# Comunicações Por Computador

Trabalho Prático 2

Grupo PL6-03:

Uma imagem com céu, exterior, pessoa, terra

Descrição gerada automaticamente Uma imagem com pessoa, exterior, céu, homem

Descrição gerada automaticamente

A95460

A96726

Ivo Miguel Alves Ribeiro A96726

Diogo Luís Almeida Costa A95460

[Comunicações Por Computador 1](#_Toc120211338)

[ServerS 13](#_Toc120211339)

[Query 14](#_Toc120211340)

[Dificuldades com que fomos deparados 14](#_Toc120211341)

[Tabela de empenho dos elementos do Grupo 15](#_Toc120211342)

[Conclusão 15](#_Toc120211343)

INTRODUÇÃO AO TRABALHO

ARQUITETURA DO SISTEMA

TOPOLOGIA

Na arquitetura do sistema, acabamos por construir uma topologia com um dominio de topo, dois dominios(.bacalhau;.robalo), dois subdominios(batata.bacalhau;arroz.robalo) e um dominio reverse.

//imagens

Na nossa topologia, o dominio de topo tem dois servidores de topo, nos quais estes são ambos primarios. Para os dois dominios (.bacalhau.moc; .robalo.ude), estes são compostos por um servidor primario, dois servidores secundarios, um servidor de resposta, dois mxs, um webserver e um cliente. Para os dois subdominios(.batata.bacalhau.moc; .arroz.robalo.ude) estes são composto de igual maneira aos dois dominios referidos anteriormente. Para finalizar, o dominio reverse é constituido por um servidor primario.

COMPONENTES DO SISTEMA

SERVIDOR PRIMARIO(SP)

O nosso modelo de servidor primario é um servidor DNS que irá responder e efetuar queries DNS, para alem disso estes serao a autoridade so seu proprio dominio DNS e por consequente irão ter acesso direto á base de dados do dominio. Qualquer informaçao importante que deva ser adicionada ou alterada num dominio DNS terá de ser feita diretamente na base de dados do SP. Aditivamente, para alem dos dados relativos ao seu dominio, o SP terá acesso a uma configuraçao especifica na qual sabe todos os dominios para qual é SP, todas as portas de atendimento, identificaçao do ficheiro da base de dados, identificaçao do ficheiro de log, informaçao de segurança para acesso ás bases de dados, identificaçao dos SS respetivos e dos SP dos ssubdominios e endereços dos servidores de topo. Para este projeto, cada SP terá um imput de um ficheiro de configuraçao, um ficheiro de base de dados para cada dominio gerido e um ficheiro com a lista de servidores de topo.Como output terá apenas um ficheiro log.

SERVIDOR SECUNDARIO(SS)

O nosso modelo de servidor secundario será um servidor DNS que irá reponder e efetuar queries DNS, este não terá acesso direto á base de dados, mas terá autoridade para ter uma replica da base de dados original do SP do seu dominio, e esta poderá ser atualizada quando necessario. Para alem disso, o SS tem de ter acesso a uma configuraçao especifica na qual este terá a informaçao dos dominios para qual o servidor é secundario, das porta de atendiento, da identificaçao do SP dos dominios para qual é SS, da identificaçao do ficheiro de log, da informaçao de segurança para acesso ao SP, dos endereços de servidores se topo. Para este projeto, cada SS terá um imput de um ficheiro de configuraçao e um ficheiro com a lista de servidores de topo.Como output terá apenas um ficheiro log. Aditivamente, neste projeto foi decidido que a informaçao da base de dados replicada do SP será armazenada apenas em memoria volatil.

SERVIDOR DE RESPOSTA(SR)

O nosso modelo de servidor de resposta será um servidor DNS que irá responder e efetuar queries DNS sobre qualquer dominio, este servirá apenas de intermediario, pois este não ira ter autoridade sobre nenhum dominio. Um SR poderá ser mplementado a muitos niveis da rede, desde um processo a cada aplicaçao cliente até um servidor DNS que responde aos clientes duma rede IP local, aos clientes dos provedores de serviçoes ou aos clientes de uma instituiçao.Um SR terá acesso a uma configuraçao especifica na qual irá ter a informaçao dos dominios e lista de servidores DNS que deve contactar, das portas de atendimentos, da indentificaçao do ficheiro lod, dos endereços dos servidores de topo. Para este projeto, cada SR terá um imput de um ficheiro de configuraçao e um ficheiro com a lista de servidores de topo.Como output terá apenas um ficheiro log.

