Logger - Documentação

José Oswaldo Cavalcante da Silva Filho - jocmail@gmail.com

Quarta 22 de Setembro de 2010

UNIVERSIDADE FEDERAL DE ALAGOAS
Instituto de Computação

Logger - Documentação

Nome: José Oswaldo Cavalcante da Silva Filho

Registro Acadêmico: 2007G0393

Curso e Período: Ciência da Computação, 7º Período

Disciplina: Sistemas Distribuídos

Maceió - AL
2010

Sumário

1 Introdução **4**

1 Objetivo 4

2 Funcionamento Básico 4

3 Requisitos 4

3.1 Entrega 01 4

3.2 Entrega 02 5

3.3 Entrega 03 5

3.4 Entrega 04 5

2 Desenvolvimento **6**

1 Ferramentas-case 6

2 Tecnologias 6

3 Aplicação Servidora **7**

1 Modos de Operação 7

1.1 Monothread 7

1.2 Multithread 8

2 Configurações 8

3 Diagrama de Classes 9

4 Estados do Serviço 10

5 Teste de sobrecarga 11

4 Aplicação Cliente **12**

1 Diagrama de Classes 13

Lista de Figuras

3.1 Tela do arquivo de configuração do servidor. 8
3.2 Diagrama de classes do servidor. 9
3.3 Servidor parado. 10
3.4 Servidor rodando. 10
3.5 Resultados do teste de sobrecarga com JMeter para 5000 requisições. . . . 11

4.1 Screenshot do LoggerClient. 12
4.2 Diagrama de classes do cliente. 13

Capítulo 1

Introdução

O presente projeto, denominado "Logger" - é uma aplicação de arquitetura Cliente-Servidor, desenvolvida para o aprendizado da disciplina de Sistemas Distribuídos 2010.2, da Universidade Federal de Alagoas. O presente documento visa explicar a arquitetura e os diversos padrões utilizados na implementação do mesmo.

1 Objetivo

Logger é geralmente um mecanismo utilizado para registro de operações efetuadas num dado servidor. Um modo de controlar as requisições efetuadas no servidor e as alterações ocorridas. Suas utilidades vão desde partes de segurança até a manutenção e estabilidade de um sistema.

2 Funcionamento Básico

O funcinamento básico do Logger, consiste em ser uma aplicação de rede com arquitetura cliente-servidor, onde um cliente pode conectar-se ao servidor e requisitar um dado serviço. O serviço disponível, a título de aprendizado, consiste em após conectado ao servidor enviar-lhe uma mensagem de log. Em contrapartida o servidor deve, recebendo a mensagem, retorná-la ao cliente (echo), como confirmação da requisição e a efetiva resposta da mesma.

3 Requisitos

3.1 Entrega 01

- Aplicação Servidora com suporte a Monothread e multithread (não deve ter interação com usuário na implementação do serviço)

- Aplicação Servidora com suporte a Monothread e multithread (não deve ter interação com usuário na implementação do serviço)
- Aplicação Cliente
- A aplicação servidora deve ser flexível para alternar entre os tipos (procurar melhorar no design)
- Teste de carga com as seguinte métricas: Throughput (número de requisições atendidas/segundo); Turnaround time (quantidade de tempo para responder uma requisição particular);

3.2 Entrega 02

- Alterar o Serviço para aguardar (x segundos) - a idéia é emular um serviço que consuma razoável tempo - o impacto da mudança deve ser mínimo - (questões de design)
- Uso de ThreadPool
- Rodar novamente teste de carga
- Definição de Requisitos e Protocolo
- Deve-se adicionar um arquivo de configuração do servidor (Properties ou XML)

3.3 Entrega 03

- Implementação da aplicação cliente-servidor do Logger - Primeira versão
- Deve haver testes automatizados (de unidade e de integração)
- Documentação
- Gerar primeiro release
- Obrigatório uso de controlador de versão

3.4 Entrega 04

- Todos os serviços do Logger implementado atentando para todos os requisitos de avaliação

Capítulo 2

Desenvolvimento

1 Ferramentas-case

- Qt Creator 1.3.1 - IDE de desenvolvimento sobre o framework Qt/C++.
- Kile - IDE de escrita latex, para esta documentação.

