# 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

Indice

[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

Este projeto da cadeira de Comunicações por Comoutadores pretende consolidar os conhecimentos sobre o serviço DNS da arquitetura TCP/IP para esta primeira fase e UDP TCP já em uma fase posterior.

Pretende-se ainda consolidar competências de programação de aplicações distribuídas utilizando o paradigma dos sockets.

Para isso temos como objetivo criar um ambiente de testes que nos permita perceber o papel dos servidores de DNS que tornam o nosso dia a dia na internet bem mais simples.

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.

Uma imagem com mapa

Descrição gerada automaticamente

Figura -Topologia

Na nossa topologia, o dominio de topo tem dois servidores de topo, nos quais estes são ambos primarios. Para os dois dominios (.bacalhau;.robalo), 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;.arroz.robalo) estes são composto de igual maneira aos dois dominios referidos anteriormente. Para finalizar, o dominio reverse é constituido por um servidor primario que contem a informação do suposto sp do subdominio do reverse.

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.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Exemplo ficheiro 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.

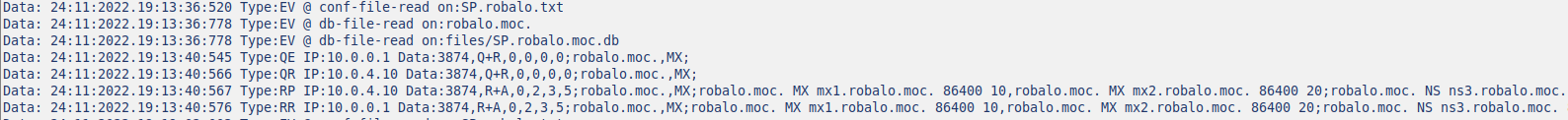


Figura - Exemplo ficheiro log

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).

Uma imagem com texto, mesa

Descrição gerada automaticamente

Figura - Exemplo ficheiro de dados

MODELO COMUNICATIVO

Modelo Comunicacional do Sistema

O modelo comunicacional é baseado em interações assíncronas feitas através de mensagens encapsuladas no protocolo UDP. Esta mensagem DNS terá um cabeçalho fixo e os dados (estes ocupam até 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

Uma imagem com mesa

Descrição gerada automaticamente

Figura - Composição mensagem DNS

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 que o elemento que o CL ou servidor que recebe a resposta não tenha de 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:

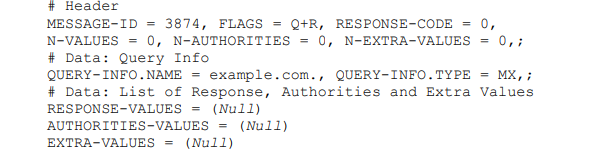


Figura -Query enviada pelo CL

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

Uma imagem com texto

Descrição gerada automaticamente

Figura - Query recebida pelo Cliente

Funcionamento da parte comunicacional

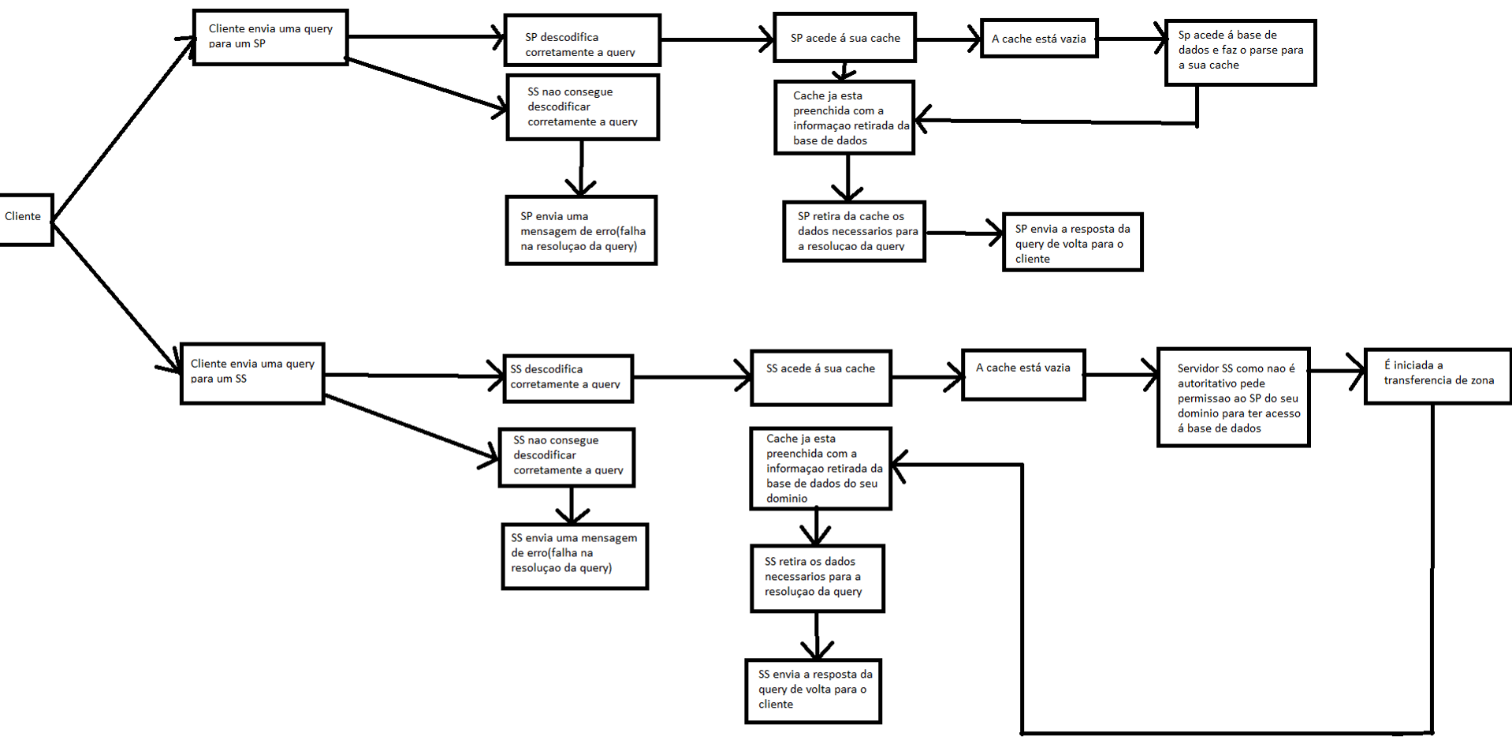
Na primeira parte deste trabalho, todas as interaçoes começam com o envio de uma query, como já sabemos, essa query vem de um CL ou de um Servidor DNS.Em baixo, segue se o exemplo de todos os passos desde o envio de uma query de um cliente para um servidor SP ou SP:

Figura - Esquema do funcionamento da comunicação

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 (explicada em baixo) 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.

