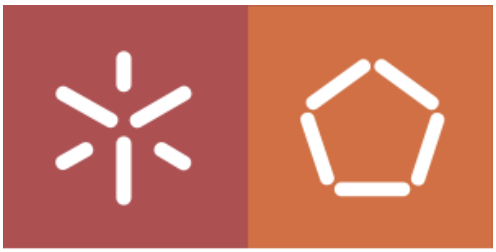

Universidade do Minho
Escola de Engenharia

Comunicações por Computador

Trabalho Prático 2 – 2ª Fase

Grupo 5.07

2022/2023

Autores:

A91671 – João Manuel Novais da Silva

A91697 – Luís Filipe Fernandes Vilas

A91660 – Pedro António Pires Correia Leite Sequeira

Docente:

Bruno Alexandre Fernandes Dias

Braga, 24 de novembro de 2022

Índice

Introdução

Foi-nos proposto, por parte dos docentes da cadeira de Comunicação por Computadores a realização de um trabalho prático que consiste na implementação de um sistema DNS (*Domain Name System*). A partir da matéria lecionada nas aulas e com a ajuda do enunciado providenciado pelos docentes foi possível, aos alunos do grupo G5.07, a formulação e o desenvolvimento deste projeto. Inicialmente será pertinente começarmos por fazer uma breve definição do que intendemos por um sistema DNS e as suas razões para existir.

Podemos constatar que um computador pode ser referenciado, numa rede, por um endereço e este pode ser representado por duas formas, por o seu nome ou pelo seu endereço de IP. Por exemplo, uma determinada máquina pode ser referenciada como **grilo.di.uminho.pt** ou pelo seu endereço de IP **193.137.56.32.** As máquinas que utilizamos necessitam do endereço de IP para localizar e identificar os serviços e dipositivos com que pretende comunicar, mas, claramente os utilizadores não necessitam nem deverão querer memorizar os conjuntos de números que identificam os serviços que pretendem utilizar. Por esta razão o DNS é útil, uma vez que permite a tradução dos endereços por escrito como Facebook.com para endereços IP necessários aos dispositivos e vice-versa.

Na realidade, o serviço DNS é composto por um número incontável de servidores espalhados por todo o mundo, cada um com a informação dos computadores e respetivos endereços de IP.

Mas o mais importante do DNS são os componentes que permitem ao funcionamento desta hierarquia. Nesta hierarquia temos os vários domínios que são representados nos endereços por extenso (grilo.di.uminho.pt, por exemplo), os servidores **Root** que se encontram no topo da hierarquia não são representados nos links que usamos no dia a dia, mas estes, como depois todos os domínios a baixo, apenas possuem informações sobre os domínios diretamente “a baixo” de si. Isto é, os servidores **Root** têm informação sobre, no nosso exemplo, o domínio **.pt**. Este domínio terá informação sobre o domínio **.uminho** e por aí em diante.

Começamos desde cedo em pensar como iríamos resolver os problemas apresentados no enunciado. Em relação ao A.1, reparamos rapidamente que era apenas necessário

descrever as especificações do sistema como estavam apresentadas no enunciado assim como os componentes de software, também seria útil apresentar um esboço de como pensamos que a rede se irá estruturar. Relativamente ao A.2 é pretendido que seja apresentada a especificação da sintaxe e da semântica dos ficheiros de configuração, base de dados e log, como também a descrição do que acontece quando acontece erros de leitura. Para o A.3 é desejado que seja elaborada a especificação completa de todas as interações possíveis entre os elementos definidos no ponto A.1 e também o comportamento deste mesmo em situação de erro. O objetivo A.4 consiste em representar um ambiente de teste que irá ser usado na apresentação a partir da ferramenta core, deverão estar detalhados os IP's e as bases de dados. Os pontos A.5 e A.6 vão ser realizados em código usando o Java para a construção de um serviço DNS de comunicação entre os elementos definidos em A.1 com os respetivos ficheiros válidos feitos no A.2 seguindo as regras de comunicação especificados em A.3.

Tarefas/Objetivos da 2ª Fase

B.1 - Arquitetura do Sistema

Descrição do sistema e especificação completa dos requisitos funcionais esperados para cada elemento do sistema; descrição dos componentes de software a utilizar e os módulos que os compõem.

O sistema DNS que iremos implementar será composto por Servidores de Topo, Servidores de Domínio de Topo, Servidores Primários, Servidores Secundários, Servidores de Resolução e Clientes. Este será representado como uma topologia Core mais tarde, mas devemos começar pela realização das especificações destes elementos.

Servidor Primário (SP):

- Responde e efetua *queries* DNS;
- Tem acesso á base de dados dum domínio DNS, sendo a autoridade que o gere;
- Qualquer atualização necessária á informação de um domínio DNS deve ser feita diretamente na base de dados do SP;
- Tem acesso a informação de configuração específica (domínios para os quais é SP, portas de atendimento, identificação dos ficheiros de base de dados, e do ficheiro log, informação de segurança para acesso às bases de dados, identificação dos SS respetivos e dos SP dos subdomínios, endereços dos servidores de topo, etc.).

Input:

- Ficheiro de Configuração;
- Ficheiro de base de dados por cada domínio gerido;
- Ficheiro com lista dos servidores de topo.

Output:

- Ficheiro de log.

Servidor Secundário (SS):

- Responde e efetua *queries* DNS;
- Tem autorização e autoridade para possuir, e tentar manter atualizada, uma réplica da base de dados do SP;
- Tem acesso á informação de configuração específica (domínios para os quais é SS, portas de atendimento, identificação dos SP dos domínios, identificação do ficheiro log, informação de segurança para acesso ao SP, endereço dos servidores de todo, etc.).
- A informação replicada do SP é armazenada apenas em memória volátil no SS.

Input:

- Ficheiros de configuração;
- Ficheiro com lista de servidores de topo.

Output:

- Ficheiro de log.

