Trabalho Prático 1:
Interface de sockets e medição de desempenho

Pedro Lopes Miranda Junior

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)

plmj@dcc.ufmg.br

Resumo. *O objetivo deste trabalho é (1) exercitar a interface de programação com sockets (em python), (2) se familiarizar com o ambiente virtualizado mininet para emulação de sistemas em rede e (3) medir e analisar o desempenho de aplicações simples neste ambiente. Para tanto, as seguintes etapas estão envolvidas:*

1. *Instalação do mininet e configuração de uma topologia.*
2. *Implementação/execução de programas de teste no mininet e medição de desempenho.*
3. *Análise de resultados e escrita do relatório.*

1. INTRODUÇÃO

Assim, para testar os conceitos de redes de computadores, foi requisitado o teste de dois pares cliente/servidor descritos abaixo.

Par 1: O cliente envia um certo número de mensagens para o servidor, sem paradas entre cada envio. Ao final, ele espera uma mensagem de um byte de volta do servidor.

Par 2: Cada vez que o cliente enviar uma mensagem, ele deverá esperar uma resposta de um byte do ser- vidor. O cliente termina depois de fazer esse processo um certo número de vezes.

2. SOLUÇÃO PROPOSTA

Para solucionar os problemas propostos foram feitos dois pares de cliente/servidor utilizando a linguagem *Python* e a biblioteca *socket* para fazer a conexão do cliente com o servidor. Outras bibliotecas usadas foram as *sys, random* e a *timeit*.
A primeira foi utilizada para recuperar os parâmetros passados na chamda do script. Já a segunda para criar as sequencias aleatórias de mensagens. A última para medir o tempo de execução da função *client* em cada um dos pares.
Os demais detalhes de implementação serão tratados na subseção a seguir

2.1. Par 1

Para o par 1 era necessário criar um cliente que enviava mensagens intermitentes esperando ao fim de todas elas uma resposta de 1 byte do servidor, e as premissas usadas em cada item do par estão descritas abaixo:

2.1.1. Cliente

O cliente recebe como parâmetro o número de mensagens e o tamanho de cada uma delas, cria as mensagens aleatórias que servirão de envio para o servidor e as envia, uma a uma, esperando ao fim uma resposta de um byte. Para o funcionamento correto do par o cliente envia ao fim das mensagens um sinal de fechamento da conexão apenas para envio, ficando de guarda quanto a resposta final do servidor, para só então fechar completamente a conexão.

A criação das mensagens foi feita separadamente do código do cliente para não interferir na análise de tempo de envio das mensagens do cliente para o servidor. Para essa medição foi utulizada a função Timer da biblioteca *timeit*

2.1.2. Servidor

2.2. Par 2

2.2.1. Cliente

2.2.2. Servidor

Esses parâmetros são o número de mensagens a serem trocadas entre os pares e o tamanho de cada uma das mensagens.
Elas foram criadas numa função separada e anteriormente a troca de mensagens entre o cliente e o servidor para não influenciar na medição do tempo.

3. AVALIAÇÃO EXPERIMENTAL

3.1. Resultados

3.2. Análises

4. CONCLUSÃO