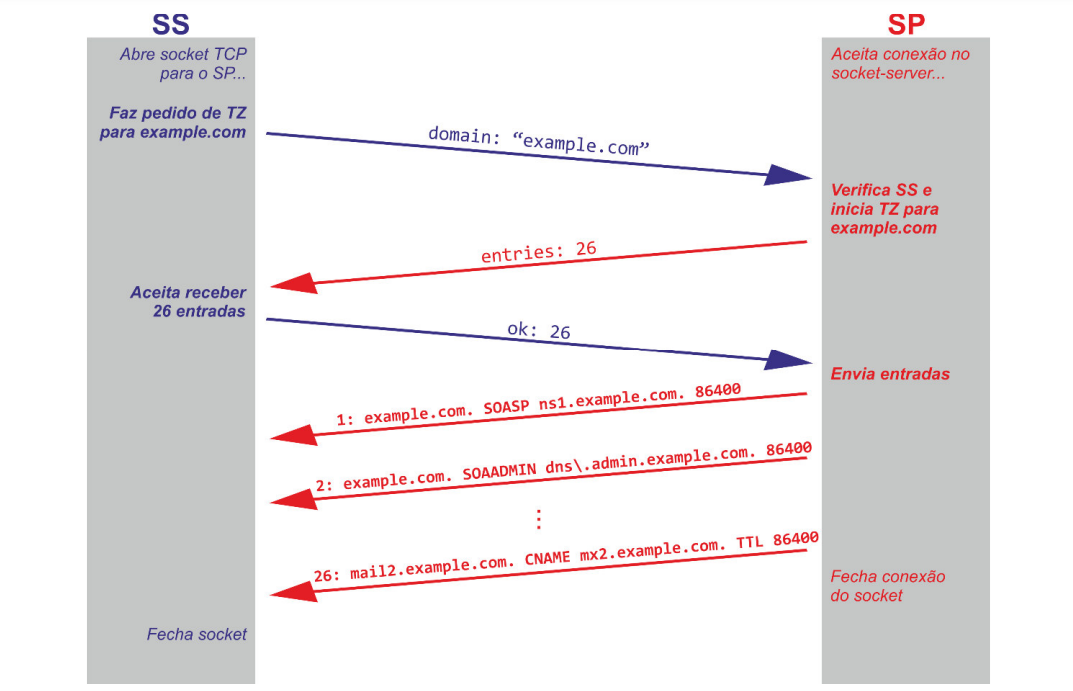
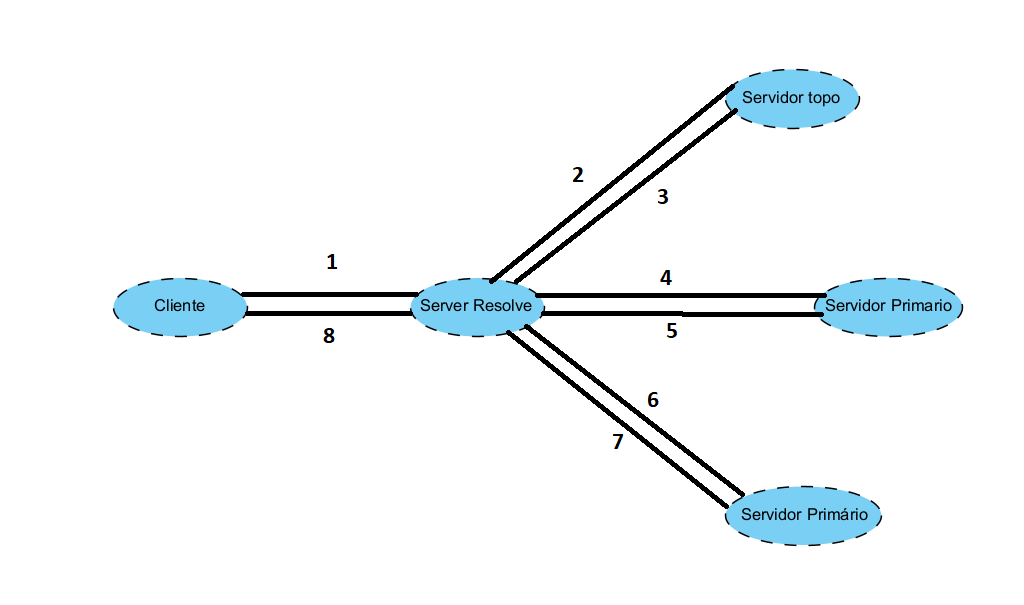


Figura - Exemplo transferência de zona

Na segunda parte deste trabalho, as interaçoes vao ser diferentes sendo estas divididas por 2 formas de interagir, interativa e recursiva.