Servidor de Resolução (SR):

- Responde e efetua *queries* DNS dobre qualquer domínio;
- Não tem autoridade sobre nenhum domínio, apenas é intermediário entre o cliente e os domínios;
- Pode ser implementado em vários níveis da rede;
- Tem acesso á informação de configuração específica (eventuais domínios por defeito e lista dos servidores DNS que deve contactar, porta de atendimento, identificação do ficheiro log, endereço dos servidores de topo).

Input:

- Ficheiro de configuração;
- Ficheiro com lista de servidores de topo.

Output:

- Ficheiro log.

Cliente (Aplicação Cliente DNS):

- É o processo que precisa da informação da base de dados dum determinado domínio;
- Obtém a informação necessária fazendo *queries* DNS a um SR;
- Tem uma lista de SR numa lista num ficheiro de configuração (endereços IP e portas de atendimento).

Input e Output:

- Linha de comandos.

Servidores Domínio de Topo (SDT):

- Comportamento igual aos SP ou aos SS (ou seja, um SP ou SS autoritários para um domínio de topo é um SDT).

Servidores de Topo (ST):

- Comportamento igual a um SP;
- Têm apenas uma base de dados que associa cada Domínio de Topo com a informação dos respetivos SDT (ou seja, os endereços IP e os nomes dos seus SS do seu SP).

Ambos os ST e os SDT devem ser implementados com o mesmo componente que implementa um SP ou um SS. Os componentes desenvolvidos devem suportar 3 parâmetros de funcionamento que podem ser passados como argumentos na altura do arranque. Estes parâmetros são usados para:

- Indicar a porta de atendimento dos servidores quando esta for diferente da porta normalizada (53);
- Indicar o valor de *timeout* quando se espera por uma resposta a uma *query*;
- Indicar se funciona em modo *debug* ou não.

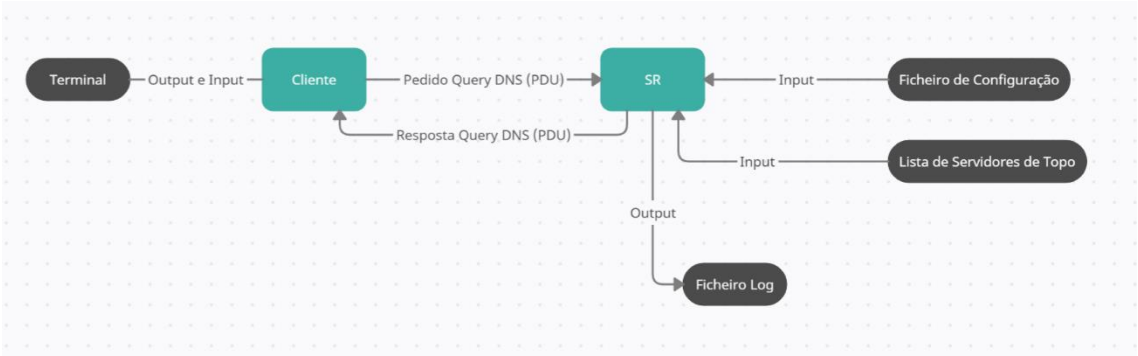

Figura 1 - Primeiro Esboço da Arquitetura dos componentes Cliente e SR

Como definido anteriormente, o SR recebe como input um ficheiro de configuração e uma lista de servidores de topo, e regista num ficheiro de log os acontecimentos. O cliente apenas lê e escreve para o terminal. O cliente comunica com o SR a partir de *queries* DNS como se indica na figura 1.

