REDES
TP1

Jean-Luc Coelho

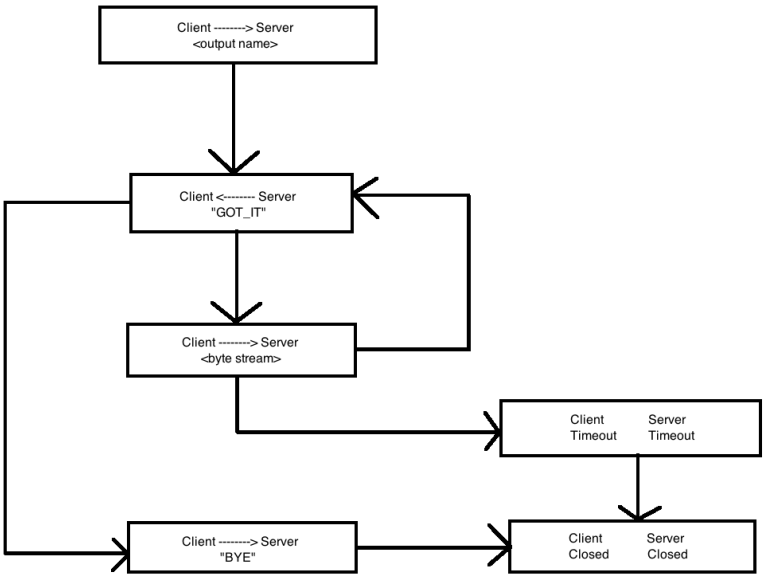
Introdução

O objetivo do trabalho era demonstrar domínio sobre a linguagem C e do protocolo IPV4/IPV6 através do protocolo de comunicação TCP. Para isso, foi requisitado que implementássemos um servidor e um cliente que comunicassem entre si. O servidor fica esperando até que o cliente envie uma lista de arquivos em um diretório especificado.

Metodologia

O método utilizado para fazer a comunicação entre o servidor e cliente foi um pouco diferente do especificado. Para evitar problemas de perda de pacotes, foi adicionada uma mensagem intermediária que confirma o recebimento de pacotes por parte do servidor. Quando o servidor demora a responder, ocorre um timeout. Isso garante que o usuário não ficará esperando por uma resposta do programa.

O fluxo de dados pela rede é demonstrado pelo gráfico a seguir:



Resultados:

Foram rodados diversos testes com variações de tamanho de buffer e de mensagem. Todos os testes foram rodados em máquinas que compartilhavam uma rede local.

Foram esses os resultados:

Varying message size:

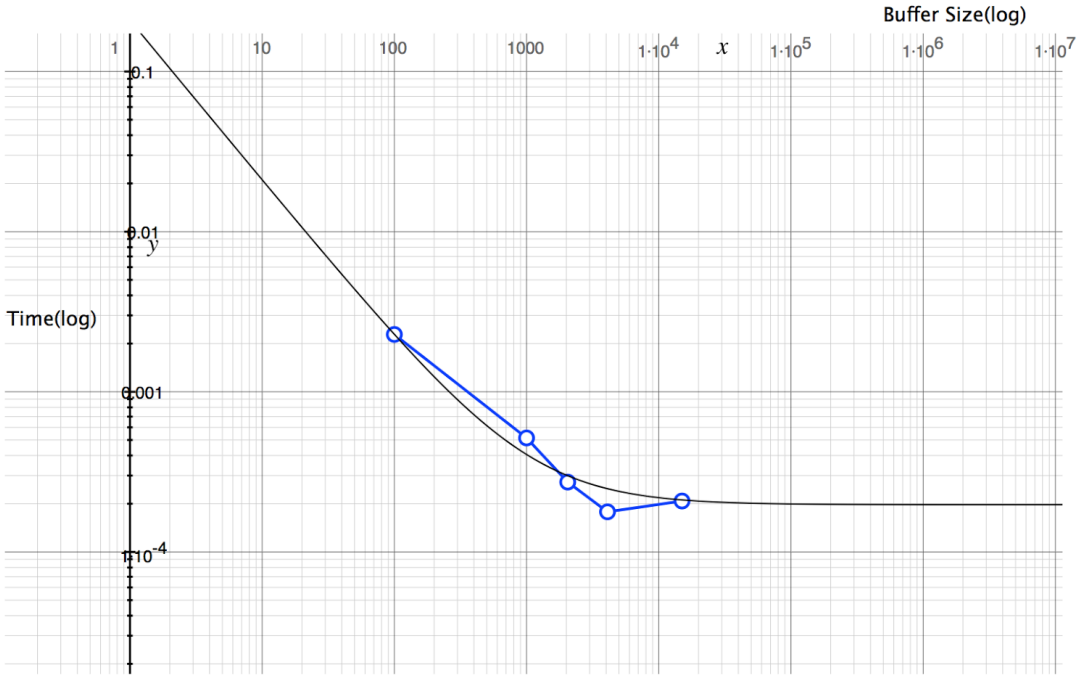
Buffer size: 10 | Message size: 10 | Time: 0.000206
Buffer size: 10 | Message size: 20 | Time: 0.000201
Buffer size: 10 | Message size: 30 | Time: 0.000275
Buffer size: 10 | Message size: 40 | Time: 0.000276
Buffer size: 10 | Message size: 50 | Time: 0.000507
Buffer size: 10 | Message size: 100 | Time: 0.000668
Buffer size: 10 | Message size: 1000 | Time: 0.003733
Buffer size: 10 | Message size: 10000 | Time: 0.028370