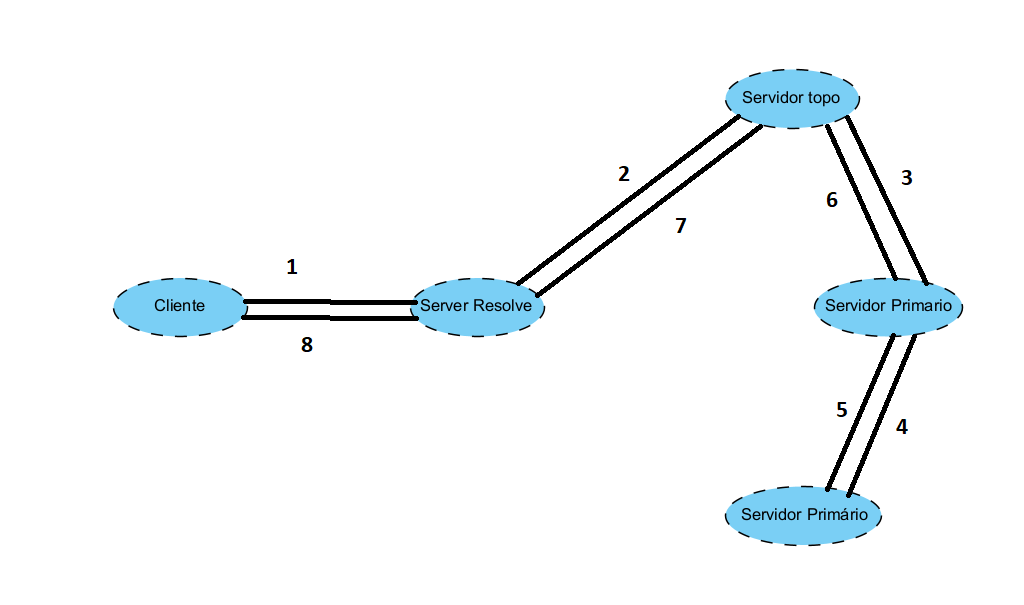
Parte interativa

****

Nesta parte, a interação vai começar por parte do cliente com uma mensagem para o SR(Server Resolve)(1), quando este recebe a mensagem vai verificar se o domínio da qual a query recebida na mensagem quer obter informações é igual ao domínio no qual o SR está .Se for pergunta a um servidor do domínio registado na query. Se não for esta vai mandar mensagem ao ST(servidor de topo) com a query recebida(2), este ST irá indicar o IP ao SR de um SP ou SS do domínio na qual a query pretende obter informações(3). Depois de o SR receber a mensagem este vai enviar então a query para o IP recebido com a query(4), e este ou ira entregar a resposta da query ou um IP de um servidor de um subdomínio ao SR(5).

Caso receba outro IP o SR vai repetir o mesmo procedimento com o SP do subdomínio(6)(7), caso receba resposta o SR envia para o Cliente a mensagem com a resposta recebida(8).

Parte recursiva



Nesta parte, a interação vai começar por parte do cliente com uma mensagem para o SR(Server Resolve)(1), quando este recebe a mensagem vai verificar se o domínio da qual a query recebida na mensagem quer obter informações é igual ao domínio no qual o SR está .Se for pergunta a um servidor do domínio registado na query. Se não for esta vai mandar mensagem ao ST(servidor de topo) com a query recebida(2) e vai aguardar pela resposta do mesmo, este vai perguntar a um servidor do domínio desejado pela query e vai aguardar pela resposta(3). Depois o SP que recebeu a mensagem se tiver a resposta a query vai enviar la de volta para tras(6), se não tiver a resposta vai perguntar a um SP do seu subdomínio(4), e este se tiver a resposta manda a de volta para o SP(5). Depois que o ST receba a resposta do SP a quem perguntou vai manda la para o SR(7), e este finalmente envia a mesma para o cliente(8).

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.

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, nValues, nAutho, nExtravalues (estes são os campos do” HEADER Fields” de uma mensagem DNS), mais o InfoName (nome do domínio), 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 resposta da 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, o processo query termina e devolve a resposta da query com response code igual a 2 e os restantes campos vazios. De seguida, vai procurar na base de dados o parâmetro type dado na query, se for sucedido termina o processo query com o response code igual a 0, as flags “R+A” (estas mantem se sempre com ou sem erro), o nValues igual ao número de respostas que encontrou, nAutho igual ao número de autoridades (SP e SS do domínio), e nExtravalues (número de parâmetro CNAME). Se não for bem-sucedido, termina o processo query e devolve a resposta da query com response code = 1. No final deste processo, a função mete a resposta em String e fica pronta para o servidor usar.

ServerP

Na classe ServerP, temos como objetivo a implementação de um servidor do tipo primário. Este servidor ao início ira inicializar duas classes (cache e log), depois irá receber um ficheiro de configuração ”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.

Nesta classe só irá existir uma função fundamental no qual basicamente irá receber uma query de um ST ou de um SR, se receber do ST é recursivo se for do SR é iterativo.

Se for iterativo, o SP vai receber uma mensagem do SR com a query, se esta for sinalizada com “smaller” significa que o SR procura o IP de um servidor do subdomínio onde o SP se encontra, e então o SP vai a sua base de dados , e devolve esse IP ao SR. Caso seja sinalizada com “reverse” então o SP esta no domínio reverse e ira devolver o nome do domínio do qual o IP do servidor que recebeu da query representa e ira devolver o nome ao SR. Caso não seja sinalizada, então significa que a query pode ser resolvida com o conteúdo da base de dados deste domínio .Resolvendo então a query e enviando uma mensagem com a resposta de volta para o SR.

Se for recursivo, o SP vai receber uma mensagem do ST com a query, e este vai ver se o local objetivo da query é o domínio onde o SP está. Se for este resolve a query e devolve para o ST a resposta. Se não for, manda a query para um dos seus servidores do seu subdomínio e aguarda a resposta do mesmo .Quando receber a resposta esta ira envia la para o ST.

Para a ligação com o SS, o servidor vai receber como tínhamos referido anteriormente “domain:”. De seguida o SP vai enviar o número de entradas da sua cache, espera a mensagem de confirmação só 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 vão estar notificados nos logs.

Logs

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

Cache

A nossa classe cache temos como objetivo a criação de uma cache especifica que possa ser implementa num modulo independente e que possa ser adicionada a qualquer servidor.

Nesta classe, iremos guardar toda a informação de uma base de dados, e cada entrada irá ficar guardada numa variável local da classe. Naturalmente, esta classe irá ter uma função que consiga fazer um parser para poder retirar e dividir toda a informação de um ficheiro base de dados.

ServerResolve

Na classe ServerResolve, temos como objetivo a implementação de um servidor do tipo primário. Este servidor ao início ira inicializar uma classe (query), depois irá receber um ficheiro de configuração, no qual vai guardar a sua informação nas variáveis locais com a ajuda da função ParserSP.

Nesta classe vamos ter 3 funções fundamentais na implementação e funcionamento da mesma. Primeiramente temos a “clienteServer” que irá fazer com que o servidor esteja apto a receber mensagens do cliente, depois deste receber a mensagem com a query, vai decifra -la e decidir para onde vai enviar a mensagem. Caso a query pretende obter informações no domínio onde o SR está, o SR vai usar a função ”SRSPDomain”, caso pretenda obter informações de outro domínio este vai usar a função ”SRST”. A função “SRSP” é usada só no modelo iterativo, na comunicação entre SR e SPs de outros domínios.