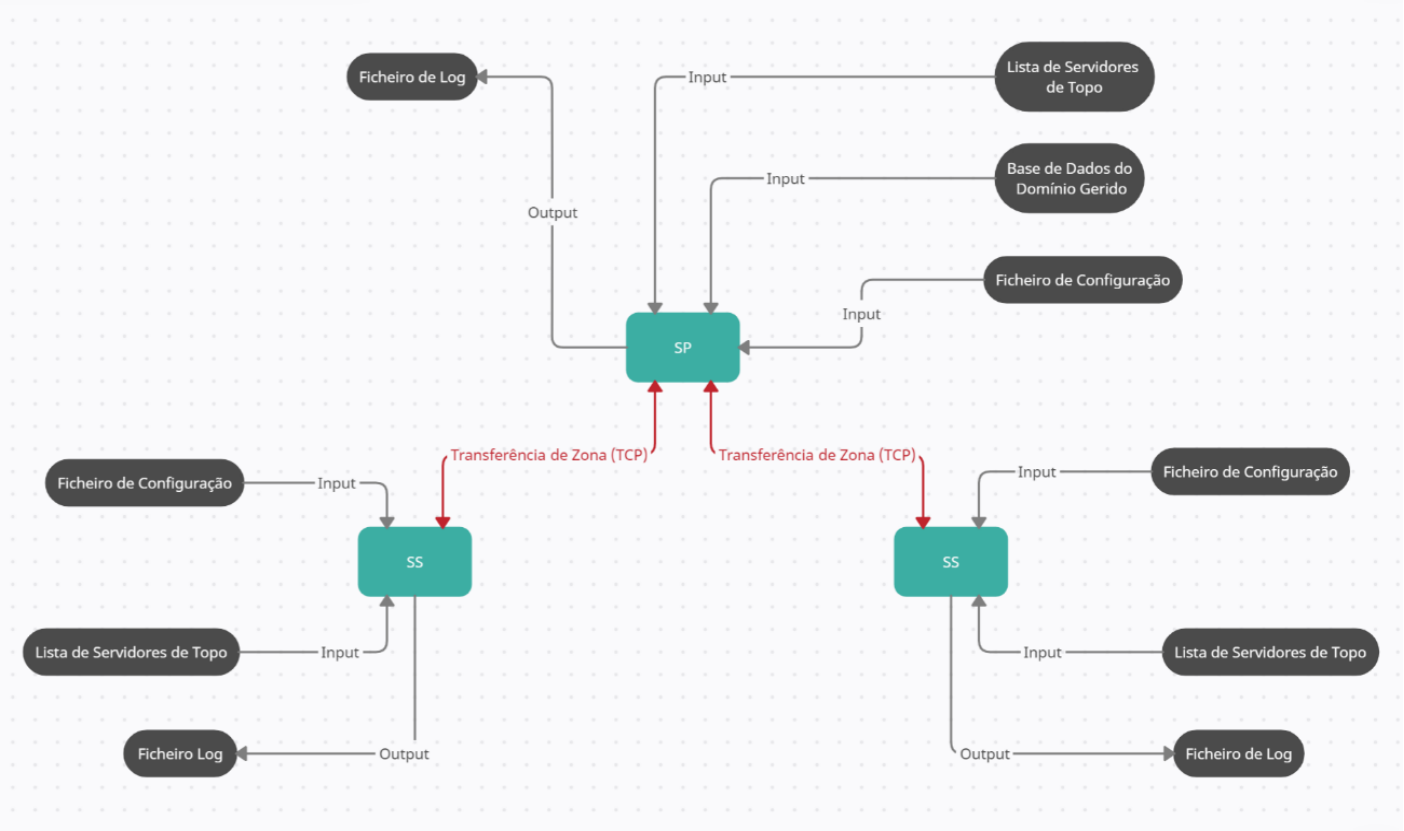

Figura 2 - Primeiro Esboço dos componentes SP e SS

O componente SP recebe, como indicado anteriormente, uma lista de servidores de topo, uma base de dados dos domínios a gerir e um ficheiro de configuração. E como todos os outros componentes (á exceção de alguns) escreve os acontecimentos para um ficheiro de

log. Os SS recebem, como os SP, um ficheiro de configuração e a lista de servidores de topo, mas não a base de dados, esta irá ser fornecida pelo SP, posteriormente, a partir da Transferência de Zona usando o TCP para copia a base de dados o SP para os SS.

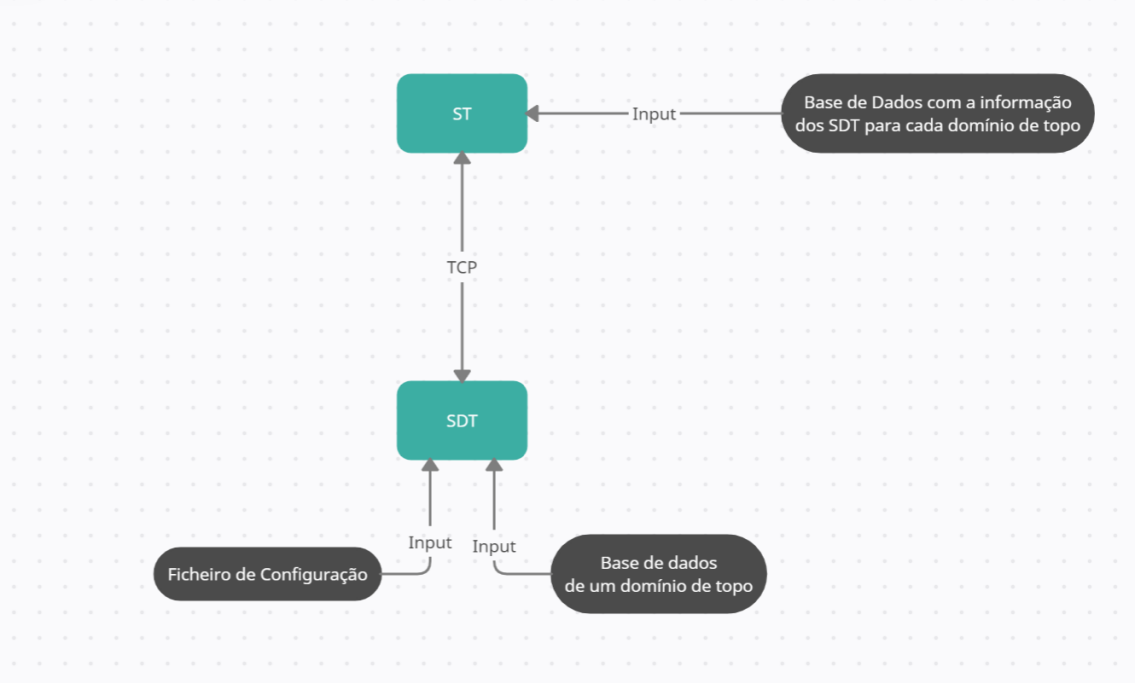

Figura 1 - Primeiro Esboço dos componentes ST e SDT

Neste projeto foi definido no enunciado que os SDT's devem funcionar como um SS ou SP autoritário de um domínio de topo, por isso este servidor recebe um ficheiro de configuração e uma base de dados do seu domínio de topo. Por outro lado, os ST's funcionam como SP e apenas possuem uma base de dados com a informação de cada SDT correspondente a um domínio de topo. Sendo assim, o ST é um servidor que gere todos os domínios de topo e os SDT guardam as informações de cada domínio de topo.

B.2 – Modelo de Informação

Especificação completa da sintaxe e da semântica de todos os ficheiros e do comportamento dos elementos em situação de erro de leitura, do PDU/mensagens DNS e eventual mecanismo de codificação binária.

Foram, previamente, definidos os ficheiros que iriam ser usados neste projeto. Todos estes têm uma sintaxe especifica que deve ser respeitada por cada grupo de trabalho. Estes ficheiros são:

- Ficheiros de configuração – são apenas lidos e processados no arranque do componente de software a que dizem respeito e moldam o seu comportamento;
- Ficheiro de dados – são consultados apenas no arranque e a sua informação deve ser armazenada em memória;
- Ficheiro de log.

