1
Introdução

O trabalho descreve o sistema LuaPS, cujo objetivo é investigar a construção de sistemas publish-subscribe utilizando uma arquitetura dividida em camadas, focada em generalidade, flexibilidade e extensibilidade. O sistema, desenvolvido em Lua, utiliza uma tabela hash distribuída como base e é executado no ambiente orientado a eventos disponibilizado pela ALua.

O capítulo introdutório busca definir os objetivos do estudo. Para possibilitar um claro entendimento do que é proposto, apresentamos o paradigma publish-subscribe e a sua relação com as tabelas hash distribuídas nas primeiras seções do capítulo. Em seguida, definimos os objetivos e a organização do estudo.

PUC-Rio - Certificação Digital Nº 0410850/CA

1.1
Publish-Subscribe

Publish-subscribe é um conhecido paradigma para o desenvolvimento de sistemas distribuídos, caracterizado pelo desacoplamento das partes e pela comunicação assíncrona. Três componentes participam do paradigma, com *produtores* e *consumidores* de informações se comunicando com o intermédio de *serviços de notificação*. Os produtores das informações as tornam disponíveis publicando notificações de forma assíncrona. Os consumidores especificam o interesse em receber certas classes de mensagens através de inscrições, definindo sua política. O serviço de notificação tem a função de desassociar o consumidor do produtor. Para isto, o serviço de notificação encaminha as notificações recebidas do produtor para todos os consumidores que tenham se inscrito para receber as dadas informações.

Sistemas publish-subscribe podem ser classificados de acordo com o tipo de filtro utilizado sobre as notificações. A maioria dos novos sistemas publish-subscribe desenvolvidos é baseada em conteúdo (Mühl et al., 2004), embora também existam sistemas baseados, por exemplo, em tópicos, hierarquias ou canais. O filtro por conteúdo é o tipo de filtro considerado mais expressivo, possibilitando que todo o conteúdo da notificação seja comparado com o filtro.

A utilização do paradigma publish-subscribe nos remete intuitivamente a sistemas de informações onde os clientes se inscrevem para receber publicações referentes a notícias. Podemos citar diversos sistemas desenvolvidos com este propósito, como o Kenamea (www.kenamea.com) ou o Radio UserLand (radio.userland.com). Embora este tipo de sistema seja bastante comum, o paradigma publish-subscribe pode ser utilizado no desenvolvimento de outras classes de aplicações. O Mercury (Bharambe et al., 2002), por exemplo, utiliza o paradigma para o desenvolvimento de jogos para a internet.

O paradigma publish-subscribe é bastante adequado ao desenvolvimento de aplicações móveis. Além da possível limitação de recursos dos dispositivos móveis, quedas momentâneas de conexão são bastante freqüentes, conseqüência das variações rápidas nas condições do ambiente. A comunicação desacoplada e assíncrona do paradigma garante que desconexões e limitações de recursos de um cliente não afetem o sistema de forma global.

A utilização exclusiva das funcionalidades básicas disponibilizadas pelo paradigma, isto é, a criação e a inscrição em tópicos e a publicação de notificações, pode restringir o desenvolvimento de sistemas robustos. Desta forma, a maior parte dos sistemas publish-subscribe incorpora extensões, considerando as necessidades específicas das aplicações. Uma vez que as alterações propostas estão muito mais ligadas às aplicações desenvolvidas do que ao paradigma propriamente dito, podemos considerar estas extensões como serviços implementados pelo sistema.

PUC-Rio - Certificação Digital Nº 0410850/CA

1.2 Objetivos e Organização

O trabalho realizado, baseado no desenvolvimento do sistema LuaPS, define uma arquitetura dividida em camadas focada em generalidade, flexibilidade e extensibilidade para o desenvolvimento de aplicações publish-subscribe. O sistema, desenvolvido em Lua, utiliza uma tabela hash distribuída como base e é executado no ambiente orientado a eventos disponibilizado pela ALua.

