Simulação de Pool de Impressão Dístribuida Utilizando Socket Exercício Computacional II - Sistemas Distribuídos

Rafael Gonçalves de Oliveira Viana[1]

[1]Sistemas de Informação – Universidade Federal do Mato Grosso do Sul (UFMS)
Caixa Postal 79400-000 – Coxim – MS – Brazil

rafael.viana@aluno.ufms.br
23 de outubro de 2017

Resumo. *Este relatório introduz a arquitetura de uma Pool de Thread.*

1. Introdução

De acordo com [1] em um ambiente de trabalho, onde deseja se imprimir documentos enviados num curto período de tempo, nessecita se utilizar o pool de impressão.

Uma pool de impressão permite que várias impressoras físicas possam ser controladas por uma única impressora lógica, sempre que um trabalho for enviado para a impressora lógica, esta consultará o estado das impressoras físicas para verificar qual equipamento está livre no momento e enviará o trabalho para ela .

2. Fundamentação Téorica

Nesta sessão serão tradatas, as tecnologias que foram utilizadas neste documento, como é o exemplo das bibliotecas Java Sockets e Threads.

2.1. Socket

Segundo [2].[3]. Os sockets são compostos por um conjunto de primitivas do sistema operacional e foram originalmente desenvolvidos para o BSD Unix. Podem ser utilizados nos mais variados sistemas operacionais com recursos de comunicação em rede, sendo suportados pela maioria das linguagens de programação. Sockets são suportados em Java desde o JDK 1.0, para sua utilização devemos fazer uso das classes contidas no pacote java.net. Um exemplo interessante da programação de sockets em Java são os drivers JDBC do tipo 4, que usam sockets para comunicar-se diretamente com a API de rede do banco de dados.

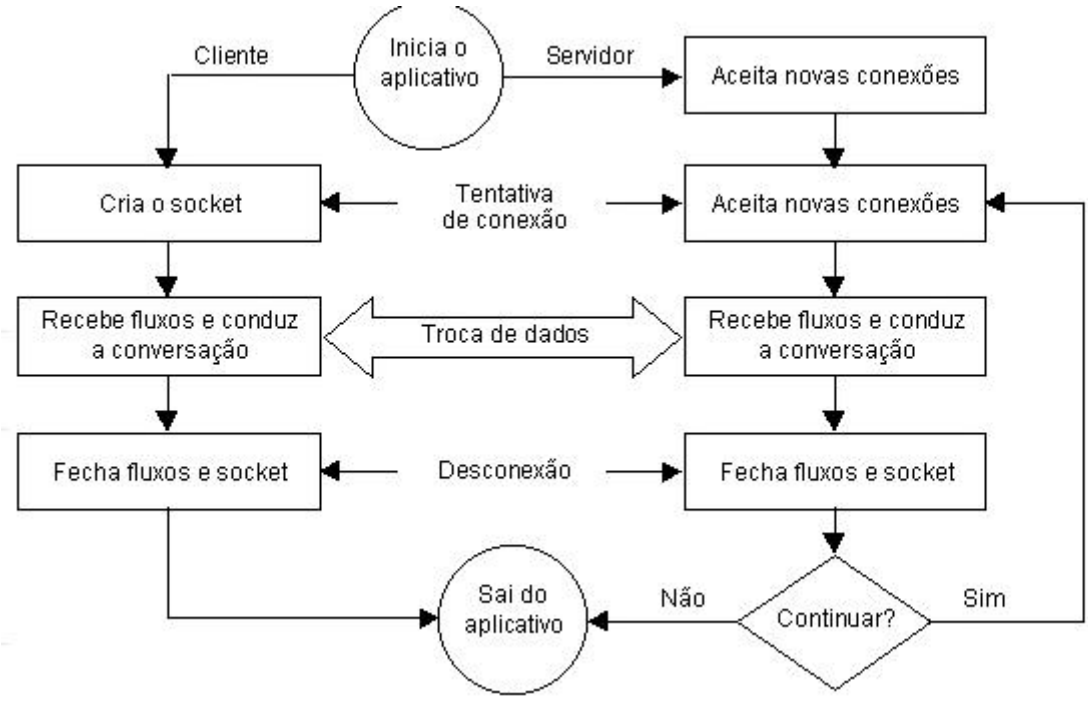

Figura 1. Fluxo de troca de dados com sockets.

```
 /**
 * This is a doc comment.
 */
public void executa() throws IOException {
        Buffer.addImpressora(new
           Impressora("127.0.0.1",2323,"2323"));
        Buffer.addImpressora(new
           Impressora("127.0.0.1",2324,"2324"));
        SaidaDocumento escalonador = new SaidaDocumento();
        new Thread(escalonador).start();
        ServerSocket servidor = new
           ServerSocket(this.porta);
        System.out.println("Porta "+porta+" aberta!");
        while (true) {
                Socket cliente = servidor.accept();
              ChegadaDocumento tc = new
                 ChegadaDocumento(cliente, new
                 Scanner(cliente.getInputStream()).nextLine(),
                new
                   PrintStream(cliente.getOutputStream()));
                new Thread(tc).start();
        }

}
```

2.2. Threads - Problema de Concorrência

Da necessidade de dois computadores se comunicarem, surgiram diversos protocolos que permitissem tal troca de informação: o protocolo que vamos usar aqui é o TCP (Transmission Control Protocol) [3].

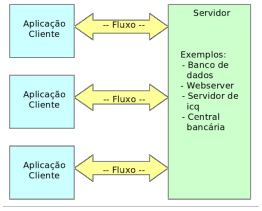

Figura 2. Conexão TCP.

É possível conectar mais de um cliente ao mesmo servidor, como é o caso de diversos banco de dados, servidores Web, etc.

3. Desenvolvimento

Para resolver o trabalho proposto, foi desenvolvido uma aplicação java que simula, uma pool de impressões que as ordena e envia para uma impressora. Foram criadas 3 classes Javas sendo elas Cliente, Pool de Impressão e Impressora as mesmas serão comentandqas nas póximas sessões

3.1. Clientes

Essa classe se conecta a classe servidor com objetivo de repassar informações via Socket, que serão processadas posteriormente notificando o cliente do processamento da impressão.

3.2. Pool de Impressão

Classe responsável pela comunicação dos Sockets (Cliente/Impressora), esse como as demais e de extrema importancia para o desenvolverdos.

3.3. Impressoras

Classe responsável pela impressão dos documentos contidos no buffer da Pool de Impressão 3.2. Ela é notificada pelo escalonador de impressões de que novos documentos chegaram, e que os mesmos podem ser impressos .

4. Conclusão

Neste relatório foi apresentado a estrutura básica de um.

Referências

[1] E. Software, "Pool de impressão – você sabe sua finalidade?." `http://www.devmedia.com.br/java-sockets-criando-comunicacoes-em-java/9465`, 2017. [Online; acesso em 22-Outubro-2017].

[2] Caelum, "Apêndice - Sockets." https://www.caelum.com.br/apostila-java-orientacao-objetos/apendice-sockets/, 2017. [Online; acesso em 18-Outubro-2017].

[3] Caelum, "Apêndice - Problemas com concorrência." https://www.caelum.com.br/apostila-java-orientacao-objetos/apendice-problemas-com-concorrencia/, 2017. [Online; acesso em 20-Outubro-2017].