FTPS
Sistema de Edição/Subscrição Tolerante a Faltas

Hugo Pinto
23112
hmmpinto@gmail.com

Luís Fernandes
20102
luisfmfernandes@gmail.com

Resumo

Com o advento da Internet, a dimensão potencial dos sistemas distribuídos foi consideravelmente aumentada, englobando estes cada vez mais entidades, geograficamente distribuídas e móveis, onde o factor escalabilidade é preponderante.

As comunicações ponto-a-ponto e síncronas levam à criação de aplicações rígidas, dificultando o desenvolvimento de sistemas de larga escala. O paradigma de interacção edição/subscrição fornece um poderoso modelo de comunicação particularmente bem adaptado às arquitecturas distribuídas com requisitos de disseminação de informação. A principal vantagem apresentada por este paradigma reside na total separação da comunicação em termos de tempo, espaço e sincronização.

Neste artigo apresentamos um sistema de edição/subscrição simplificado, o FTPS de arquitectura distribuída, baseado em tópicos, com qualidade de serviço (QoS) e características de tolerância a faltas.

1. Introdução

Os sistemas *edição/subscrição* estão especialmente adaptados à natureza da interacção distribuída em aplicações de larga escala, devido ao seu modelo de interacção iminentemente assíncrono.

A interacção baseada em eventos que os sistemas de *edição/subscrição* proporcionam oferece a separação da comunicação em termos de tempo, espaço e sincronização. A separação em espaço permite que os intervenientes que estão a interagir não necessitem de se conhecer mutuamente. A separação em tempo permite que a interacção não seja feita necessariamente ao mesmo tempo, permitindo mesmo que o subscritor esteja desligado no momento da publicação ou o editor no momento da notificação. A separação de sincronização, permite que os eventos sejam gerados de forma assíncrona, não sendo necessário que os clientes (editores ou subscritores) permaneçam bloqueados [1, 6].

Normalmente, neste tipo de sistemas, existem três tipos de actores: os *editores*, que produzem informação, os *subscritores*, que consomem a informação produzida e os *brokers*, entidades mediadoras que asseguram o armazenamento, encaminhamento e entrega da informação.

No modelo de interacção baseado em tópicos, os editores ou subscritores interessados em produzir ou consumir informação sobre um determinado assunto têm basicamente que tornar-se membros de um determinado grupo e comunicar num canal de subscrição específico – a maioria dos sistemas comerciais é implementado desta forma. Modelos alternativos são baseados no conteúdo, onde a subscrição indica uma condição para a recepção do evento e os baseados em tipos, onde são subscritos tipos de eventos [1, 5, 6, 7, 12].

A qualidade de serviço oferecida pode incluir várias dimensões: persistência, prioritização, transacções e fiabilidade. A persistência da informação está habitualmente presente em sistemas centralizados, que guardam as mensagens até que os *subscritores* as consumam. Os sistemas com arquitectura distribuída não apresentam necessariamente esta persistência, dado que as mensagens são muitas vezes enviadas directamente do *editor* para todos os *subscritores*. Nestes sistemas não é garantida a entrega de mensagens a processos não correctos. Algums sistemas oferecem a possibilidade de definir prioridades para as mensagens enviadas e outros a capacidade de agrupar múltipas operações em blocos atómicos onde todas as operações são realizadas ou nenhuma o é. A fiabilidade é uma importante característica dos sistemas distribuídos, sendo nos sistemas de *edição/subscrição* importante implementar a propagação fiável dos eventos e promover a garantia de entrega [6].

O artigo está organizado da seguinte forma: a próxima secção introduz a *framework Appia*, que será utilizada no desenvolvimento do sistema. As secções 3 a 5 apresentam a arquitectura, funcionamento, modelos protocolares e demais informações sobre o sistema a implementar. Nas

últimas duas secções fazemos consideramos alguns o trabalho futuro para extender o sistema e apresentamos algumas conclusões.