2 Tecnologias

O projeto foi desenvolvido em linguagem C++ com uso do framework Qt e o módulo QtNetwork, que disponibiliza ferramentas para construção de aplicações de rede. Consta de interface gráfica e configuração do servidor por aquivo, de forma análoga a Properties Java.

Capítulo 3

Aplicação Servidora

1 Modos de Operação

Um dos requisitos do projeto, eram os modos de alternância de operação do servidor. Ele deveria ser flexível para operar em modo Monothread e Multithread. Os quais serão melhor descritos nas sessões seguintes. Foi também requisistado que o a alternância entre os modos não tivessem interação com usuário na implementação do serviço.

Foi utilizado o padrão de projeto Strategy para alternância entre os modos de operação de forma elegante, clara e concisa. A Classe "Server", possui um método abstrato, virtual puro, ou seja, obrigatoriamente deve ser implementado pelas classes filhas, que herdam da Classe "Server" chamado incomingConnection. A Classe server por sua vez herda da Classe QTcpServer, que possui implementações do framework Qt para modo servidor. Sua implementação, consiste em: Assim que uma nova conexão é requisitada, o método incomingConnection(int socketDescriptor) é automaticamente executado, e o socketDescriptor passa a ser o identificador dessa nova conexão.

Para os modos de operação foram criadas as classes: MonothreadServer e MultithreadServer, que como descrito anteriormente herdam da classe Server. Assim a depender do modo esclhido e determinado pela implementação do Strategy, o modo como o método incomingConnection é implementado varia de acordo com o modo de operação do servidor.[1]

1.1 Monothread

No modo monothread, o servidor, tendo sido requisitado para atender uma dada requisiçã, deve resolver uma a uma, deixando as outras esperando, enquanto resolve-se a requisição atual. Este modo pode ser escolhido no arquivo de configuração do servidor.

[1]Para uma melhor compreensão, ver diagrama de Classes.

1.2 Multithread

No modo multithread, a aplicação servidora, deve recolher todos as requisições e enfileirá-las como threads, ficando a cargo do sistema operacional o tratamento das mesmas para execução do servidor a fim de atender a cada requisição de conexão pendente. Para tanto, utilizou-se as facilidades do framework Qt para implentação de ThreadPool, onde o tamanho das filas de requisição também podem ser determinadas no arquivo de configuração.

2 Configurações

É possível configurar o funcionamento do servidor através de um arquivo de configuração (settings/settings.txt).

Figura 3.1: Tela do arquivo de configuração do servidor.

É possível escolher entre os modos de operação: Monothread e Multithread, escolher o tamanho da ThreadPool e a porta de conexão com servidor.

3 Diagrama de Classes

Abaixo, diagrama de classes do Logger Server.