Servidor Topo

Na classe ServidorTopo, temos como objetivo a implementação de um servidor do tipo primário. Este servidor ao início ira inicializar uma classe (cache), depois irá receber um ficheiro de configuração, no qual vai guardar a sua informação nas variáveis locais com a ajuda da função ParserST, e um ficheiro de base de dados no qual vai guardar na sua cache.

Nesta classe vamos ter uma função fundamental chamada “STservidor” na qual vai receber a mensagem com a query, este irá decifrar a query para saber se a mesma é recursiva ou iterativa. Se for iterativa ele vai procurar o local objetivo da query, se encontrar manda o IP do SP do domínio desejado pela query de volta para o SR. Se não encontrar, então sabemos que este faz parte de um subdomínio, e manda o IP do SP do domínio na qual o subdomínio faz parte com a indicação disso mesmo “smaller”.

Se for recursiva, este irá enviar a mensagem com a query para o domínio no qual o local objetivo da query está.

Ambiente de testes

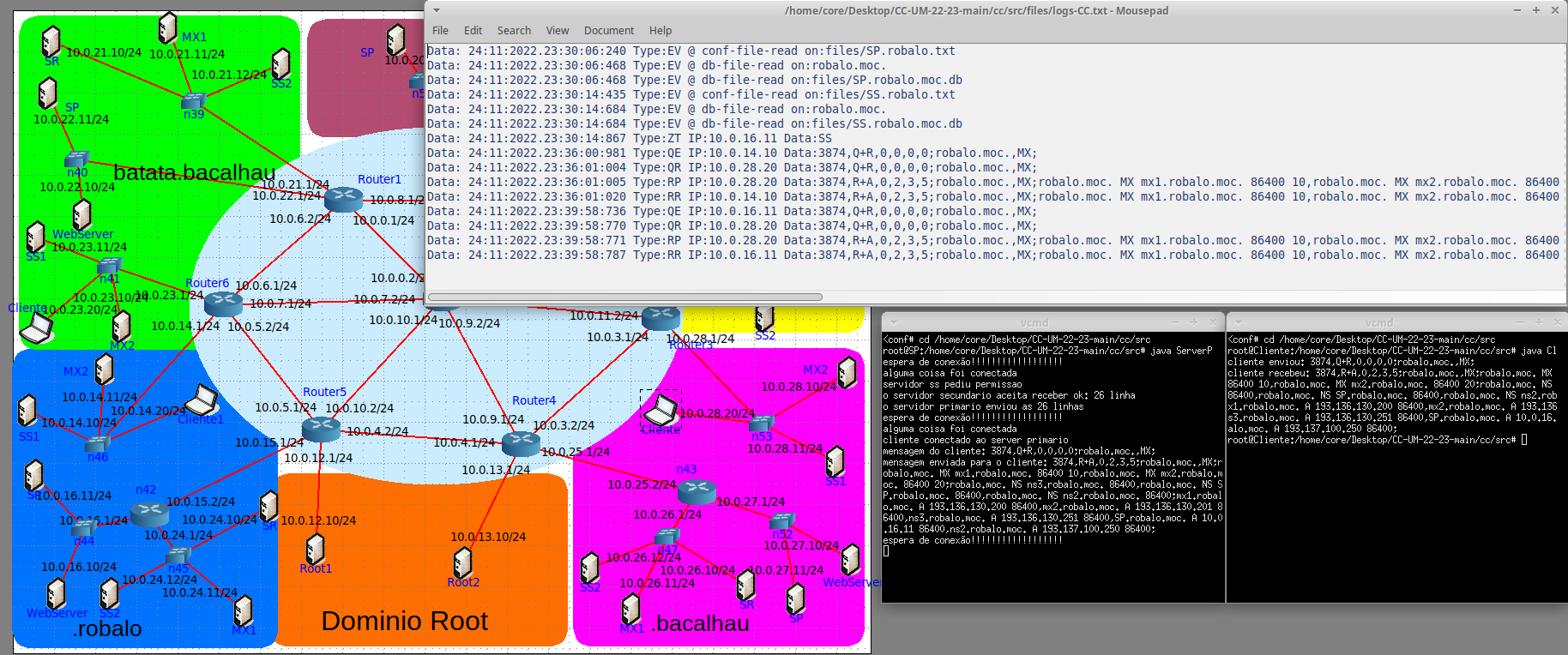
****

Figura - Print do funcionamento do envio de uma query de um Cl para um SP no Core

