Redes De Computadores - Trabalho Prático 1

Alunos: Renato Sérgio Lopes Júnior - 2016006875
Tiago Negrisoli de Oliveira - 2016006956

1 Introdução

Neste trabalho foi desenvolvido um sistema de log remoto, consistindo de um servidor e um cliente. O cliente envia mensagens de log para o servidor, e este salva essas mensagens em um arquivo. Vários clientes podem enviar mensagens para o mesmo servidor ao mesmo tempo. As mensagens são codificadas e enviadas pela rede por meio de pacotes UDP.

2 Desenvolvimento

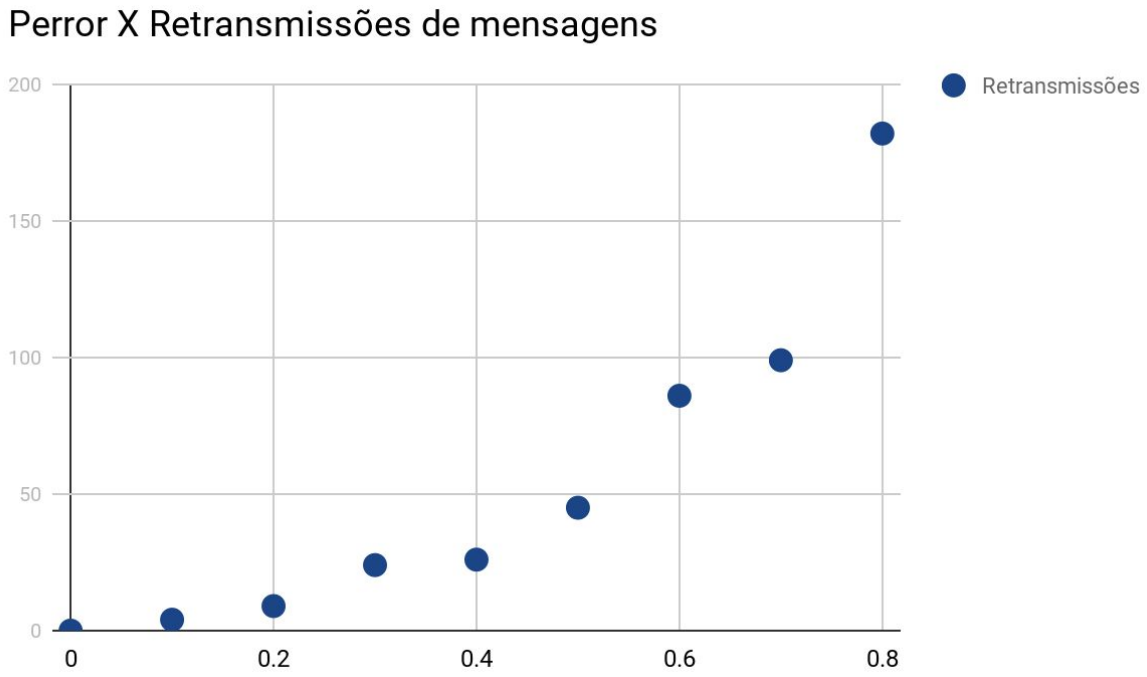
O trabalho foi implementado na linguagem Java. Foi usada a classe `java.net.DatagramSocket` para a criação dos Sockets de comunicação entre o Cliente e o Servidor.