CLIENTE(CL)

O nosso modelo de cliente será um processo que vai precisar da informaçao da base de dados de DNS dum determinado dominio. Para obter essa informaçao, irá realizar queries DNS a um SR(para já na 2º fase irá perguntar diretamente a um SP ou SS sobre a base de dados do dominio na qual pretenderá obter informaçoes). Habitualmente, um CL teria um ficheiro de configuraçao com uma lista de SR, mas, para este projeto, cada cliente terá o imput e o output atraves da linha de comando, sem necessidade de ficheiro de configuraçao. ´

SERVIDORES DE DOMINIOS DE TOPO(SDT)

O nosso modelo de serviodor de dominio de topo é basicamente igual aos SP ou aos SS (ou seja, um SP ou SS autoritários para um domínio de topo é um SDT) ainda que estes não tenham domínios hierarquicamente acima na árvore DNS.

SERVIDORES DE TOPO(ST)

O nosso modelo de servidor de topo será como um SP, mas terão apenas uma base de dados para cada dominio de topo, onde inclui informação dos SDT respetivos (os nomes e os endereços IP dos seus SS e do seu SP).

FICHEIROS DO SISTEMA

Neste projeto foram definidos alguns ficheiros de configuração, de dados e de log com uma sintaxe predefinida. Os ficheiros de configuração serão apenas lidos e processados no início do componente de software a que dizem respeito e moldam o seu comportamento. Os ficheiros de dados também são consultados apenas no início e a sua informação deverá ter de ficar armazenada em memória. Para alterar ou atualizar o comportamento dos servidores com informação modificada nos ficheiros de configuração será necessário reiniciar os servidores.

//IMAGES Exemplo de um dos nosso ficheiros de configuração

FICHEIRO DE LOG

Os ficheiros de log irão registar toda a atividade relevante do componente. Neste momento apenas temos um único ficheiro de logs geral que regista a atividade a qualquer momento.

//images

FICHEIRO DE DADOS

Existem ainda ficheiros de base de dados para cada servidor onde esta contida informação relevante a esses mesmos servidores, que são acedidos ao iniciar os servidores a eles associados. Essa informação fica então guardada em cache (uma cache por servidor).

//IMAGE

**MODELO COMUNICATIVO**

Na terceira secção do relatório vamos detalhar o modelo comunicativo, explicando os PDU criados e todas as formas de interação possíveis entre os elementos da arquitetura. Além disso, vamos tentar indicar todas as situações de erro e como o sistema lida com elas.

**Modelo Comunicacional do Sistema**

O modelo comunicacional é baseado em interaçoes assíncronas feitas através de mensagens encapsuladas no protocolo UDP. Esta mensagem DNS terá um cabeçalho fixo e os dados(este ocupam ate 1 kbyte). Os dados estão divididos em quatro partes:

* Os dados da query original
* Os resultados dessa query
* Informaçao dos servidores com autoridade sobre os dados da resposta
* Informaçao adicional indiretamente ligada aos resultados dessa query

//IMAGE FORMATO MENSAGEM

Esta mensagem assincrona será responsavel pelo transporte da query do cliente para os servidores e pelo transporte da resposta da query para o cliente. Esta mensagens está dividida em varios campos, terão de ser implementados da seguinte maneira:

* MESSAGE ID – identificador de mensagem que irá ser usado para relacionar as respostas recebidas com a query original;
* FLAGS – devem ser suportadas as flags Q, R e A; a flag Q ativa indica que a mensagem é uma query, senão é uma resposta a uma query; se a flag R estiver ativa na query indica que se deseja que o processo opere de forma recursiva e não iterativa; se a flag R estiver ativa na resposta indica que o servidor que respondeu suporta o modo recursivo; se a flag A estiver ativa na resposta indica que a resposta é autoritativa ;
* RESPONSE CODE – indica o código de erro na resposta a uma query; se o valor for zero então não existe qualquer tipo de erro e a resposta contém informação que responde diretamente à query; essa resposta deve ser guardada em cache; se houver erros, o sistema deve suportar os seguintes códigos de erro: 1, o domínio incluído em NAME existe mas não foi encontrada qualquer informação direta com um tipo de valor igual a TYPE OF VALUE; este caso é identificado como resposta negativa e pode ser guardada em cache; 2, o domínio incluído em NAME não; este caso também é identificado como resposta negativa e pode ser guardada em cache; 3, a mensagem DNS não foi descodificada corretamente;
* NUMBER OF VALUES – número de entradas relevantes que respondem diretamente à query e que fazem parte da lista de entradas incluídas no campo RESPONSE VALUES;
* NUMBER OF AUTHORITIES – número de entradas que identificam os servidores autoritativos para o domínio incluído no RESULT VALUES;
* NUMBER OF EXTRA VALUES – número de entradas com informação adicional relacionada com os resultados da query ou com os servidores da lista de autoridades;
* QUERY INFO – informação do parâmetro da query (NAME) e o tipo de valor associado ao parâmetro (TYPE OF VALUE); os tipos suportados são os mesmos suportados na sintaxe dos ficheiros de base de dados dos SP; na resposta a queries, os servidores devem copiar a informação do QUERY INFO e incluí-la na mensagem de resposta;
* RESPONSE VALUES – lista das entradas que fazem match no NAME e TYPE OF VALUE incluídos na cache ou na base de dados do servidor autoritativo; cada entrada deve ter a informação completa tal como é definida na base de dados DNS do SP do domínio referente ao NAME;
* AUTHORITIES VALUES – lista das entradas que fazem match com o NAME e com o tipo de valor igual a NS incluídos na cache ou na base de dados do servidor autoritativo; cada entrada deve ter a informação completa tal como é definida na base de dados DNS do SP do domínio referente ao NAME;
* EXTRA VALUES – lista das entradas do tipo A e que fazem match no parâmetro com todos os valores no campo RESPONSE VALUES e no campo AUTHORITIES VALUES de forma a que o elemento que o CL ou servidor que recebe a resposta não tenha que fazer novas queries para saber os endereços IP dos parâmetros que vêm como valores nos outros dois campos; cada entrada deve ter a informação completa tal como é definida na base de dados DNS do SP do domínio referente ao NAME.