Uma imagem com mapa

Descrição gerada automaticamente

Figura - Print da ocorrência de uma transferência de zona no Core

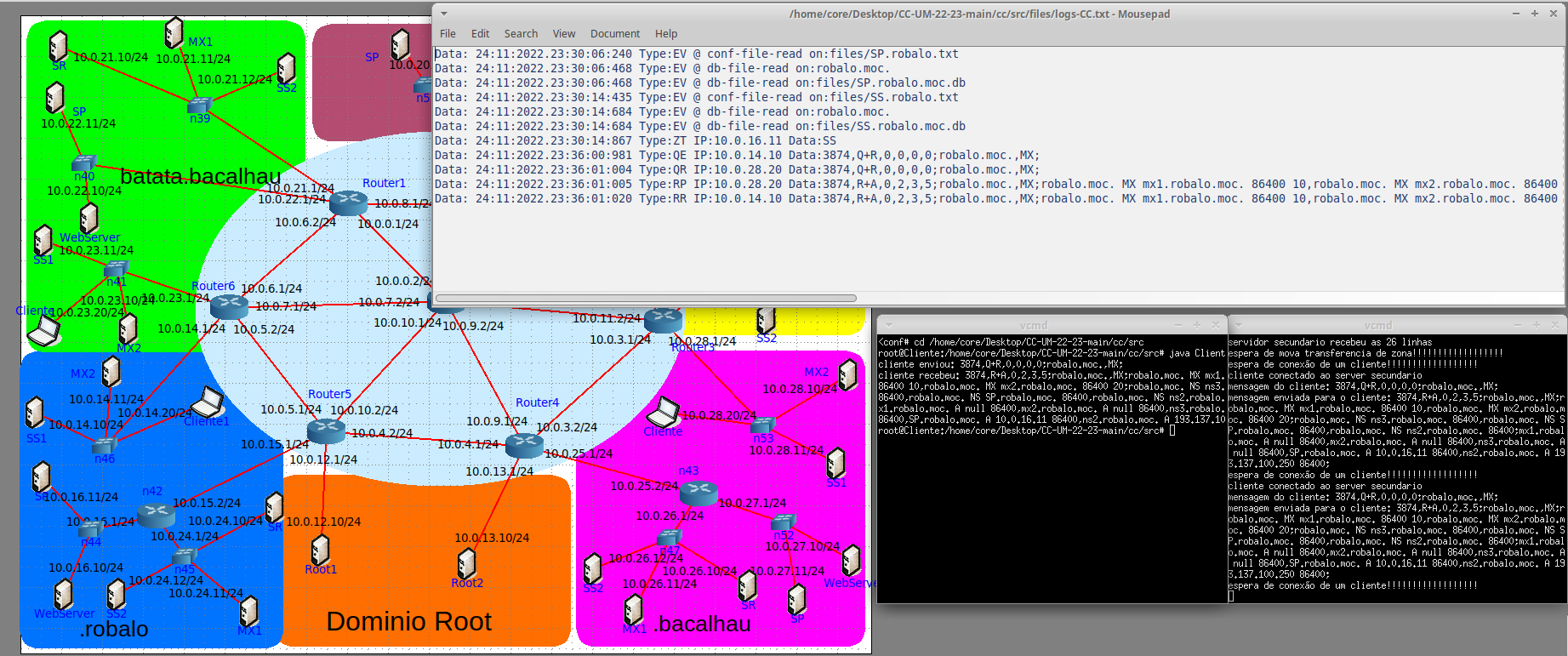


Figura - Print do funcionamento do envio de uma query de um Cl para um SS no Core

Ambiente de testes da parte dois

Uma imagem com texto

Descrição gerada automaticamente

Figura - Exemplo de um ficheiro de logs, neste caso o global

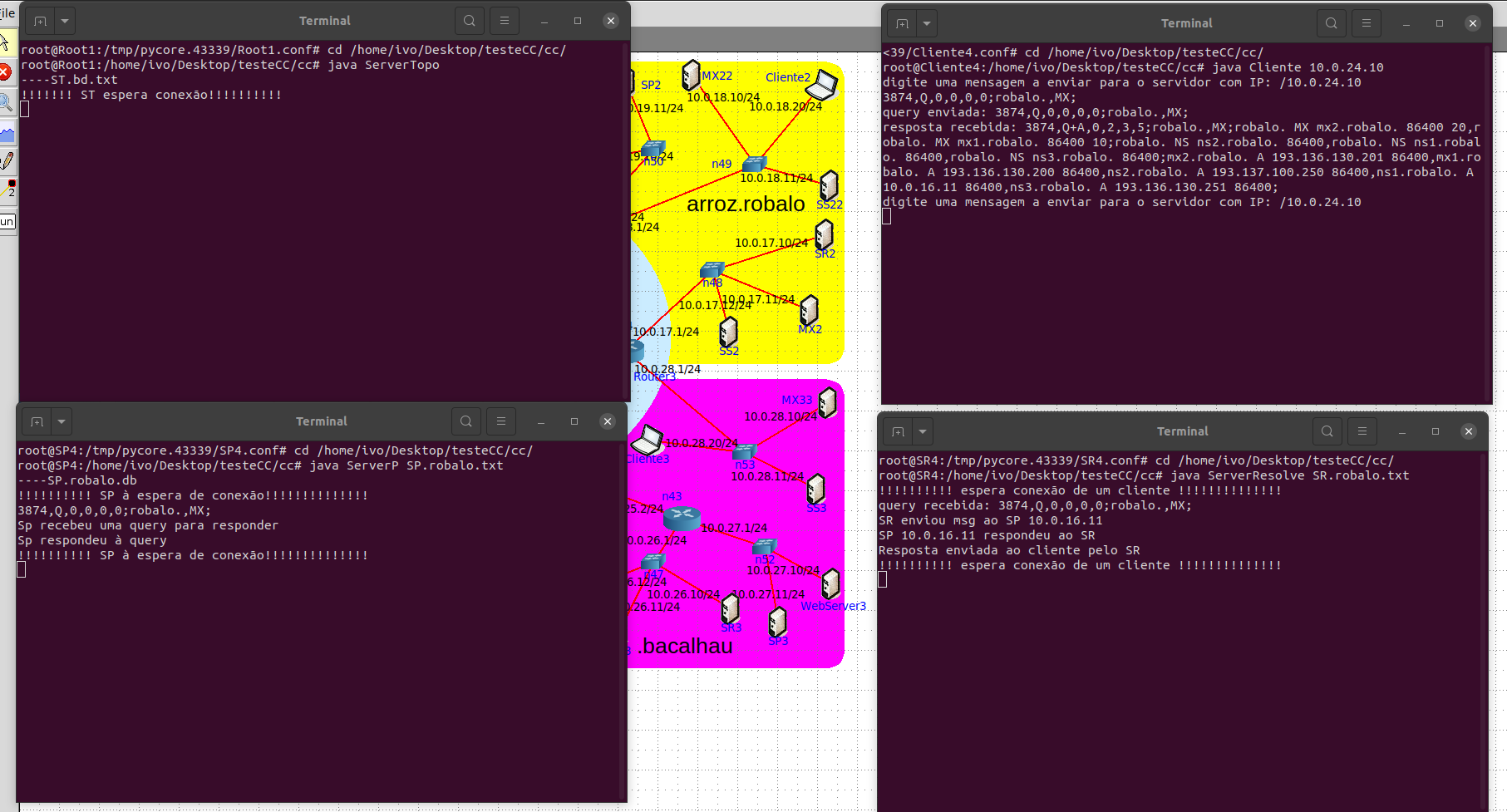


Figura 14 - Print do funcionamento do envio de uma query de um Cl para um SR e respondida por um SP do mesmo dominio