2. A *framework Appia*

No desevolvimento do sistema FTPS será utilizado o *Appia* [3, 8, 9, 10, 11]. O *Appia* é uma *framework* de composição e execução de protocolos implementada na linguagem de programação Java, que oferece comunicação em grupo.

O modelo de composição do *Appia* é baseado em três abstracções principais: *camadas*, *sessões* e *eventos*. As camadas *(layers)* são responsáveis por descrever o comportamento de um protocolo. As sessões *(sessions)* são instâncias dos protocolos que mantêm o estado necessário para a execução do protocolo. Os eventos *(events)* são estruturas de dados usadas pelas sessões para trocar informação [3].

Os canais *(channels)* são pilhas de sessões de protocolos, e são definidos com uma qualidade de serviço *(QoS)* específica. Os eventos circulam num canal, transportando informação.

Apesar de ter sido desenhado para suportar vários paradigmas de comunicação, o *Appia* é oferecido com suporte à comunicação em grupo através de vários protocolos. Estes protocolos foram desenhados com os mecanismos necessários para criar *sincronia virtual*. A *sincronia virtual* é um paradigma de comunicação em grupo que assenta no facto de todos os participantes num grupo receberem informação sobre a filiação no grupo sob a forma de *vistas* e que todos recebem as alterações às vistas pela mesma ordem [11].

3. O Sistema FTPS

O sistema a desenvolver implementa uma arquitectura de *edição/subscrição* distribuída, baseada em tópicos, com qualidade de serviço *(QoS)* e com características de tolerância a faltas.

O sistema FTPS utiliza um grupo de servidores replicados que mantém informação sobre os tópicos, os grupos e as qualidades de serviço por estes utilizadas. Basicamente são mantidas duas listas, uma lista T que associa cada tópico ao grupo correspondente e uma lista G, que associa a cada grupo uma qualidade de serviço:

$$T = \bigcup_{tópico} \langle tópico, grupo \rangle$$

$$G = \bigcup_{grupo} \langle grupo, QoS \rangle$$

A quantidade de tópicos disponível é ilimitada. Cada tópico tem um único grupo associado, no entanto, os grupos existem em número limitado, podendo haver necessidade de associar vários tópicos a um mesmo grupo. Cada grupo tem uma qualidade de serviço associada.

O grupo de servidores elege um coordenador de forma muito simples – o primeiro processo da vista, que será o responsável por responder aos pedidos dos clientes, executar quaisquer operações que sejam necessárias e actualizar as réplicas.

Para manter as réplicas actualizadas é utilizada replicação passiva esporádica – o coordenador actualiza as réplicas sempre que fôr criado um novo tópico e só nesses casos, isto é, sempre que as listas T ou G sejam alteradas. Sendo previsível que a criação de tópicos tenha um pico no início da operação e depois decaia de tal forma que a criação de tópicos se torne numa operação relativamente rara, de forma a minimizar as perdas em caso de *crash* do coordenador e evitando a verificação periódica das listas, acredita-se que este modo de replicação seja vantajoso.

Apesar de existir no sistema, um grupo de servidores, encarregue manter as listas de tópicos e de grupos, estes servidores não realizam qualquer função de encaminhamento de mensagens, pelo que se pode dizer que não actuam como um *broker* tradicional.

Os clientes do sistema são os *editores* e os *subscritores* que, em relação a cada tópico, anunciam as suas pretensões ao sistema, são informados do identificador de grupo e da qualidade de serviço a utilizar e posteriormente se juntam ao grupo indicado, não ocorrendo da parte do cliente, em relação a esse tópico e acção mais interacções com o sistema.

Os *editores*, assim que se juntam ao grupo onde se enquada o tópico sobre o qual pretendem publicar, podem iniciar imediatamente a publicação de informação.

