6
Camada de Aplicação

O desenvolvimento de uma camada de aplicação robusta não é um dos pontos mais importantes do projeto. O desenvolvimento de uma camada simples, entretanto, se faz necessário para exemplificar o processo, ilustrar as situações descritas e possibilitar a realização de testes completos. Por uma questão de simplicidade e clareza, optamos por definir mais de um módulo para a camada de aplicação, possibilitando que cada um dos módulos enfoque objetivos específicos.

A primeira seção deste capítulo define as metodologias de teste utilizadas. A segunda seção descreve os testes das funcionalidades, onde iremos testar a camada publish-subscribe, o módulo de console e os módulos de serviço. A última seção descreve os testes de carga.

6.1
Metodologias de Teste

Os módulos desenvolvidos para a camada de aplicação seguem basicamente duas metodologias. O módulo que possibilita o teste do console para interface com o usuário disponibiliza uma interface com a qual o usuário pode solicitar a execução das funções publish-subscribe. Este módulo define a primeira metodologia utilizada, onde não existem testes automatizados e a qualidade dos testes dependerá do usuário. Nos outros módulos de teste desenvolvidos, os testes são automatizados e não existe interação com o usuário. Os módulos possibilitam o teste das funções publish-subscribe e das funções dos módulos de serviço.

Para implementar testes automatizados, utilizamos o LuaTimer para definir um script de execução. Ao iniciar a aplicação, um processo ALua, que será denominado processo master, é criado. Este processo inicia uma aplicação ALua e os outros processos que irão participar do teste. Em seguida, o processo define as funções que serão executadas e, utilizando os temporizadores, define a

seqüência de execução. Cada vez que um temporizador é executado, o processo master envia uma mensagem para um dos processos da aplicação, definindo um comando que deverá ser executado. Desta forma, este processo comanda a execução de todos os comandos nos processos participantes da aplicação.

6.2 Teste das Funcionalidades

O primeiro módulo de aplicação desenvolvido, o módulo *testeConsole*, busca testar e demonstrar a utilização do console desenvolvido integrado com a camada de aplicação do projeto LuaPS. A aplicação, embora muito simples, nos permite perceber a flexibilidade de utilização do console desenvolvido. A aplicação disponibiliza uma interface de comandos para o usuário e, após a digitação do comando, executa o comando publish-subscribe requisitado. Para que o usuário conheça os comandos disponibilizados, a aplicação imprime os comandos e as respectivas sintaxes na tela. Um comando do usuário que não esteja presente na lista é ignorado.

O segundo módulo de aplicação desenvolvido, o módulo *testeHistory*, tem o objetivo de testar as funcionalidades disponibilizadas pela camada publish-subscribe e pelo módulo de acesso a notificações antigas. O módulo testa as funções utilizadas para criação de tópicos, inclusão e exclusão de inscrições com diferentes funções de publicação, publicação de notificações e solicitação de histórias. O módulo testa, ainda, a associação de novos campos à estrutura de gerência de tópicos e a manutenção da consistência das informações após a entrada ou saída de nós da rede.

O objetivo do terceiro módulo de aplicação desenvolvido, o módulo *testeGateway*, é testar a utilização do módulo de gateway para clientes móveis. O módulo testa a execução das funções da camada publish-subscribe utilizando o gateway e verifica a persistência das conexões.

Assim que a aplicação é iniciada, o processo master cria processos que, somados a ele, formam a rede publish-subscribe. Além dos processos da rede LuaPS, o processo master instancia também processos que representam clientes móveis. Estes processos não estão conectados diretamente na rede publish-subscribe, podendo interagir com ela apenas através do gateway.

Uma vez que os clientes móveis não estão na rede publish-subscribe, o único módulo que precisa ser carregado por estes processos é a camada de aplicação, definida no arquivo *testeGatewayApp* e que se comunica com o gateway. Os clientes fixos criados não executam nenhuma aplicação, tendo apenas

a função de manter a infra-estrutura publish-subscribe. O processo master representa o gateway publish-subscribe. Desta forma, o processo também não tem camada de aplicação, mas deve importar o módulo de gateway, provendo as funcionalidades esperadas pelos clientes móveis.

O módulo de aplicação carregado pelos clientes móveis é bastante simples, contendo uma função para configurar o gateway e funções publish-subscribe que direcionam as requisições para o gateway. Esta é uma das vantagens de se utilizar o gateway, permitindo que clientes móveis, com recursos limitados, executem um código mais simples.

O último módulo da camada de aplicação que iremos descrever, o módulo *testeBackup*, foi implementado com o objetivo de possibilitar o teste do módulo de tolerância a falhas da camada de serviços. O módulo testa o backup e a restauração das informações após a falha de um nó.

No capítulo que apresentou a camada pastry, havíamos destacado uma falha no sistema LuaPastry que impedia a re-estruturação da rede pastry. Desta forma, o teste de restauração do backup simula a falha de um cliente para testar o módulo de serviço. A falha de um cliente é identificada pela chegada de uma requisição a um cliente que se refira a um tópico que está presente na tabela de backup. Uma mensagem só seria roteada para este nó no caso de falha do nó que originalmente gerenciava o tópico. Para simular uma falha, enviamos diretamente para um nó uma requisição referente a um tópico na tabela de backup. A chegada desta requisição inicia o processo de restauração do backup, que é executado como se o nó em falha não mais existisse na rede. Neste teste, entretanto, as duas entradas da tabela de folhas estão corretas desde o início do processo. Apesar disto, o processo acontece normalmente e roteia as mensagens que ajustam estas entradas.