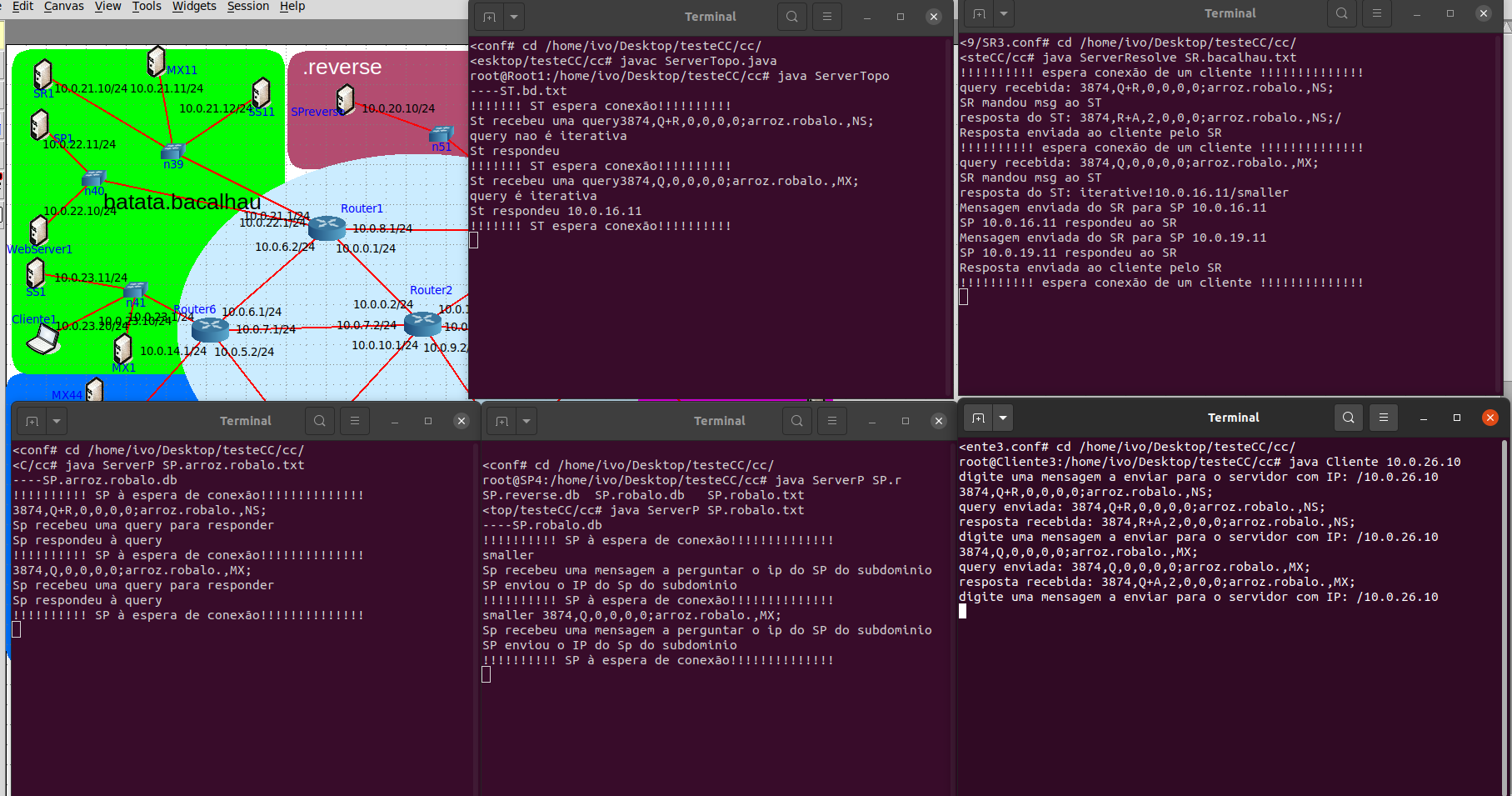


Figura 15 - Print do funcionamento do envio de uma query de um Cl para um SR de um domínio e respondida por um SP de um subdomínio de um outro domínio de maneira iterativa pelo sistema