Segue se entao o exemplo da interaçao assincrona de um CL e um SP:

//IMAGE FIG 4 ENUNCUADO

De seguida, encontra se a resposta do SP ao CL:

//IMAGEM FIG6 ENUNCIADO

//Codificaçao binaria das menagens--------------------------------------------------------------------------

Neste trabalho, todas as interaçoes começam com o envio de uma query, como já sabemos, essa query vem de um CL ou de um Servidor DNS.Em baixo, segue se o exemplo de todos os passos desde o envio de uma query de um cliente para um servidor SP ou SP:

//IMAGEM PAINT

Estes passos passam resumidamente por o cliente enviar a query para o SP, o servidor descodificar corretamente a query, de seguida este irá primeiro aceder á sua cache para procurar a informaçao necessaria para a resoluçao da query. Se este não for bem sucedido, este irá procurar na sua base de dados, se este for bem sucedido irá retornar a query resolvida para o cliente com todos os campos devidamente preenchidos.

O mesmo vai acontecer com o SS, a única diferença será a ocorrencia da transferencia de zona caso o SS não tenha a copia atualizada da base de dados do SP.

**TRANSFERENCIA DE ZONA**

Todas as interaçoes numa operaçao de transferencia de zona deverao ser feitas utilizando uma conexao TCP. O SS vai tentar a transferencia de zona se a base de dados do respetivo SP for mais atual que a sua copia. A transferencia ocorrerá com o SS a mandar o nome do dominio da qual quer receber a base de dados. De seguida, o SP vai verificar a validade desse dominio e se SS tem autorizaçao para receber a copia da sua base de dados. Se isto acontecer, o SP vai mandar o numero de entradas do ficheiro da base de dados primeiro, depois o SS vai verificar o numero de linhas de entrada que vai receber. Quando o SS verificar e aceitar esse numero, o SP vai mandar todas as entradas em formato de texto, em ordem crescente, uma de cada vez, enquanto isso, o SS vai recebendo as linhas e vai verificar se recebeu as entradas esperadas. Quando o tempo predefenido da transferencia acabar, a conexao TCP vai terminar e com isso a transferencia tambem. Um SP também não deve aceitar pedidos consecutivos de transferência de zona do mesmo SS com intervalo menor de SOARETRY segundos.

// imagem tranferencia de zona

**ESTRATEGIAS DE IMPLEMENTAÇAO**

O nosso trabalho irá ser composto por 6 classes, nomeadamente a class ServerP, Server S, Cliente, Query, Cache e Logs. De seguida, irei explicar cada uma das suas funcionalidades e opções de execução.

**ServerP**

Na classe ServerP, temos como objetivo a implementação de um servidor do tipo primário. Este servidor ao inicio ira incializar duas classes(cache e log), depois irá receber um ficheiro de configuraçao ”teste.txt” no qual vai guardar a sua informação nas variáveis locais com a ajuda da função ParserSP.De seguida, vai buscar o nome do ficheiro da sua base de dados e irá usar a função ParserCacheServer da class Cache para extrair a informação para a sua cache. Estas duas funções vão ser declaradas no ficheiro log.

Depois desta fase, o servidor estará apto para receber uma ligação, neste caso, so haverão duas possiveis por agora, uma ligaçao de um cliente onde ira receber uma query caso receba um “QE” , ou um pedido de transferência de zona por parte de um SS,caso receba um “domain:”. Estas duas ligações também vao ser declaradas no ficheiro logs.