Se for necessário atualizar ou alterar o comportamento dos servidores com informação modificada nos ficheiros de configuração ou de dados que lhes dizem respeito a única opção é reiniciar esses servidores.

Ficheiros de Configuração dos servidores SP, SS e SR:

Este ficheiro tem uma sintaxe com as seguintes regras:

- As linhas começadas por '#' são consideradas comentários e são ignoradas;
- As linhas em branco também são ignoradas;
- Deve existir uma definição de parâmetro de configuração por cada linha seguindo a sintaxe:
  {parâmetro} {tipo de valor} {valor associado aos parâmetros}

O componente de implementação deve conseguir detetar erros na sintaxe dos ficheiros de configuração registando a informação desse erro no ficheiro de log.

Tipos de valores aceites (todas as referências a domínios são consideradas nomes completos):

- DB – o valor indica o ficheiro da base de dados com a informação do domínio indicado no parâmetro (o servidor assume o papel de SP para este domínio);
- SP – o valor indica o endereço IP [: porta] dum SS do domínio indicado no parâmetro (o servidor assume o papel de SP para este domínio) e que passa a ter autorização para pedir a transmissão da informação da base de dados (transferência de zona); podem existir várias entradas para o mesmo parâmetro (uma por cada SS do domínio);
- DD – o valor indica o endereço IP [: porta] dum SR, dum SS ou dum SP do domínio por defeito indicado no parâmetro; quando os servidores que assumem o papel de SR usam este parâmetro é para indicar quais os domínios para os quais devem contactar diretamente os servidores indicados se receberem *queries* sobre estes domínios (quando a resposta não está na cache), em vez de contactarem um dos ST; podem existir várias entradas para o mesmo parâmetro (uma por cada servidor do domínio por defeito); quando os servidores que assumem o papel de SP ou SS usam este parâmetro é para indicar os únicos domínios para os quais respondem (quer a resposta esteja em cache ou não), i.e. nestes casos, o parâmetro serve para restringir o funcionamento dos SP ou SS a responderem apenas a *queries* sobre os domínios indicados neste parâmetro;
- ST – o valor indica o ficheiro com a lista dos ST (o parâmetro deve ser igual a "root");
- LG – o valor indica o ficheiro log que o servidor deve utilizar para registar a atividade do servidor associado ao domínio indicado no parâmetro; só podem ser indicados domínios para o qual o servidor é SP ou SS; tem de existir pelo menos uma entrada a referir o ficheiro de log para toda a atividade que não seja diretamente referente aos domínios especificados noutras entradas LG (neste caso o parâmetro deve ser igual a "all").

```
# Configuration file for primary server nº 1 for lei.

lei. DB ./Database/lei.db
lei. DD 127.0.0.1
lei. LG ./var/dns/sp1lei.log
root ST ./Database/rootservers.db
all LG /var/dns/all.log
```

Figura 2 – Exemplo de um ficheiro de configuração de um dos SP's do domínio lei.

Ficheiro com a lista de ST:

Este ficheiro tem a lista de Servidores de Topo que devem ser contactados sempre que necessário (quando se quer saber informação sobre domínios acima do domínio atual e a resposta não está na cache); deve existir um endereço IP [: porta] ST por cada linha do ficheiro.

Ficheiro de log:

Estes ficheiros registam toda a atividade relevante do componente; deve existir uma entrada de log por cada linha do ficheiro; sempre que um componente arranca este deve verificar a existência dos ficheiros de log indicados no seu ficheiro de configuração; se não existir algum deve ser criado e se já existirem a novas entradas devem ser registadas a partir da última entrada já existente no ficheiro; a sintaxe de cada entrada é a seguinte:

{etiqueta temporal} {tipo de entrada} {endereço IP [: porta]} {dados de entrada}

A etiqueta temporal é a data e hora do sistema operativo na hora que aconteceu a atividade registada.

Tipos de entradas aceites:

- QR/QE – foi recebida/enviada uma *query* do/para o endereço indicado; os dados da entrada devem ser os dados relevantes incluídos na *query*; a sintaxe dos dados de entrada é a mesma que é usada no PDU de *query* no modo *debug* de comunicação entre os elementos;
- RP/RR – foi enviada/recebida uma resposta a uma *query* para o/do endereço indicado; os dados da entrada devem ser os dados relevantes incluídos na resposta à *query*; a sintaxe dos dados de entrada é a mesma que é usada no PDU de resposta às *queries* no modo *debug* de comunicação entre os elementos;
- ZT – foi iniciado e concluído corretamente um processo de transferência de zona; o endereço deve indicar o servidor na outra ponta da transferência; os dados da entrada devem indicar qual o papel do servidor local na transferência (SP ou SS) e, opcionalmente, a duração em milissegundos da transferência e o total de bytes transferidos;
- EV – foi detetado um evento/atividade interna no componente; o endereço deve indicar 127.0.0.1 (ou *localhost* ou @); os dados da entrada devem incluir

informação adicional sobre a atividade reportada (por exemplo, ficheiro de configuração/dados/ST lido, criado ficheiro de log, etc.);

- ER – foi recebido um PDU do endereço indicado que não foi possível descodificar corretamente; opcionalmente, os dados da entrada podem ser usados para indicar informação adicional (como, por exemplo, o que foi possível descodificar corretamente e em que parte/byte aconteceu o erro);
- EZ – foi detetado um erro num processo de transferência de zona que não foi concluída corretamente; o endereço deve indicar o servidor na outra ponta da transferência; os dados da entrada devem indicar qual o papel do servidor local na transferência (SP ou SS);
- FL – foi detetado um erro no funcionamento interno do componente; o endereço deve indicar 127.0.0.1; os dados da entrada devem incluir informação adicional sobre a situação de erro (por exemplo, um erro na descodificação ou incoerência dos parâmetros de algum ficheiro de configuração ou de base de dados);
- TO – foi detetado um *timeout* na interação com o servidor no endereço indicado; os dados da entrada devem especificar que tipo de *timeout* ocorreu (resposta a uma *query* ou tentativa de contato com um SP para saber informações sobre a versão da base de dados ou para iniciar uma transferência de zona);
- SP – a execução do componente foi parada; o endereço deve indicar 127.0.0.1; os dados da entrada devem incluir informação adicional sobre a razão da paragem se for possível obtê-la;
- ST – a execução do componente foi iniciada; o endereço deve indicar 127.0.0.1; os dados da entrada devem incluir informação sobre a porta de atendimento, sobre o valor do *timeout* usado (em milissegundos) e sobre o modo de funcionamento (modo “shy” ou modo *debug*).