6.3 Teste de Carga

Os testes de carga analisam o desempenho do sistema, buscando dar uma noção do seu nível de escalabilidade. Desta forma, a escolha pela utilização do sistema pode ser feita tendo em mente o desempenho previsto. Os testes são realizados considerando casos extremos. Podemos admitir, portanto, que em situações reais de uso os resultados não serão piores do que os obtidos nos testes.

Ao projetarmos os testes de carga visamos analisar dois pontos críticos do sistema. O primeiro ponto está relacionado com a estrutura utilizada para

gerência dos tópicos, descrita no capítulo que apresenta a camada publish-subscribe. Precisamos verificar se as operações publish-subscribe realizadas sobre esta estrutura apresentam tempo de execução aceitável. O segundo ponto se refere ao número de mensagens trafegado na rede. Em função da arquitetura utilizada no sistema, as operações poderão originar um grande número de mensagens sendo trafegadas na rede. Precisamos analisar esta questão e verificar se o sistema se comporta de forma aceitável.

A criação de um tópico ou a inclusão de uma inscrição são operações com tempo constante. A remoção de uma inscrição ou a publicação de uma notificação pode, no pior caso, exigir que todos os elementos de tabela de inscrições de um tópico sejam percorridos. Sendo assim, a execução de uma operação tem ordem O(N) no pior caso, onde N é número de inscrições em um tópico. Para analisarmos o tempo das operações sobre a estrutura, portanto, podemos considerar apenas o pior caso da operação de publicação e admitir que o sistema terá melhor desempenho nas outras situações.

Para testar a carga de mensagens trafegadas na rede utilizamos o mesmo raciocínio. Além das mensagens de comando, a operação de publicação pode, no pior caso, originar uma mensagem para cada um dos clientes inscritos. Utilizamos este caso para analisar a carga de mensagens na rede.

Para realizar os testes de carga utilizamos os testes automatizados definidos no módulo *testeCarga*. O módulo cria uma rede publish-subscribe, cria um tópico, realiza as inscrições e a publicação das notificações. Utilizamos duas funções de publicação diferentes nas inscrições. A primeira função de publicação nos permite determinar o tempo que o gerente do tópico demora para executar a publicação das notificações, enviando uma mensagem para cada um dos clientes inscritos. Neste teste, não consideramos o tempo que as mensagens levam para chegar ao destino. A segunda função de publicação nos permite determinar o tempo de total, isto é, o tempo da operação somado ao tempo da transmissão e execução da mensagem.

Uma vez que estamos utilizando o tempo como medida de adequação, precisamos definir o ambiente em que os testes foram executados. Já que não queremos estabelecer uma configuração mínima para a execução do sistema, utilizamos um PC486, uma máquina com baixo poder de processamento, para a realização dos testes. Como estamos sempre optando pelo pior caso, podemos admitir que os resultados seriam melhores em situações normais de uso do sistema.

O número de inscrições variou de 500 a 2000 nos testes realizados. Para a situação e o ambiente de execução descritos, não conseguimos chegar a 2500 inscrições sem que houvesse perda de notificações, conseqüência da

impossibilidade de processamento das mensagens antes da sobrecarga do buffer de recebimento. Os valores atingidos, entretanto, são bastante razoáveis, principalmente se considerarmos as condições adversas propostas pelos testes.

A figura 6.1 ilustra os resultados encontrados. A seqüência mais escura representa os resultados do teste sobre as operações e a seqüência mais clara representa os resultados do teste de transmissão.

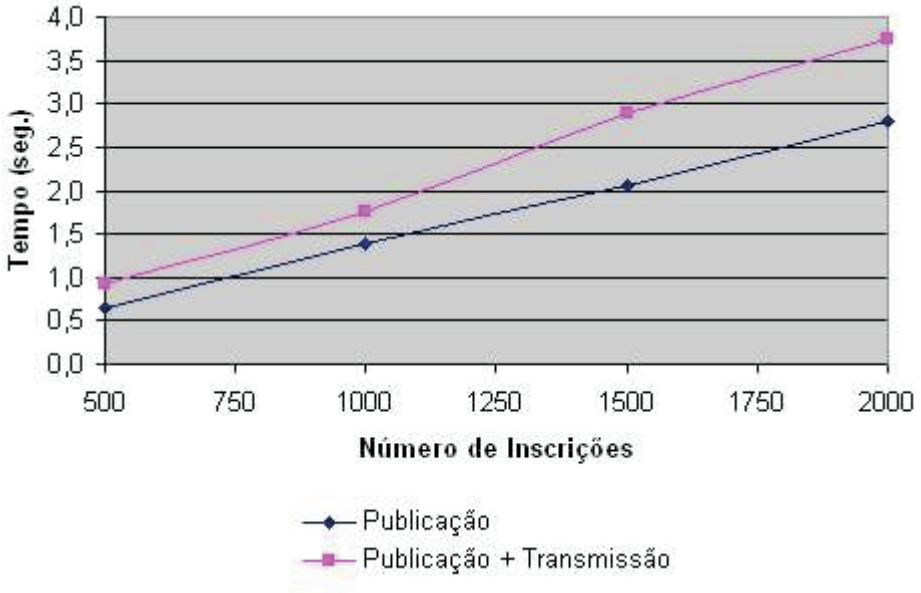

Figura 6.1: Resultados do Teste de Carga