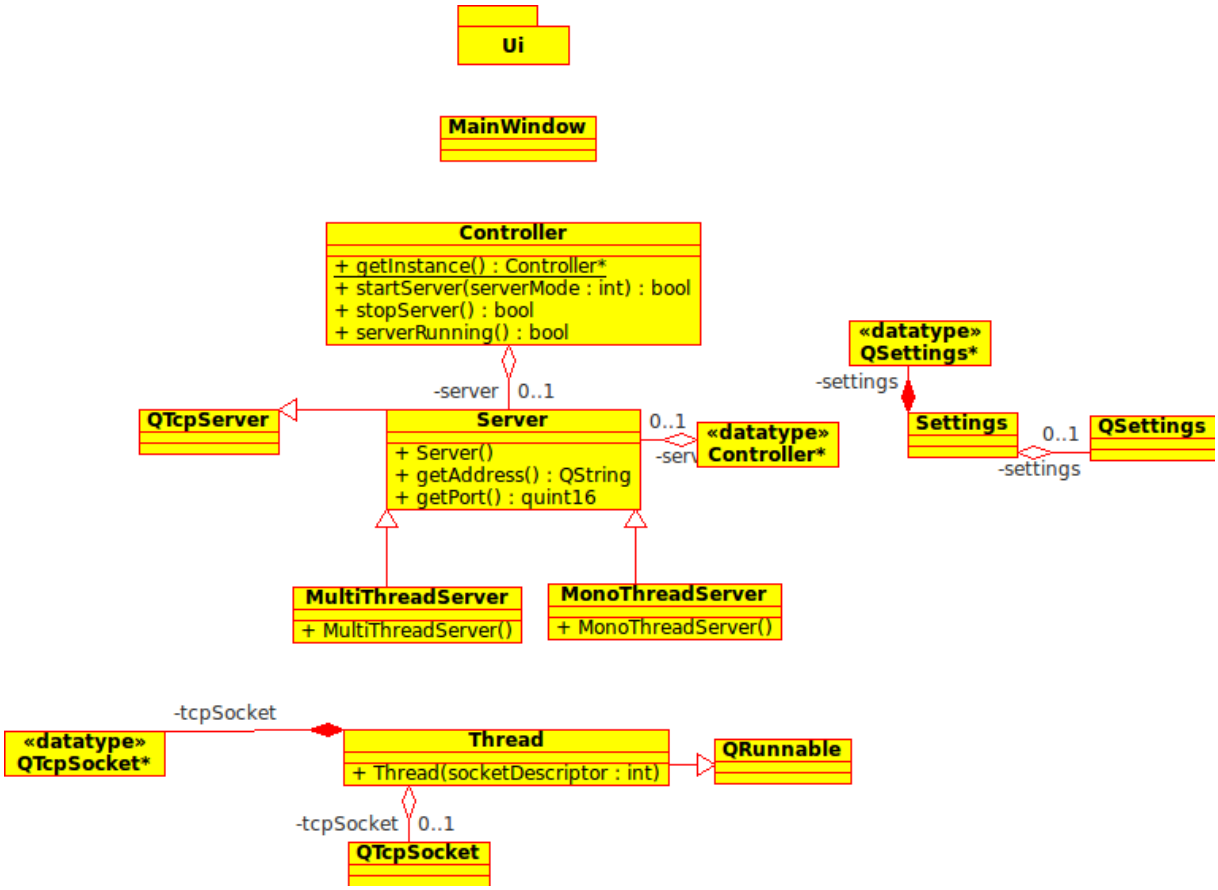

Figura 3.2: Diagrama de classes do servidor.

4 Estados do Serviço

Os serviços podem ser iniciados e parados através da interface gráfica, no entanto, quando há algum problema pelo qual o serviço não pôde ser iniciado, uma mensagem é exibida via console relatando o problema.

Quando os serviços estão disponíveis, a interface exibe as configurações de conexão do servidor: Endereço IP e a porta.

Figura 3.3: Servidor parado.

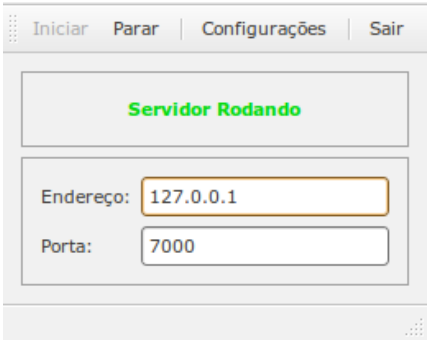

Figura 3.4: Servidor rodando.

5 Teste de sobrecarga

Para os testes de sobrecarga do servidor, utilizou-se a aplicação Java JMeter. Uma aplicação bastante abrangente para testes de sobrecarga em rede. Abaixo os resultados para um teste de sobrecarga com 5000 requisições.