19:10:2022.11:20:50:020	ST	127.0.0.1	53 2000 shy
19:10:2022.11.20:51:783	EV	@db-file-read	exemplo-com.db

Tabela 1 - Exemplo de Ficheiro Log

Ficheiro de dados do SP

Este ficheiro tem uma sintaxe com as seguintes regras:

- As linhas começadas por "#" são consideradas comentários e, portanto, ignoradas;
- As linhas em branco também são ignoradas;
- Deve existir uma definição de parâmetro de dados por cada linha seguindo esta sintaxe:

{parâmetro} {tipo de valor} {valor} {tempo de validade} {prioridade}

O tempo de validade (TTL) é o tempo máximo em segundos que os dados podem existir na cache dum servidor. Quando o TTL não é suportado num determinado tipo, o seu valor é igual a zero.

O campo da prioridade é valor inteiro menor que 256 e que define uma ordem de prioridade de vários valores associados ao mesmo parâmetro. Quanto menos o valor maior a prioridade. Para parâmetros com um único valor ou para parâmetros em que todos os valores têm a mesma prioridade, o campo não deve existir. Este campo ajuda a implementar um sistema de backup de serviço/hosts quando algum está em baixo (número de prioridade entre 0 e 127) ou balanceamento de carga de serviços/hosts que tenham vários servidores/hosts aplicacionais disponíveis (números de prioridade entre 128 e 255).

Os nomes completos de e-mail, domínios, servidores e hosts devem terminar com um "." (assim: *exemplo.com.*). Quando os nomes não terminam com um "." entende-se que são concatenados com um prefixo por defeito definido através do parâmetro @ do tipo DEFAULT.

Tipo de valores a suportar (os tipos de valores marcados com um * são opcionais):

- DEFAULT* – define um nome (ou um conjunto de um ou mais símbolos) como uma macro que deve ser substituída pelo valor literal associado (não pode conter espaços nem o valor dum qualquer parâmetro DEFAULT); o parâmetro @ é reservado para identificar um prefixo por defeito que é acrescentado sempre que um nome não apareça na forma completa (i.e., terminado com '.'); o valor de TTL deve ser zero;
- SOASP – o valor indica o nome completo do SP do domínio (ou zona) indicado no parâmetro;

- SOAADMIN – o valor indica o endereço de e-mail completo do administrador do domínio (ou zona) indicado no parâmetro; o símbolo '@' deve ser substituído por '.' e '.' no lado esquerdo do '@' devem ser antecedidos de '\';
- SOASERIAL – o valor indica o número de série da base de dados do SP do domínio (ou zona) indicado no parâmetro; sempres que a base de dados é alterada este número deve ser incrementado;
- SOAREFRESH – o valor indica o intervalo temporal em segundos para um SS perguntar ao SP do domínio indicado no parâmetro qual o número de série da base de dados dessa zona;
- SOARETRY – o valor indica o intervalo temporal para um SS voltar a perguntar ao SP do domínio indicado no parâmetro qual o número de série da base de dados dessa zona, após um *timeout*;
- SOAEXPIRE – o valor indica o intervalo temporal para um SS deixar de considerar a sua réplica da base de dados da zona indicada no parâmetro como válida, deixando de responder a perguntas sobre a zona em causa, mesmo que continue a tentar contactar o SP respetivo;
- NS – o valor indica o nome dum servidor que é autoritário para o domínio indicado no parâmetro, ou seja, o nome do SP ou dum dos SS do domínio; este tipo de parâmetro suporta prioridades;
- A – o valor indica o endereço IPv4 dum host/servidor indicado no parâmetro como nome; este tipo de parâmetro suporta prioridades;
- CNAME – o valor indica um nome canónico (ou alias) associado ao nome indicado no parâmetro; um nome canónico não deve apontar para um outro nome canónico nem podem existir outros parâmetros com o mesmo valor do nome canónico;
- MX – o valor indica o nome dum servidor de e-mail para o domínio indicado no parâmetro; este tipo de parâmetro suporta prioridades;
- PTR – o valor indica o nome dum servidor/host que usa o endereço IPv4 indicado no parâmetro; a indicação do IPv4 é feita como nos domínios de DNS reverso (*rDNS*) quando se implementa *reverse-mapping*;

```
# DNS database file for domain lei.

@ DEFAULT lei.
TTL DEFAULT 86400


@ SOASP sp1.lei. TTL
@ SOASP sp2.lei. TTL
@ SOAADMIN admin.lei. TTL
@ SOASERIAL 20222023 TTL
@ SOAREFRESH 14400 TTL
@ SOARETRY 3600 TTL
@ SOAREXPIRE 604800 TTL


@ NS sp1.lei. TTL
@ NS sp2.lei. TTL

vilas.@ NS spvilas.vilas.lei. TTL
joao.@ NS spjoao.joao.lei. TTL
sequeira.@ NS spsequeira.sequeira.lei. TTL



sp1 A 10.3.3.1 TTL
sp2 A 10.2.2.2 TTL


spvilas.vilas A 10.2.2.3 TTL
spjoao.joao A 10.4.4.12 TTL
spsequeira.sequeira 10.2.2.1 TTL


10.3.3.1 PTR sp1 TTL
10.2.2.2 PTR sp2 TTL


sp1lei CNAME sp1 TTL
sp2lei CNAME sp2 TTL
```

Figura 3 - Ficheiro de base de dados do domínio lei.