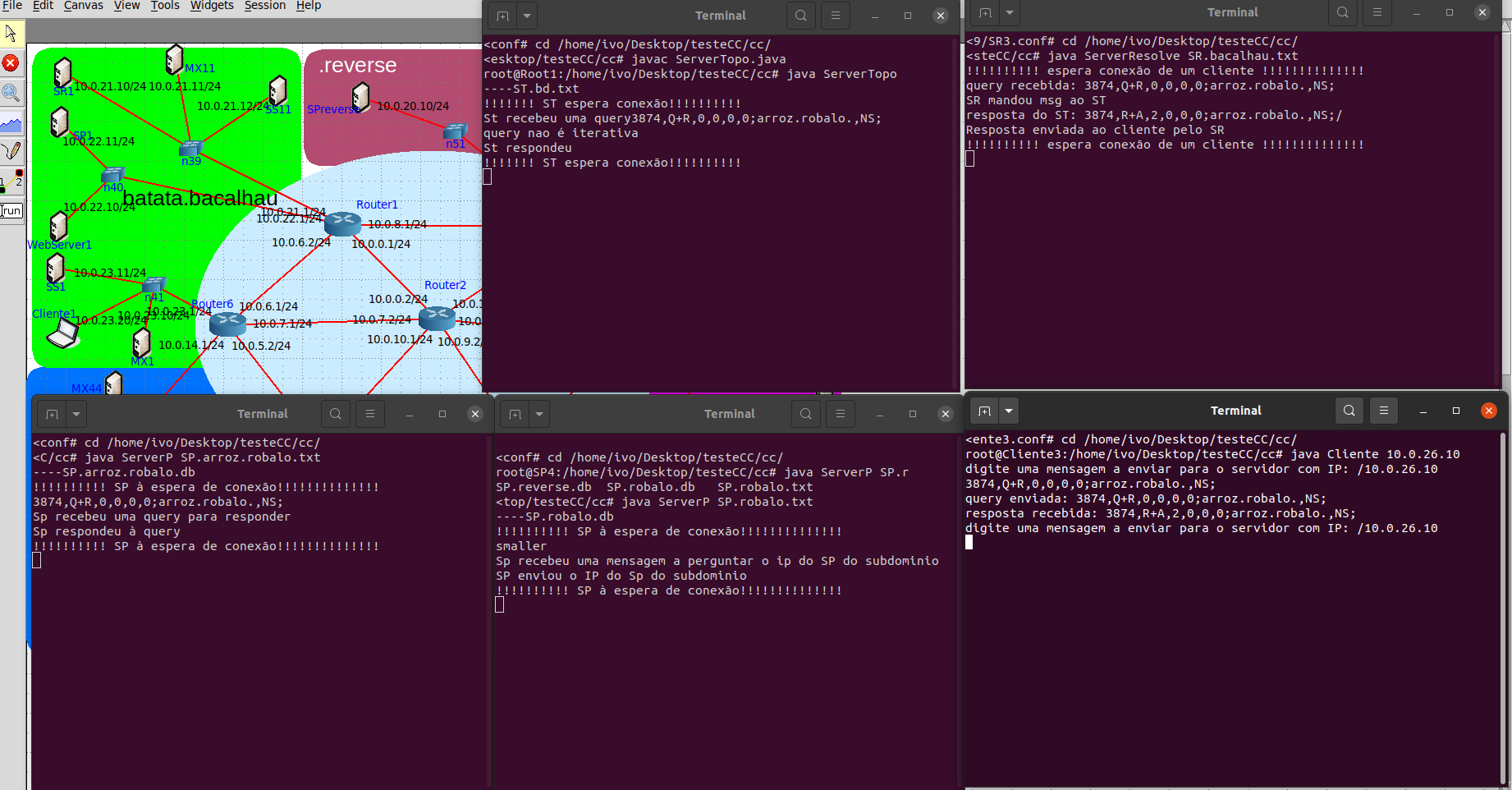


Figura 16 - Print do funcionamento do envio de uma query de um Cl para um SR de um domínio e respondida por um SP de um subdomínio de um outro domínio de maneira recursiva pelo sistema

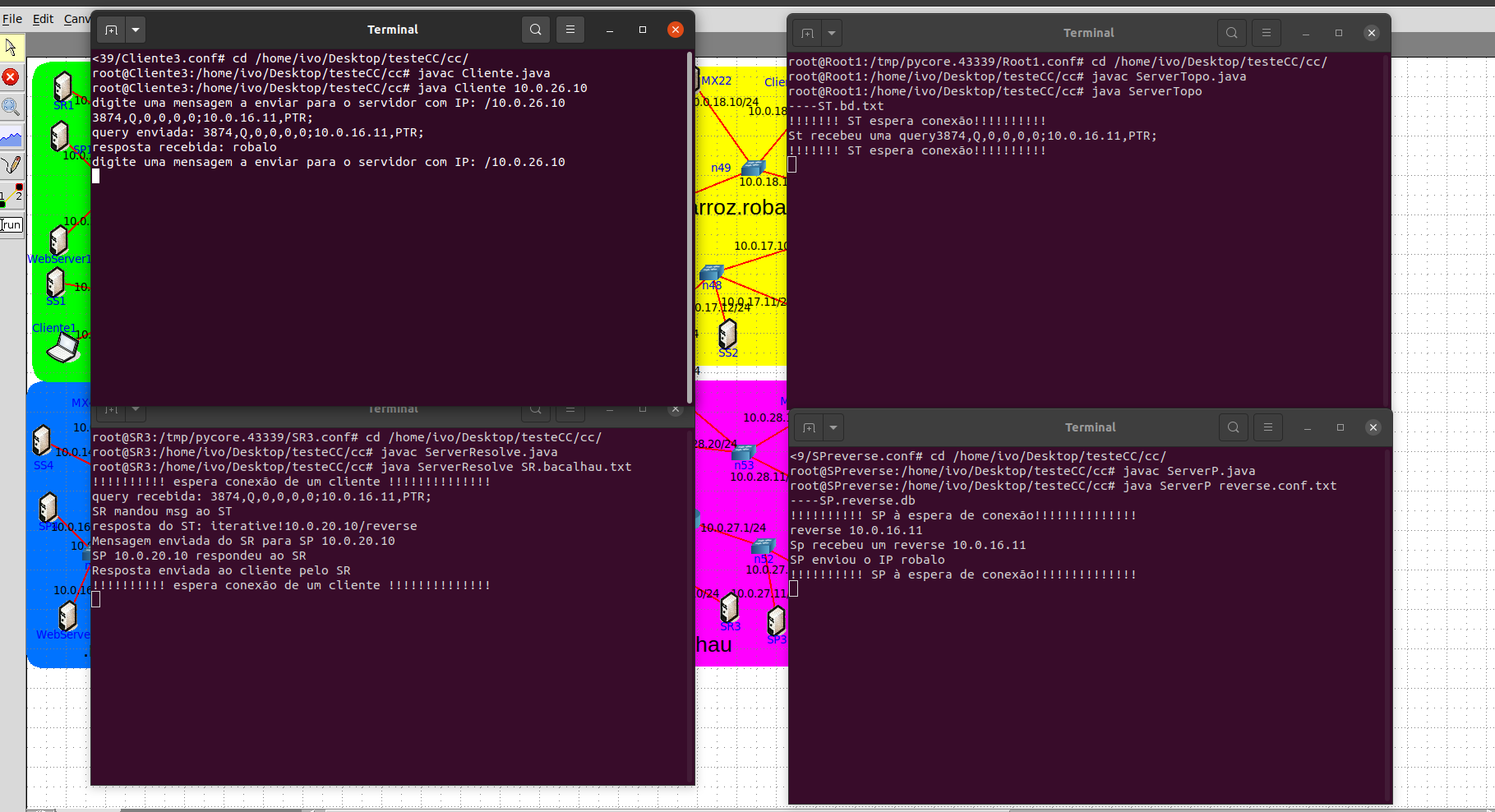


Figura 17 - Print do funcionamento do envio de uma query de um Cl para um SR de um domínio e respondida por um SP domínio reverse de maneira iterativa pelo sistema

