na vista

6. Conclusão

Os sistemas de edição-subscrição distribuídos que dependem de um servidor para disseminar eventos, têm uma fiabilidade baixa, já que existe um único ponto de falha. O sistema proposto neste artigo apresenta soluções para maximizar a fiabilidade deste tipo de sistemas. Embora esta proposta ainda dependa de um servidor, para associar grupos de comunicação a qualidades de serviço distintas, a tolerância a faltas consegue ser garantida de várias formas. Os servidores de nomes são replicados e faltas omissivas são toleradas transparentemente pelas funcionalidades de comunicação em grupo fornecidas pelo Appia. Foram propostas soluções para tolerar faltas assertivas: no

ServerLayer
VsyncLayer
StableLayer
HealLayer
InterLayer
IntraLayer
SuspectLayer
MergeOutLayer
GossipOutExtLayer
BottomLayer
TcpCompleteLayer

Tabela 3. Pilha do Appia para servidores

caso das faltas sintácticas as mensagens são descartadas; no caso de faltas semânticas estas passam por um processo de validação.

Referências

[1] Appia documentation. Javadoc.

[2] J. Mocito, L. Rosa, N. Almeida, and L. Rodrigues. *AppiaXML: A Brief Tutorial*. Faculdade de Ciências da Universidade de Lisboa, Sep 2004.

[3] A. Pinto. *Appia Group Communication*, Oct 2005.

[4] A. Pinto. Concretização do suporte à comunicação grupo no sistema appia. Slides da aula, 2006.

[5] P. Veríssimo and L. Rodrigues. *Distributed Systems for System Architects*. Kluwer Academic Publishers, 2001.