Os *subscritores* começam a receber as notificações dos eventos do tópico pretendido a partir do momento que se juntam ao grupo. Eventualmente, no caso de haver necessidade de veicular mais do que um tópico num determinado grupo, o cliente poderá que que descartar as mensagens recebidas que sejam relativas a tópicos que não subscreveu.

O sistema não integra qualquer solução de armazenamento persistente de mensagens, o que afecta as características de separação de tempo e separação de sincronização, não existindo uma tão grande liberdade de acção dos clientes, por exemplo, os *subscritores* só são notificados dos novos eventos enquanto forem membros do grupo correspondente, nada recebendo sobre os eventos gerados antes da sua entrada no grupo ou após a sua saída.

4. Arquitectura e funcionamento do sistema

O sistema a desenvolver irá tirar partido das capacidades do *Appia* utilizando como base comunicação sobre TCP.

Assume-se a existência de um *coordenador* no grupo de servidores, que responderá aos pedidos feitos ao sistema.

O sistema está limitado a criar um número de grupos máximo – N, que no mínimo será N terá que ser superior à quantidade de qualidades de serviço disponíveis. Estão actualmente disponíveis duas qualidades de serviço, a saber: *Sincronia na Vista (SV)* e *Ordem Total (SV+OT)*. Cada qualidade de serviço é guardada sob a forma de uma descrição XML da pilha protocolar do *Appia* que concretiza essa mesma qualidade de serviço, assim, será possível no futuro fazer facilmente a extensão do sistema a outras qualidades de serviço.

Inicialmente os clientes procederão à descoberta do coordenador e emitirão os seus pedidos de *anunciar()* ou *subscrver()*, conforme se tratem de *editores* ou *subscritores*.

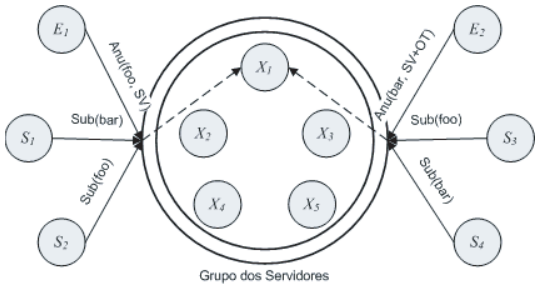

Figura 1. Pedidos de anúncio e subscrição dos clientes.

Os *editores* anunciam ao sistema a intenção de publicarem informações sobre um determinado tópico e com uma certa qualidade de serviço (ver figura 1).

$$anunciar\langle id, tópico, QoS\rangle$$

O *coordenador* verificará a prévia existência do tópico e se a qualidade de serviço coincide com o esperado, no caso de já existir. Se ainda não existir, insere o par (grupo, QoS) na lista de grupos G e o par (tópico, grupo) na lista de tópicos T. Comunica depois ao cliente a identificação do grupo associado ao tópico e caso este seja *subscritor* indica-lhe também a qualidade de serviço requerida.

Os *subscritores* indicam ao sistema que pretendem subscrever um dado tópico.

$$subscrever\langle id, tópico\rangle$$

O *coordenador* responde aos *subscritores* indicando a identificação do grupo correspondente ao tópico e a QoS associada.

Após a recepção da informação sobre a identificação do grupo correspondente ao tópico o cliente junta-se a esse grupo.

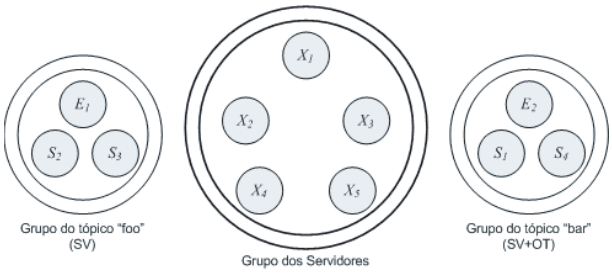

Figura 2. Estado do sistema após junção dos clientes aos grupos correspondentes aos seus pedidos.