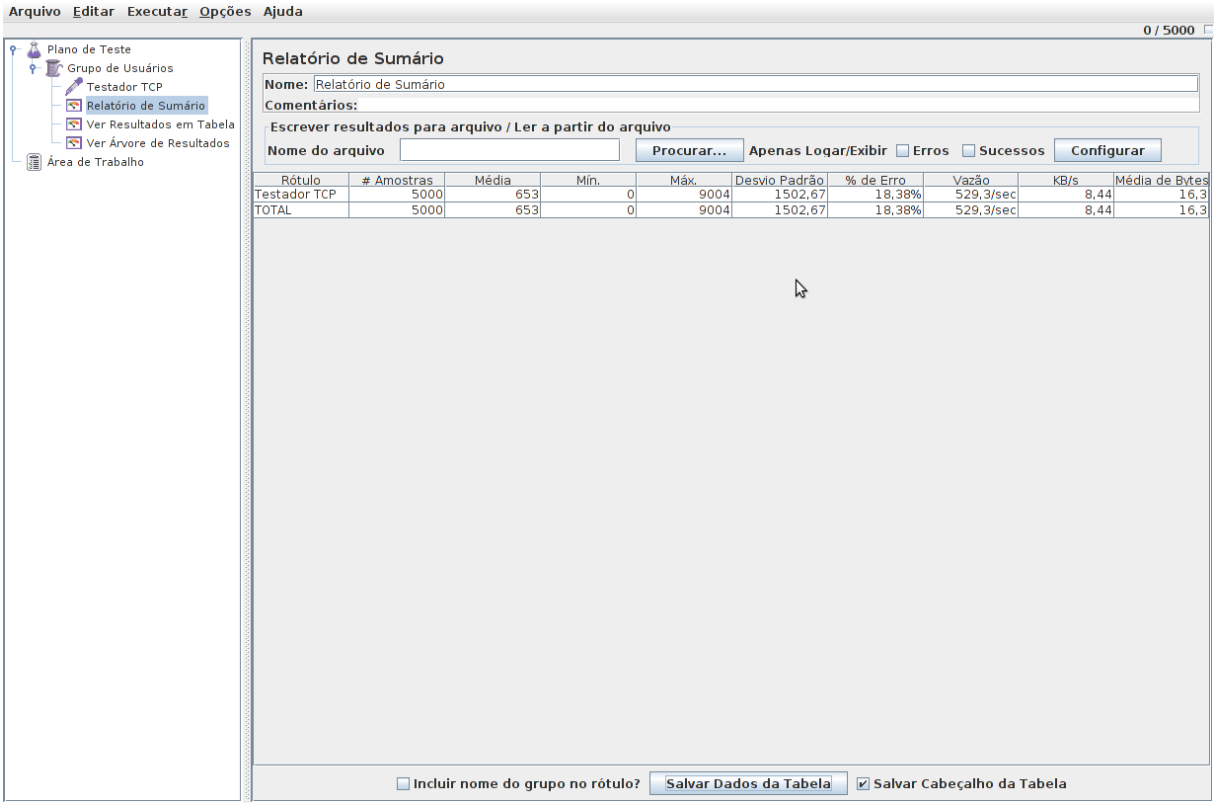

Figura 3.5: Resultados do teste de sobrecarga com JMeter para 5000 requisições.