B.3 – Modelo de Comunicação

As interações assíncronas neste sistema serão efetuadas a partir de mensagens DNS encapsuladas no protocolo UDP. As mensagens de DNS possuem um cabeçalho de tamanho fixo e um segmento de dados que ocupa no máximo 1KByte (figura 6). O cabeçalho é composto pelos: campos message id, flags, response code, number of values, number of authorities, number of extra values. O campo dos dados é dividido em 4 sub-Campos: query info, response values, authorities values e extra values. Para representar a as mensages de DNS vamos usar uma String.

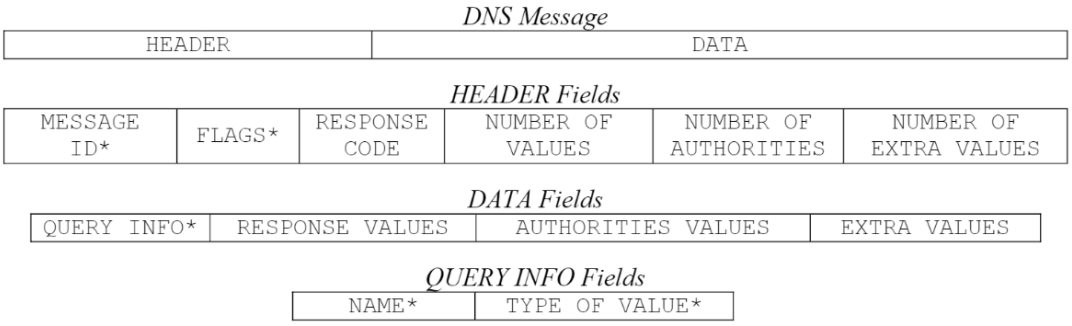

Figura 4 - Representação lógica das mesnsagens DNS

- Message ID: identificador da mensagem gerada pelo cliente ou pelo servidor que faz a *query* original e é um inteiro entre 1 e 65535 e que ocupará 5 bytes do cabeçalho.
- Flags: este campo apresenta 3 valores possíveis que podem estar ativos ou não, Q, R e A. A *flag* Q indica que a mensagem de DNS é uma *query* se estiver ativa, se não estiver ativa então a mensagem é uma resposta a uma *query*. A *flag* R indica se estiver presente numa *query* indica que o processo tem de ser feito de forma recursiva e não iterativa, se estiver ativa numa resposta indica que o servidor que envia a resposta suporta o modo recursivo. A *flag* A indica que a resposta é autoritária. Este campo ocupa 3 bytes da mensagem de DNS
- Response code: este campo possui um inteiro que pode ser 0, 1, 2 ou 3. Se for 0 significa que não ouve erros na resposta e esta possui a informação que responde a *query*, a resposta é então guardada na cache. Se for 1 o domínio apresentado no campo *Name da query info* existe, mas não foi encontrada informação direta relativa ao campo *type values*. Se for 2 então significa que o domínio no campo

main não foi encontrado. Por fim se for 3 significa que a mensagem DNS não foi bem descodificada. Este campo ocupa 1 byte da mensagem.

- Number of values: Número de entradas que respondem diretamente á *query*. Ocupa 1 byte.
- Number of authorities: número de entradas que correspondem aos servidores autoritários do domínio. Ocupa 1byte
- Number of extra values: número de entradas com informação adicional sobre a *query* ou sobre os servidores autoritários. Ocupa 1 byte.
- Querry info: está dividido em 2 campos o *name* (parâmetro da *query*) e *type of values*(tipo de valor associado ao parâmetro).
- Response values: Lista de todas as entradas que respondem diretamente á *query*.
- Authorities values: Lista de todas as entradas que correspondem ao campo *name* e ao tipo de valor NS.
- Extra values: lista de entradas do tipo A que correspondem no parâmetro com todos os valores nos campos *Response values* e *Authorities Values*.

```
# Header
MESSAGE-ID = 3874, FLAGS = Q+R, RESPONSE-CODE = 0,
N-VALUES = 0, N-AUTHORITIES = 0, N-EXTRA-VALUES = 0,;
# Data: Query Info
QUERY-INFO.NAME = example.com., QUERY-INFO.TYPE = MX,;
# Data: List of Response, Authorities and Extra Values
RESPONSE-VALUES = (Null)
AUTHORITIES-VALUES = (Null)
EXTRA-VALUES = (Null)
```

Figura 5 - Query enviada do CL para o SP

```
3874,Q+R,0,0,0,0;example.com.,MX;
```

Figura 6 - Query enviada do CL para o SP (formato conciso)

Além destas definições também será pertinente documentar e analisar todos os tipos de comunicação, entre que componentes é feita a comunicação e o que é comunicado. Para tal podem ser consultadas as figuras 1, 2 e 3 para alguns exemplos.

Daí a seguinte lista:

- Pedido de *query* do cliente para o SR;
- Resposta á *query* do cliente;
- *Query* entre o SR e os servidores dos ST's e vice-versa.

- Transferência de zona entre um SS e um SP.