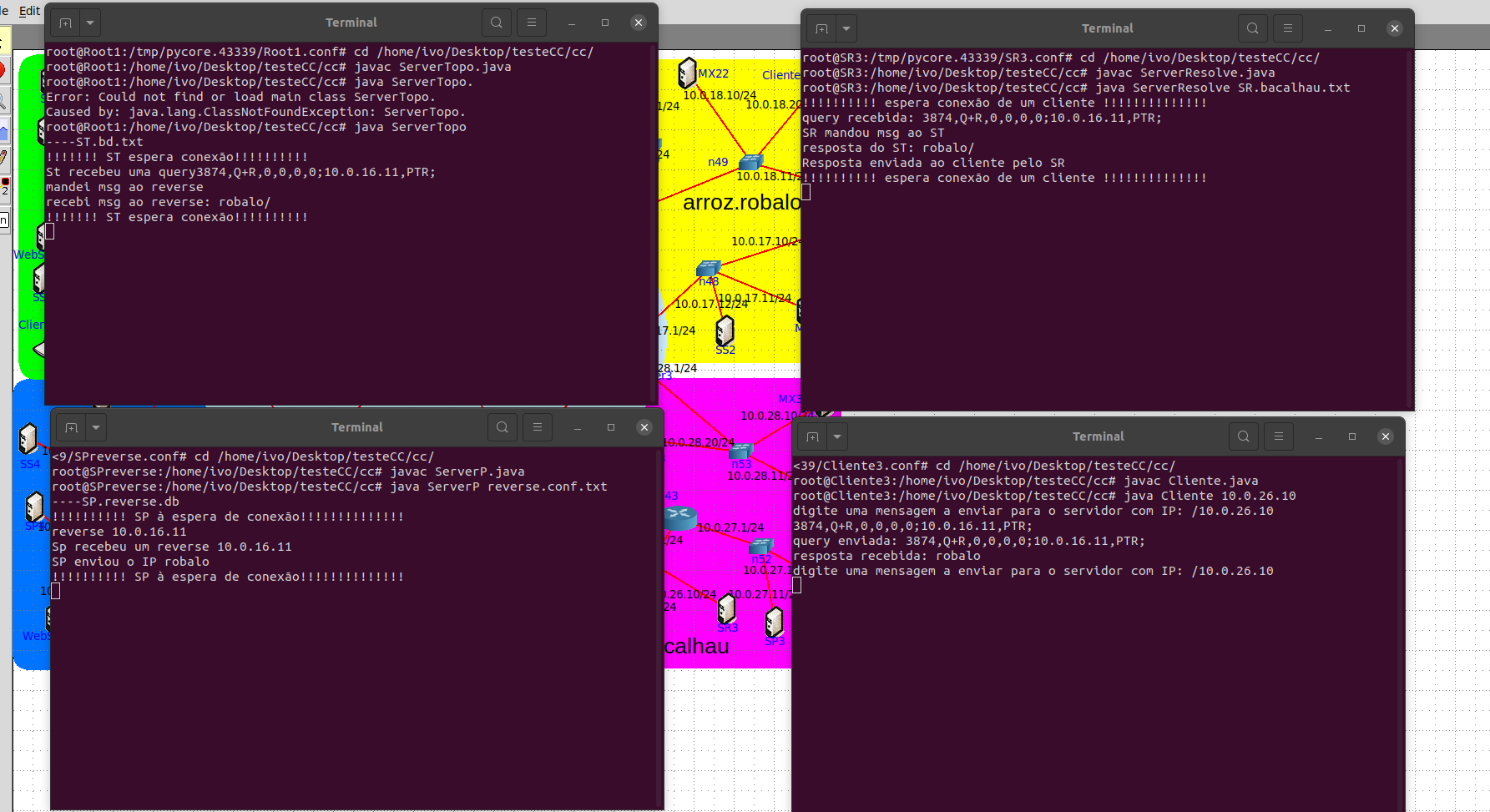


Figura 18 - Print do funcionamento do envio de uma query de um Cl para um SR de um domínio e respondida por um SP domínio reverse de maneira recursiva pelo sistema

Modificações sobre a primeira fase

Quanto a alterações sobre os objetivos a alcançar com a primeira fase do trabalho modificamos as comunicações do cliente com o servidor, que eram antes feitas por TCP para UDP e tanto as mensagens entre cliente e servidor como as de servidor e servidor para serem realizadas em bytes.

Fizemos com o cliente realize mensagens concorrentemente com o SR e qualquer cliente pode aceder a qualquer SR de qualquer domínio;

Dificuldades com que fomos deparados

Enfrentamos algumas dificuldades na realização desta segunda fase, mas decidimos apenas citar as dificuldades que não conseguimos ultrapassar. Primeiramente o domínio .reverse não tem nenhum subdomínio portanto o ficheiro de base de dados do Servidor Primário do domínio .reverse é que tem a informação relativa aos PTRs.

Tabela de empenho dos elementos do Grupo

|  |  |  |
| --- | --- | --- |
| **Tarefas:** | **Elementos do Grupo:** | |
| Ivo | Diogo |
| Modelação da topologia | 35% | 65% |
| Elaboração do modelo de comunicação servidor/cliente | 40% | 60% |
| Elaboração do modelo de comunicação servidor primario/servidor secundario | 60% | 40% |
| Implementaçao da leitura dos fichieros de configuraçao | 60% | 40% |
| Implementaçao da leitura dos fichieros de base de dados | 40% | 60% |
| Implementaçao da comunicação clente/servidor | 60% | 40% |
| Implementaçao da comunicação servidor primario/servidor secundario | 60% | 40% |
| Implementaçao da escrita no ficheiro de logs | 50% | 50% |
| Testes em ambiente de teste e deteção de erros | 65% | 35% |
| Relatorio | 35% | 65% |
| Segunda Parte | | |
| Atualização da Arquitetura do Sistema | 50% | 50% |
| Atualização do Modelo de Informação | 50% | 50% |
| Modelo de Comunicação | 60% | 40% |
| Planeamento do Ambiente de Teste | 50% | 50% |
| Implementação do ST | 40% | 60% |
| Implementação do SR | 40% | 60% |
| Implementação da comunicação iterativa | 60% | 40% |
| Implementação da comunicação recursiva | 60% | 40% |
| Implementação do reverse | 60% | 40% |
| Implementação da comunicação iterativa do reverse | 50% | 50% |
| Implementação da comunicação recursiva do reverse | 50% | 50% |
| Correção de erros | 60% | 40% |
| Relatório | 40% | 60% |

Conclusão

Apesar de não conseguirmos ter entregado tanto a primeira como a segunda e fase a 100% achamos que conseguimos adquirir desde já bastante conhecimento sobre o funcionamento de um servidor de DNS, e de comunicações entre computadores, como por exemplo comunicações entre os servidores e perceber o que esta por detrás do clique que fazemos quando queremos alcançar alguma informação através de um motor de busca.