Varying buffer size:

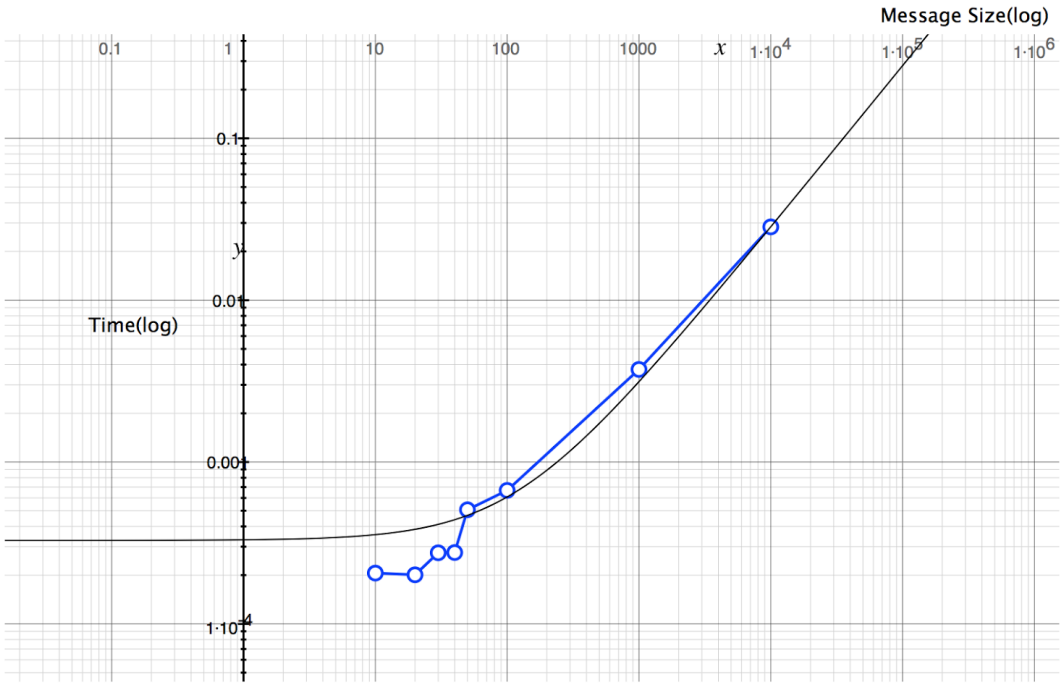
Buffer size: 100 | Message size: 10000 | Time: 0.002281
Buffer size: 1000 | Message size: 10000 | Time: 0.000515
Buffer size: 2048 | Message size: 10000 | Time: 0.000273
Buffer size: 4096 | Message size: 10000 | Time: 0.000178
Buffer size: 15000 | Message size: 10000 | Time: 0.000208

Plot dos resultados:

$y=a+\frac{b}{x}$, $a=1.9749e\text{-}4$, $b=0.2091$



$y=a\cdot x+b$, $a=2.8101e\text{-}6$, $b=3.2781e\text{-}4$



Detalhe: Foram usados sistemas de coordenadas logarítmicos para facilitar a visualização dos dados.

Análise dos dados:

Para os testes rodados, foram obtidos resultados bem previsíveis. Enquanto os testes de variação de tamanho de mensagem apresentaram um tempo com crescimento linear, os testes com variação de tamanho de buffer mostraram uma queda proporcional em relação ao tamanho do buffer, com assíntota no tamanho da mensagem. Houve muito jitter na transmissão dos dados por causa do comportamento da rede, porém isso é esperado.

Embora não tenham sido rodados testes grandes o bastante para que isso ocorra, é esperado que após uma certa quantidade, o crescimento do buffer comece a aumentar o tempo de propagação da mensagem, já que a memória deste começará a ser alocada em unidades de memória mais lentas. Além disso, pela implementação, o tamanho do buffer não interfere nas outras camadas de rede, já que ele sempre enviará streams de byte no máximo do tamanho das mensagens.

Conclusão:

O trabalho foi um grande sucesso. Os resultados foram satisfatórios e a execução foi impecável. Embora a implementação pudesse ter sido um pouco mais limpa e modularizada, ela foi suficientemente simples para que não ocorressem muitos problemas durante a implementação.

A maior dificuldade do trabalho foi com a documentação da API de sockets em C. Ela é muito mal documentada e não descreve o comportamento das funções de forma satifatória. Foi necessário muita pesquisa para realizar esse trabalho.

Bibliografia:

http://www.linuxhowtos.org/C_C++/socket.htm
https://github.com/Temennigru/TPR01
http://www.cplusplus.com/