Capítulo 4

Aplicação Cliente

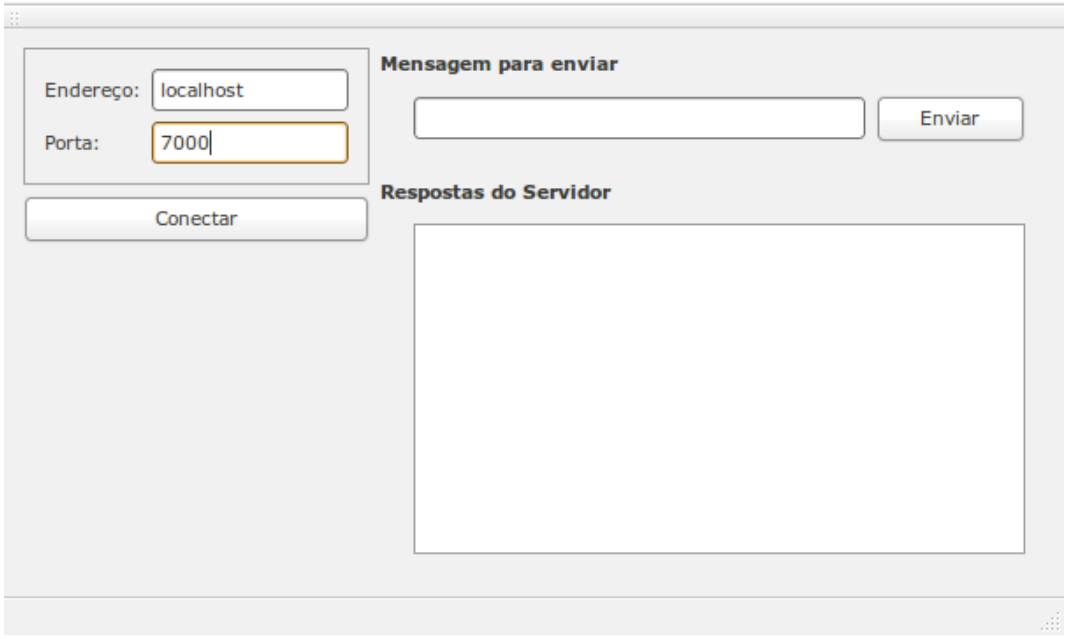

Figura 4.1: Screenshot do LoggerClient.

1 Diagrama de Classes

Abaixo, diagrama de classes do Logger Client.

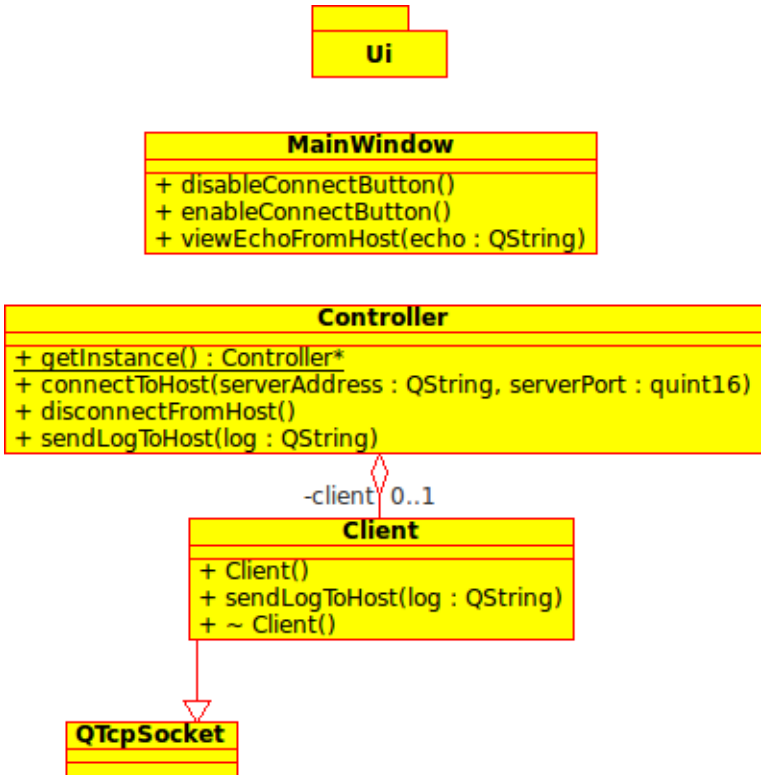

Figura 4.2: Diagrama de classes do cliente.