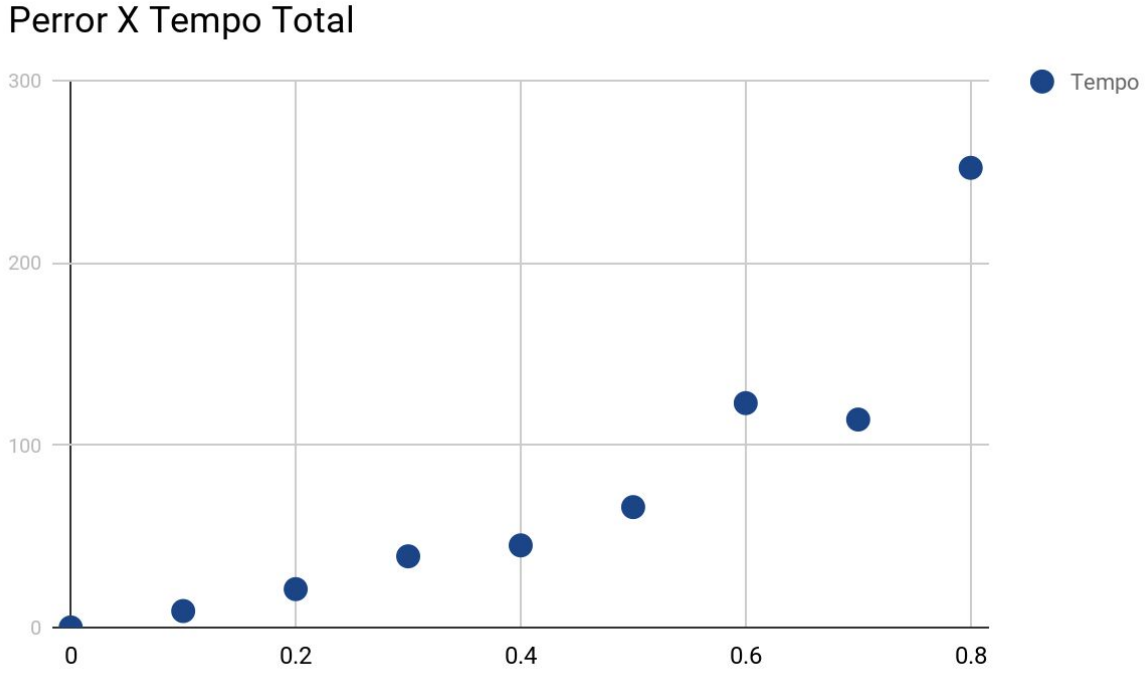
Durante o trabalho algumas dificuldades foram encontradas: como o servidor irá identificar o cliente e guardas as informações de cada cliente e como implementar o *timeout* das mensagens. A primeira foi resolvida com a criação da classe `server.ClientIdentifier`, que usa o IP e a porta do pacote para identificar aquele cliente. Para guardar as informações, que no caso é a janela deslizante de recepção alocada para aquele cliente, foi usado um HashMap, onde a chave é a representação em string da classe `ClientIdentifier` e o item é a referência para a janela de recepção do cliente com aquele identificador. Para se implementar o *timeout*, foi usada a seguinte estratégia: na janela deslizante, classe `window.ClientSlidingWindow`, foi criado um outro vetor que guarda o horário em que o pacote foi adicionado em millisegundos (Obtido com `System.currentTimeMillis()`). A cada iteração do loop principal do cliente, é verificado quais pacotes já estouraram o *timeout*. Aqueles que estouraram são reenviados e o tempo deles é definido como o tempo atual.

No mais, o desenvolvimento se deu de maneira tranquila, com as dúvidas encontradas sendo solucionadas após pesquisas na Internet.

3 Análises

A seguir são apresentados os gráficos que ilustram o desempenho do programa desenvolvido. Para a construção dos dados, foram usados os seguintes parâmetros: tanto a janela deslizante do servidor quanto a do cliente tem tamanho 5, a probabilidade de erro no servidor é 0 e o timeout do cliente é de 3s e o cliente lê um arquivo com 50 mensagens. Assim, foi feita uma variação da probabilidade de erro no cliente para verificar o seu impacto no Número de retransmissões de mensagens e no tempo para o cliente enviar todas as mensagens.





Por meio da análise desses gráficos, é possível verificar que quanto maior a probabilidade de erro, maior será o número de retransmissão de mensagens e, consequentemente, o tempo para enviar todas as mensagens. Isso mostra o porquê da Internet ficar extremamente lenta quando se há um problema no cabo, por exemplo. Como um grande número de pacotes terá que ser retransmitido, perde-se latência e o uso total da largura de banda, tornando a Internet “lenta”.

4 Instruções de compilação e execução

O trabalho foi desenvolvido utilizando a IDE NetBeans IDE 9.0. Entretanto, foi criado um Makefile para possibilitar uma fácil compilação sem usar a IDE.

Para compilar e gerar os JARs:

1. Abra um terminal na pasta src.
2. Execute o seguinte comando para compilar:
   ```
   $ make
   ```
3. Após a execução do comando acima, execute o **servidor** com o seguinte comando:
   ```
   $ java -jar server.jar arquivo port Wrx Perror
   ```
4. Para executar um **cliente**, use o seguinte comando:
   ```
   $ java -jar client.jar arquivo IP:port Wtx Tout Perror
   ```

5 Conclusão

O programa foi desenvolvido de acordo com a especificação disponibilizada. Alguns problemas foram encontrados durante a implementação, mas foram resolvidos. Por fim, o programa desenvolvido apresentou os resultados esperados.

Respondendo à pergunta proposta na especificação, se servidor receber duas mensagens com o mesmo número de sequência e hashes MD5 corretos, mas conteúdo (texto) diferente, um caso anômalo, ele não verificará se o conteúdo das mensagens é igual. Assim, a primeira que chegar será escrita no arquivo de saída e a segunda será apenas confirmada, visto que outra mensagem com o mesmo seqnum já foi recebida e confirmada.