TP 2 - Redes

Ingrid E. Spangler

O objetivo deste trabalho prático é fazer um servidor DNS que recebam entradas, realizam pesquisa se comunique com outros servidores para resolver hostnames.

Funções:

- **main()** é responsável por chamar a thread que executa o servidor na função connection_handler() e a thread que escuta a linha de comando, dns(). Pelo ARGV, opcionalmente, pode-se passar uma porta e o nome de um arquivo com comandos. Caso não passe a porta, a conexão do programa será aberta em uma porta definida como default, 8888.
- a **connection_handler()** cria um socket (na porta definida na função main()) e escuta por datagramas enviados por clientes para pesquisa de DNS. Caso haja alguma, esta função chama a função search()
- **dns()** é uma função que recebe as linhas de comandos do arquivo opcional ou, caso receber NULL, abre um loop for(;;) e os lê do stdin, faz o parsing e executa **add** (dentro do código mesmo, sem ser em uma função separada), **search** ou **link_srv**. **add** e **link_srv()** funcionam da mesma maneira, abaixo:
- **link_srv()** Em ambos os arrays, os endereços são adicionados de forma sequencial, um loop percorre a lista de entradas e insere a nova entrada na primeira posição vazia que encontrar. Como não há comando delete, não vai dar grandes problemas e resolvi não tratar isto.
- A função mais importante, a **search()** recebe um hostname, itera pelo array DNSTable até encontrar uma entrada correspondente ou uma entrada vazia, se encontrar o ip para aquele hostname, search() retorna a string para a função que a chamou. Caso não encontre neste loop, um socket UDP é aberto e em um loop pelas entradas da LookupTable de servidores DNS ligados a ele, envia uma mensagem com o hostname, espera durante um tempo, especificado pela variável global TIMEOUT_SEC.

Estruturas e decisões:

para guardar os endereços, fiz uma estrutura de dados Tentry que contém duas strings. A DNSTable é um array de Tentry de comprimento máximo MAXENTRIES - que é definida como variável global, a capacidade máxima de endereços. Também há uma estrutura TLink que guarda um ip (string) e uma porta (inteiro) de outros servidores DNS adicionados pelo comando link. O funcionamento é o mesmo para ambas.

A conexão ser IPV4 ou IPV6 depende de uma variável global IS_IPV4, como no TP anterior. Se for true, é IPV4, se for false, é IPV6.

(meu programa espera 1 segundo, mas se não for suficiente é só mudar) pela resposta. Caso obtenha uma resposta diferente de "-1", ele retorna esta resposta. Caso receba -1 ou não receba nada antes do timeout, retorna -1.

Defini a constante TIMEOUT_SEC como 1 segundo. Caso haja problemas com isto, é fácil mudar.

Das funções de socket, usei a flag SOCK_DGRAM para UDP, e a função setsockopt com a flag SO_RCVTIMEOUT para o timeout da função search.