Sistemas publish-subscribe complexos são definidos por diversas decisões de projeto que levam a implementações totalmente diferentes. Podemos diferenciar os sistemas existentes levando em conta, por exemplo, o ambiente de execução, a forma de comunicação ou os serviços disponibilizados. Na maioria dos sistemas existentes, entretanto, as diferentes decisões de projeto não são modularizadas, sendo o sistema final definido pelo conjunto de decisões tomadas. O sistema LuaPS busca se diferenciar destes sistemas

monolíticos, definindo diferentes camadas e módulos que agrupam decisões de implementação relacionadas.

Embora o desenvolvimento em camadas não seja novidade em sistemas de computação, sua aplicação a sistemas publish-subscribe nos permite um melhor entendimento das características inerentes a tais sistemas e das possibilidades de extensão.

A separação em camadas nos permite individualizar cada uma das decisões tomadas e, desta forma, entender melhor a solução desenvolvida e perceber as alterações que seriam necessárias para alterar a estrutura do sistema. Se considerarmos que cada uma das camadas é especializada em uma parte da solução completa, podemos perceber que o desenvolvimento estará mais focado e as soluções implementadas poderão ser mais bem avaliadas. Cada uma das camadas poderá ser desenvolvida e testada separadamente, garantindo que os problemas sejam mais facilmente detectados e corrigidos. No que se refere ao desenvolvimento, entretanto, a principal contribuição da arquitetura em camadas é a evolução e alteração do sistema. Uma vez que a interface continue sendo respeitada, melhorias nas funcionalidades podem ser feitas sem que as outras camadas do sistema sejam afetadas.

Inicialmente, o trabalho apresenta os sistemas utilizados e as alterações realizadas nos mesmos, corrigindo problemas existentes ou desenvolvendo funcionalidades necessárias aos objetivos do projeto.

O terceiro capítulo apresenta a arquitetura utilizada pelo sistema, analisando a interação entre diversos nós da rede e a estrutura de um nó individualmente. Ao analisarmos a arquitetura considerando a de rede de nós, apresentamos diferentes arquiteturas e justificamos a escolha da arquitetura utilizada. Ao analisarmos a arquitetura considerando um nó individualmente, discutimos a divisão em camadas, justificamos sua utilização e apresentamos as camadas do sistema.

O capítulo seguinte discute com mais especificidade a camada publish-subscribe do sistema. Apresentamos uma análise sobre gerência dos tópicos, entradas e saídas da rede e execução e buferização de comandos. Definimos a função de publicação e sua natureza genérica, aumentando a flexibilidade do sistema. Ilustramos, ainda, a utilização da função de publicação para executar filtros por conteúdo.

O quinto capítulo apresenta a camada de serviços, cujo principal objetivo é estender a camada publish-subscribe com funcionalidades que não caracterizam o paradigma de comunicação, mas que são necessárias para aplicações publish-subscribe complexas. Analisamos a utilização de serviços para permitir o acesso a notificações antigas, para tolerância a falhas e para o desenvolvi-

PUC-Rio - Certificação Digital Nº 0410850/CA

mento de um gateway publish-subscribe, particularmente interessante ao considerarmos a utilização do sistema por clientes móveis. Apesar da relevância dos serviços desenvolvidos, o principal objetivo da seção é ilustrar a extensibilidade do sistema.

Em seguida, apresentamos a camada de aplicação. Uma vez que o desenvolvimento de uma aplicação publish-subscribe não é o principal objetivo do estudo, este capítulo busca apenas ilustrar o processo de desenvolvimento que deve ser seguido na definição de aplicações que utilizem o sistema. O principal objetivo dos módulos desenvolvidos, portanto, será o teste das funcionalidades implementadas.

O último capítulo conclui o estudo e apresenta os pontos de destaque e as principais contribuições do projeto. Neste capítulo, apresentamos também alguns trabalhos relacionados e os comparamos com o sistema LuaPS.

PUC-Rio - Certificação Digital Nº 0410850/CA