A figura 2 mostra os grupos de comunicação constituídos para a disseminação de informação sobre os tópicos *"foo"* e *"bar"*, a par do grupo dos servidores.

O funcionamento restante do sistema tem lugar dentro dos grupos de comunicação com os *editores* a publicarem informação sobre o tópico.

$$publicar\langle id, tópico, informação\rangle$$

Note-se que é necessário o *editor* indicar sempre o tópico de cada mensagem, pois o grupo de comunicação pode estar associado a vários tópicos diferentes, tendo os *subscritores* que filtrar as mensagens recebidas e descartar as que não interessam antes destas chegarem ao nível da aplicação.

5. Pilhas Protocolares

Apresentamos de seguida as pilhas protocolares a utilizar por cada um dos tipos de intervenientes no sistema.

5.1. Servidores

O modelo do servidor visa garantir fundamentalmente a uniformidade do grupo de servidores: coordenador e suas réplicas. A sua pilha protocolar apresenta assim as camadas *UniformLayer* e *LoopBackLayer* que pretendem exactamente garantir a uniformidade do conjunto. Com o objectivo de possibilitar aos clientes a obtenção dos elementos do grupo de servidores, foi necessário substituir a camada *GossipOutLayer* do modelo de comunicação em grupo pela camada apresentada *RemoteGossipOutLayer* por forma a possibilitar a comunicação entre os clientes e o servidor.

Cabe à camada *FTPSServerLayer* a gestão dos tópicos e sua atribuição aos grupos disponíveis, a manutenção das

listas de tópicos e grupos e o desencadeamento dos eventos necessários à replicação das listas por todos os servidores. É também esta camada que selecciona a identificação do grupo correspondente ao tópico pretendido pelo cliente e se encarrega de a comunicar. Cabe aos clientes juntarem-se ao grupo especificado, que é automaticamente criado se ainda não existir.

Todas as camadas presentes na pilha, à excepção da camada *FTPSServerLayer* são fornecidas pelo *Appia*.

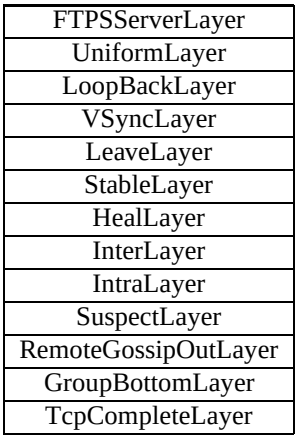

Figura 3. Pilha protocolar dos servidores

5.2. Clientes

O modelo protocolar da figura 4 refere-se à fase em que o cliente declara a sua intenção de publicar ou subscrever informações sobre um dado tópico. Esta é a fase da descoberta. Os clientes usando a camada *RemoteGossipOutLayer* descobrem quais os endereços dos servidores. A camada *FTPSConnectLayer* tem por função garantir que a comunicação é bem sucedida, o que significa, garantir que não só se consegue um grupo de servidores, como também se consegue um grupo para publicação ou subscrição. Esta camada garante à camada *FTPSClientLayer* um nome de grupo. Deixando transparente para esta última todo o processo inerente a falhas de comunicação. Estas falhas, consistem principalmente na não obtenção de um grupo por falha do servidor. Assim sendo, a camada *FTPSConnectLayer* volta a pedir o conjunto de servidores pertencentes ao grupo de servidores. Continuando sucessivamente até não haver mais servidores ou obter um nome de grupo. A camada *FTPSClientLayer* tem como principal função criar mediante a qualidade de serviço a nova aplicação.

Durante a fase normal de operação, onde os clientes *editores* publicam informação e os *subscritores* lêem essa informação, é utilizada a pilha protocolar da figura 5.