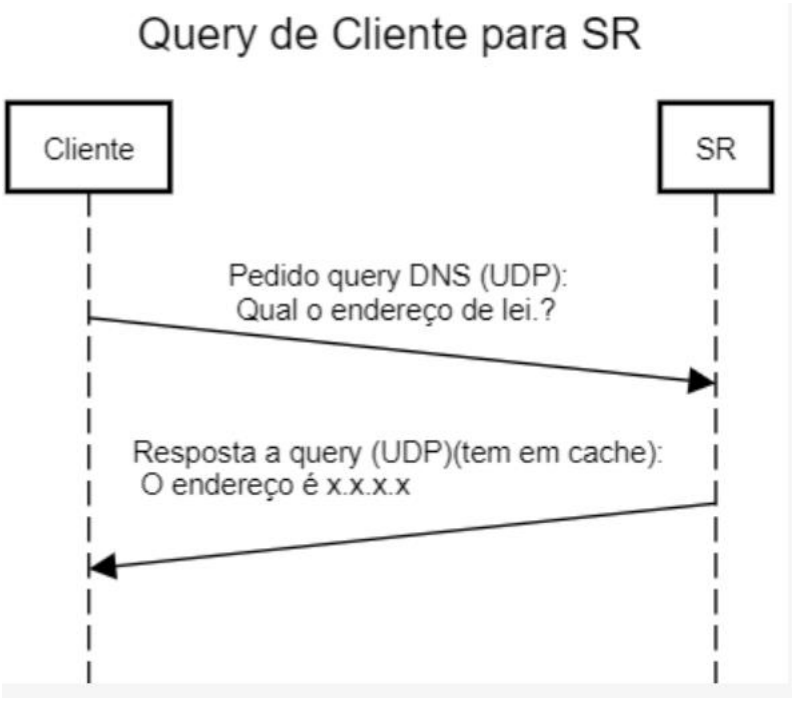

Figura 7 - Diagrama representante do envio de uma query por parte do Cliente e a sua resposta pelo SR

No exemplo da figura 7 é demonstrado um pedido de *query* do Cliente que é, imediatamente, respondido pelo SR uma vez que este já tem a resposta em cache por, possivelmente, já ter realizado a procura de resposta da *query*.

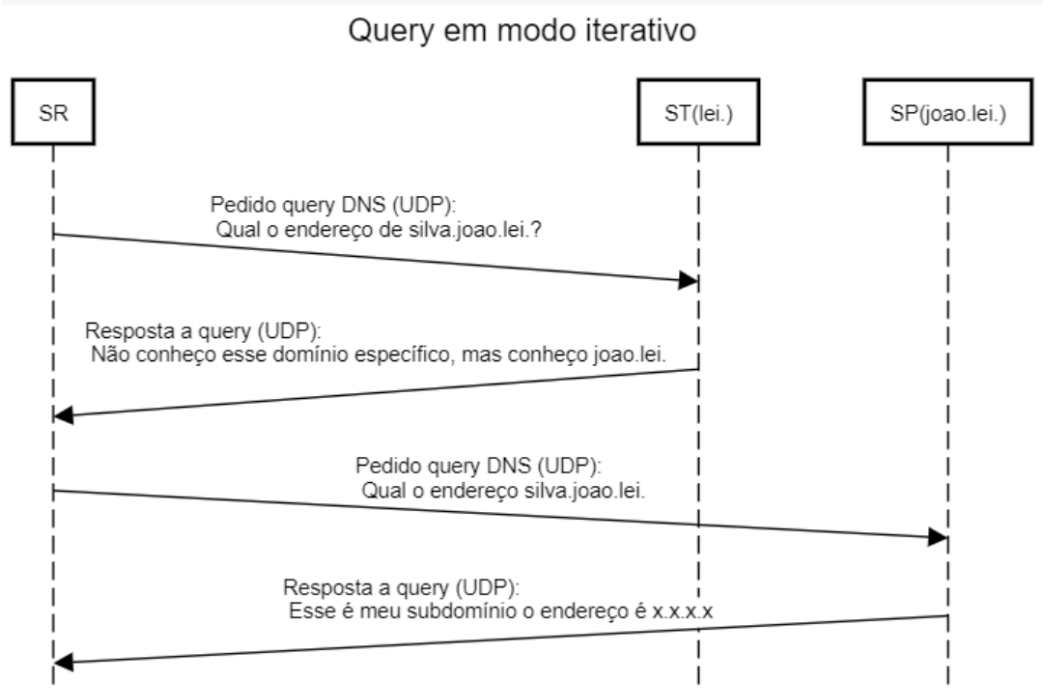

Figura 8 - Diagrama representante da obtenção de reposta à query em modo iterativo em que o SR procura o endereço do domínio joao.lei.

O exemplo da figura 8 mostra o caso onde a resposta não se encontra em cache e, por isso, o SR tem que, em modo iterativo, procurar o endereço começando a perguntar ao servidor de topo e obtendo a resposta da existência do subdomínio deste que é o domínio pretendido. A resposta enviada do SR para o cliente com o endereço está oculta nesta figura, mas é insinuada.