Para a ligação com o cliente, o servidor vai receber a mensagem do CL num formato conciso e depois de já estar formulada a resposta este irá envia la de volta para o cliente. A rececao da querie e o envio da resposta da querie vao estar notificadas na logs.

//prints da alguma cena possivelmente

Para a ligação com o SS , o servidor vai receber como tínhamos referido anteriormente “domain:”. De seguida o SP vai enviar o numero de entradas da sua cache, espera a mensagem de confirmação so SS e depois envia um socket por cada linha de entrada retirada da cache.Cada mensagem enviada ou recebida neste processo de transferência vai ser notificada no logs.

ServerS

Na classe ServerS, temos como objetivo a implementação de um servidor do tipo primário. Este servidor vai ser quase igual ao SP no que toca á criação de uma cache, do log e da ligação com o cliente. As queries recebidas por este servidor vão ser descodificadas e resolvidas da mesma forma que fizemos com o SP.

A transferência de zona também vai ser feita quando iniciamos o servidor SS, mandando assim a mensagem “domain:” para iniciar o processo de transferência. Depois de receber o número de linhas, vai mandar mensagem a “ok + o número de linhas que recebeu do SP”. Quando o SP começar a mandar as linhas de entrada da base de dados, o SS vai receber e guardar na cache uma de cada vez. Cada mensagem recebida ou enviada neste processo de transferência vai ser notificada no logs.

**Cliente**

Na classe Cliente, temos como objetivo a implementação de uma aplicação cliente, este cliente ira ter como variável local apenas o seu Ip. Este cliente vai mandar socket para o SP ou SS, com uma querie em formato conciso já predefinida. Depois vai receber a resposta através de um socket.Tanto o envio da querie como a resposta recebida vao estar notificados nos logs.

**Logs**

Na classe Logs, pretendemos guardar informações de todas as açoes ocorridas nestes processos. Criando um ficheiro ao incio e ir acrescentando linhas com definidas pelo seu tipo de log, pela hora e pela informação revelada.

Query

A classe query vai ser a classe que vai trabalhar na descodificação da query e formulação da sua resposta. Para isso, as variáveis locais desta classe vão ser o id, Flags, nResponse, n Values, nAutho, nExtravalues(estes são os campos do HEADER Fields de uma mensagem DNS), mais o InfoName (nome do dominio), e Type(vai ser o tipo de dados que vão querer extrair da base de dados).

Primeiramente, esta classe recebe uma string (a mensagem concisa da query que chegou ao servidor), depois descodifica e faz o parse da mesma. Caso ocorra algum erro durante esta fase, o processo query termina e devolve a query com response code igual a 3. De seguida, verifica-se se o domínio da query é igual ao da base dados (para termos a certeza que esta a ser procurada a resposta no domínio certo). Caso não seja

Dificuldades com que fomos deparados

No decorrer desta fase do projeto a nossa principal dificuldade foi definir e com correção uma topologia correta, e por sua vez testar o nosso código no nosso ambiente de testes. Por isso e para não perder mais tempo do que o já perdido começamos pela implementação em código do trabalho em si. Ao usar este método achamos melhor criar uma classe para os servidores primários e outa para os secundários de maneira que fosse mais fácil a realização de testes, nomeadamente o teste da transferência de zona, no próprio IDE. Mais tarde apercebemo-nos que de facto não há necessidade da existência destas duas classes uma vez que são mesmo bastante idênticas. Só ainda não o fizemos à data de realização deste relatório por medo de ao fazermos essa alteração o nosso código deixar de ser em parte funcional.

Para alem disto também apenas criamos um único ficheiro de logs geral, o qual guarda neste momento toda a informação, da maneira pedida no enunciado.

Tabela de empenho dos elementos do Grupo

|  |  |  |
| --- | --- | --- |
| **Tarefas:** | **Elementos do Grupo:** | |
| Ivo | Diogo |
| Modelação da topologia |  |  |
| Elaboração do modelo de comunicação servidor/cliente |  |  |
| Elaboração do modelo de comunicação servidor primario/servidor secundario |  |  |
| Implementaçao da leitura dos fichieros de configuraçao |  |  |
| Implementaçao da leitura dos fichieros de base de dados |  |  |
| Implementaçao da comunicação clente/servidor |  |  |
| Implementaçao da comunicação servidor primario/servidor secundario |  |  |
| Implementaçao da escrita no ficheiro de logs |  |  |
| Testes em ambiente de teste e deteção de erros |  |  |
| Relatorio |  |  |

Conclusão