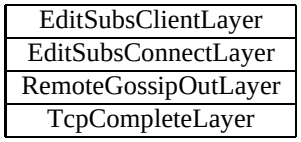

Figura 4. Pilha Protocolar dos Clientes - fase de descoberta

Esta pilha só utiliza a cadama *TotalSequencerLayer* para grupos onde esta capacidade é requerida. A camada *FTPSClientLayer* fornece o interface com os clientes e faz a adesão do cliente ao grupo de comunicação pretendido. A camada *TopicFilterLayer* só existe para os *subscritores* e tem como função filtrar os assuntos não subscritos que também circulam no grupo.

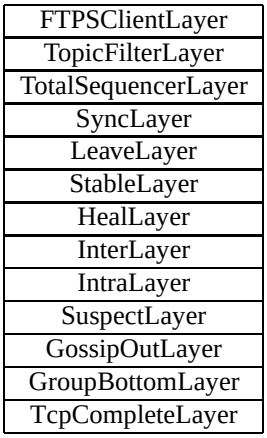

Figura 5. Pilha protocolar dos clientes - fase de operação

6. Conclusões

A criação de um sistema de *edição/subscrição* recorrendo a uma arquitectura distribuída com vários servidores replicados e grupos de comunicação distintos para os vários tópicos permite minimizar em grande parte os problemas inerentes a uma solução centralizada. Isto poderá ter, no entanto, o custo de uma maior complexidade, principalmente no caso de se pretender oferecer um sistema completo, sendo necessário recorrer a repositórios distribuídos ou outros mecanismos.

7. Trabalho Futuro

A solução apresentada contempla um grupo de servidores onde é eleito um coordenador sendo os outros servidores meras réplicas, prontas a substituí-lo. Num ambiente onde haja muitos clientes poderá ser necessário utilizar todos os servidores concorrentemente, com balanceamento de carga, para satisfazer os pedidos de forma mais eficiente. Nesse caso, os pedidos de leitura da lista de tópicos não levantariam problema de maior, sendo atendidos individualmente por cada servidor, no entanto, a criação de novos tópicos ou grupos de comunicação teria que ser tratada com algum cuidado.

A introdução de persistência da informação na solução seria também um ponto a considerar, criando um repositório distribuído, no entanto, seria necessário subverter o esquema de sincronia virtual proposto pelo *Appia*, ou procurar outra solução alternativa.

Referências

[1] F. Araújo and L. Rodrigues. On qos-aware publish-subscribe. In *ICDCS Workshops*, pages 511–515, 2002.

[2] N. Carvalho, F. Araújo, and L. Rodrigues. Scalable qos-based event routing in publish-subscribe systems. In *NCA*, pages 101–108, 2005.

[3] N. Carvalho and L. Rodrigues. Implementing reliable broadcast protocols in appia. Technical report, Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, Lisboa, Nov. 2003.

[4] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, 20(8):1489–1499, 2002.

[5] A. K. Datta, M. Gradinariu, M. Raynal, and G. Simon. Anonymous publish/subscribe in p2p networks. In *IPDPS*, page 74, 2003.

[6] P. T. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.

[7] R. Guerraoui and L. Rodrigues. *Introduction to Reliable Distributed Programming*. Springer-Verlag, Berlin, Germany, 2006.

[8] H. Miranda, A. Pinto, N. Carvalho, and L. Rodrigues. Appia protocol development manual. Technical report, Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, Lisboa, Dec. 2005.

[9] H. Miranda, A. Pinto, and L. Rodrigues. Appia: A flexible protocol kernel supporting multiple coordinated channels. volume 00, page 0707, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[10] J. Mocito, L. Rosa, N. Almeida, and L. Rodrigues. Appiaxml. Technical report, Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, Lisboa, Sept. 2004.

[11] A. Pinto. Appia group communication. Technical report, Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, Lisboa, 2005.

[12] R. Szarowski. Hybrid publish-subscribe: A compromise approach for large-scale. In *CAiSE Short Paper Proceedings*, 2003.

[13] P. Verissimo and L. Rodrigues. *Distributed Systems for System Architects*. Kluwer Academic Publishers, Norwell, MA, USA, 2001.