Relatório Trabalho Prático 2
Redes de Computadores

Introdução

O objetivo deste trabalho é a implementação de um servidor DNS simplificado. Um servidor DNS faz a correspondência entre endereços web e seus endereços IP. A versão apresentada é capaz de adicionar informações de tradução ao seu banco de dados, se conectar a outros servidores e buscar endereços IP com base em um hostname, no banco local e nos servidores conectados.

Implementação

O servidor foi criado com base em quatro arquivos fonte. O arquivo `servidor_dns.c` é onde estão implementadas as funções que definem o comportamento do servidor. O arquivo `funcoes.c` cuida da parte de criação e definição de protocolos de comunicação entre servidores, e possui um header `funcoes.h` . Por fim, o `Makefile` é utilizado para ligar as bibliotecas necessárias e facilitar a compilação.

No arquivo `servidor_dns.c` temos a função `main`, que inicialmente cria uma thread separada para lidar com as requisições de outros servidores para o servidor atual. Depois é verificado se há um arquivo de texto como entrada ou não, as duas possibilidades são tratadas separadamente mas são análogas. É lido um comando, que será classificado como um dos comandos válidos ou como inválido. Todos os comandos requisitados produzem uma saída relatando resultados ao usuário.

No caso do comando `add`, é verificado se o hostname passado já existe nos dados do servidor, se sim, seu IP é atualizado, se não, é criada uma nova entrada para ele. No comando `search`, é chamada uma função para buscar o hostname solicitado. O comando `link` cria uma nova entrada para o servidor indicado nos dados do servidor atual. Caso seja digitado um comando inválido, ele é ignorado.

No mesmo arquivo, temos a função `aguarda_conexao`, utilizada pela thread criada no início da função `main`. Ela recebe como parâmetro a porta do servidor atual e não tem valor de retorno. Nela é criado um socket UDP para o servidor atual, e são aguardadas solicitações de outro servidor em um loop. No loop é criado um `sockaddr` para cada servidor que envie uma requisição e a thread fica aguardando o recebimento de um pedido. Quando chega uma solicitação, o servidor chama a mesma função de busca de IP chamada pela `main` e envia de volta o resultado.

Por fim, a função `busca_hostname` recebe como parâmetros o hostname que deve encontrar e um array para colocar o endereço IP caso ele seja encontrado, e retorna o valor 1 caso encontre e 0 caso contrário. Primeiramente a busca é feita nos dados do servidor atual. Caso não seja encontrado, a função itera pela lista de servidores conectados e solicita a um de cada vez para procurar o endereço. Para isso o servidor age como um cliente, criando um socket UDP para si e enviando a mensagem com o hostname procurado. Ela então aguarda a resposta do outro servidor e a examina, caso seja positiva, o endereço IP é retornado, caso seja negativa ela passa para o próximo servidor conectado da lista. Foi adicionado um timeout para que o servidor não fique indefinidamente aguardando a resposta do outro. Se a função iterar sobre todos os servidores ligados e não obtiver sucesso, ela retorna o valor 0, indicando que não encontrou o endereço IP do hostname solicitado.

No arquivo `funcoes.c`, a função `logexit` ajuda a encontrar a localização de erros na execução. `client_init` inicializa os atributos de um agente que está na função de cliente em uma comunicação, enquanto `server_init` faz o mesmo para aquele que está na função de servidor.

Estruturas

Foram utilizadas estruturas `sockaddr_storage` para armazenar os dados da estrutura `sockaddr`, já que a estrutura `sockaddr` é pequena, e assim evitamos preocupações com seu

tamanho durante a execução. Após ser instanciada `sockaddr_storage` passa por um cast para `sockaddr_in` ou `sockaddr_in6`, dependendo do protocolo usado. Essa estrutura é inicializada na função `client_init` ou `server_init`. No caso de `server_init`, há uma constante `int versao` que deve ser manualmente modificada para 0 ou 1 dependendo da versão do protocolo que se deseja usar (IPv4 ou IPv6).

Utilizamos uma estrutura `pthread_t` para os dados da thread que aguarda conexões de outros servidores.

Para armazenar as listas de hostnames, endereços IP e portas foram utilizados arrays globais, para que pudessem ser acessados por todas as funções no arquivo `servidor_dns.c` , seu tamanho máximo é definido no início do arquivo, e deve ser modificado caso se deseje inserir mais do que mil entradas em alguma das tabelas.

Para as mensagens trocadas pelo servidor e cliente foram usados vetores de chars, cuja primeira posição sempre contém o tipo da mensagem (1 ou 2) e as demais variam de acordo com esse tipo.