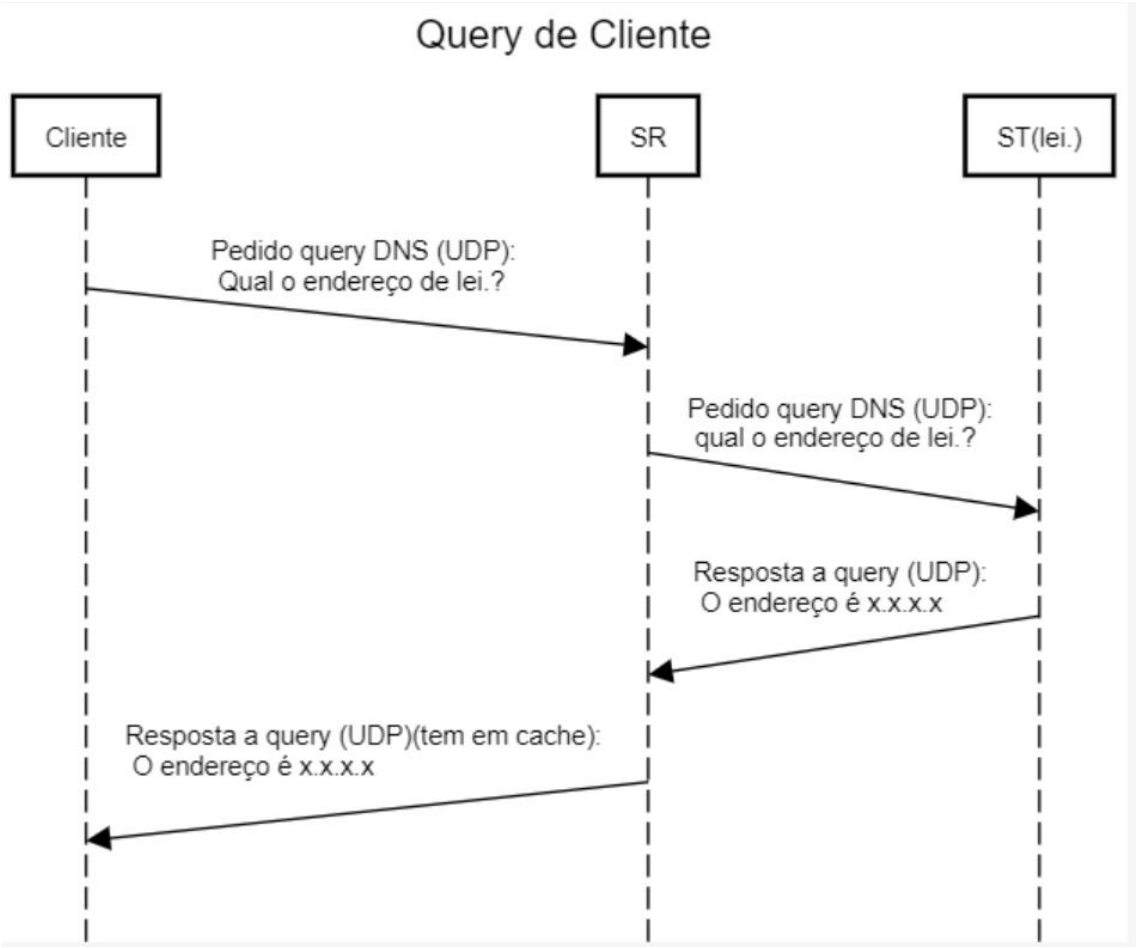

Figura 9 - Diagrama que apresenta todas as mensagens no caso de ser procurado o endereço lei.

Na figura 9 já é incluída a mensagem de resposta ao cliente ocultada na imagem anterior e, portanto, é apresentada uma imagem do fluxo completo do que a comunicação entre os componentes implementados.

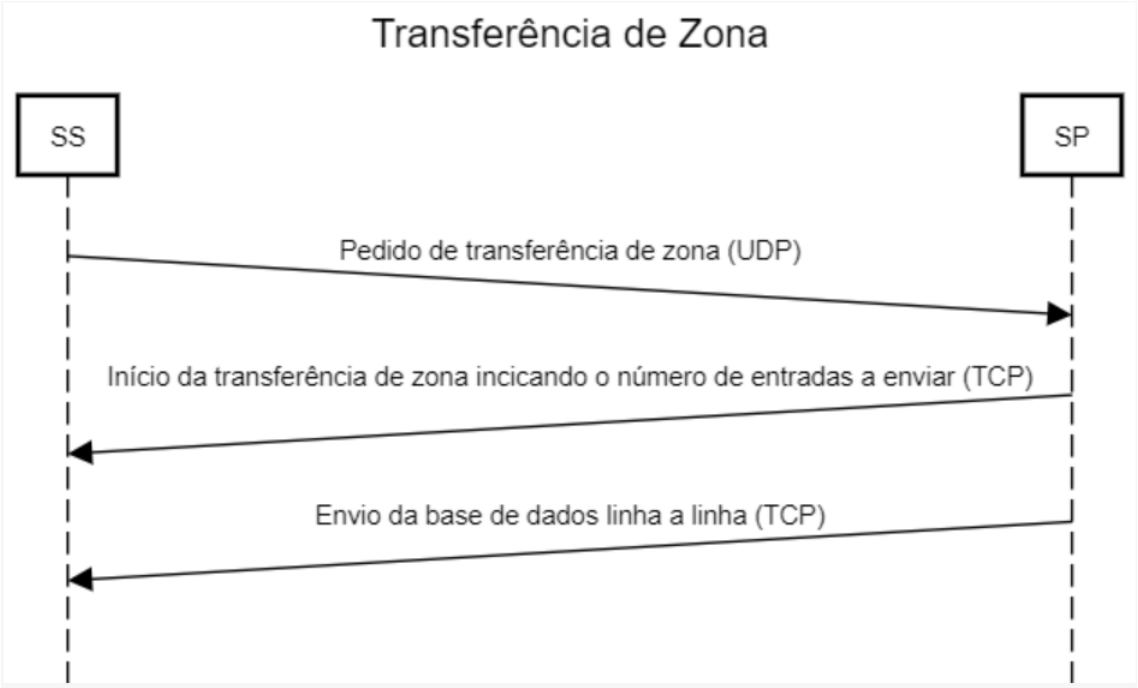

Figura 10 - Diagrama que representa a transferência de zona entre um SS e um SP

Por fim, temos as comunicações que refletem o processo de transferência de zona que, ao contrário de todas as mensagens mencionadas anteriormente, é feito em TCP. Inicialmente a mensagem de pedido para transferência de zona é em UDP, mas depois destes ambos os servidores abrem *sockets* e o SP envia uma mensagem ao SS com número de entradas que serão enviadas, e como o SP envia linha a linha a base de dados, ou seja, o número de linha da base de dados. Após esta "notificação" são enviadas as linhas todas seguidas e é fechada o *socket* por parte do SP e posteriormente o SS também faz o mesmo.

B.4 – Planeamento do Ambiente de Teste

Especificação completa de todas as interações possíveis e do comportamento dos elementos em situação de erro

Para esta fase será necessário construir um ambiente de teste com uma estrutura que represente o que foi definido no enunciado e nos pontos anteriores. Em conjunto com isto também podemos usar o exemplo do core do último trabalho prático:

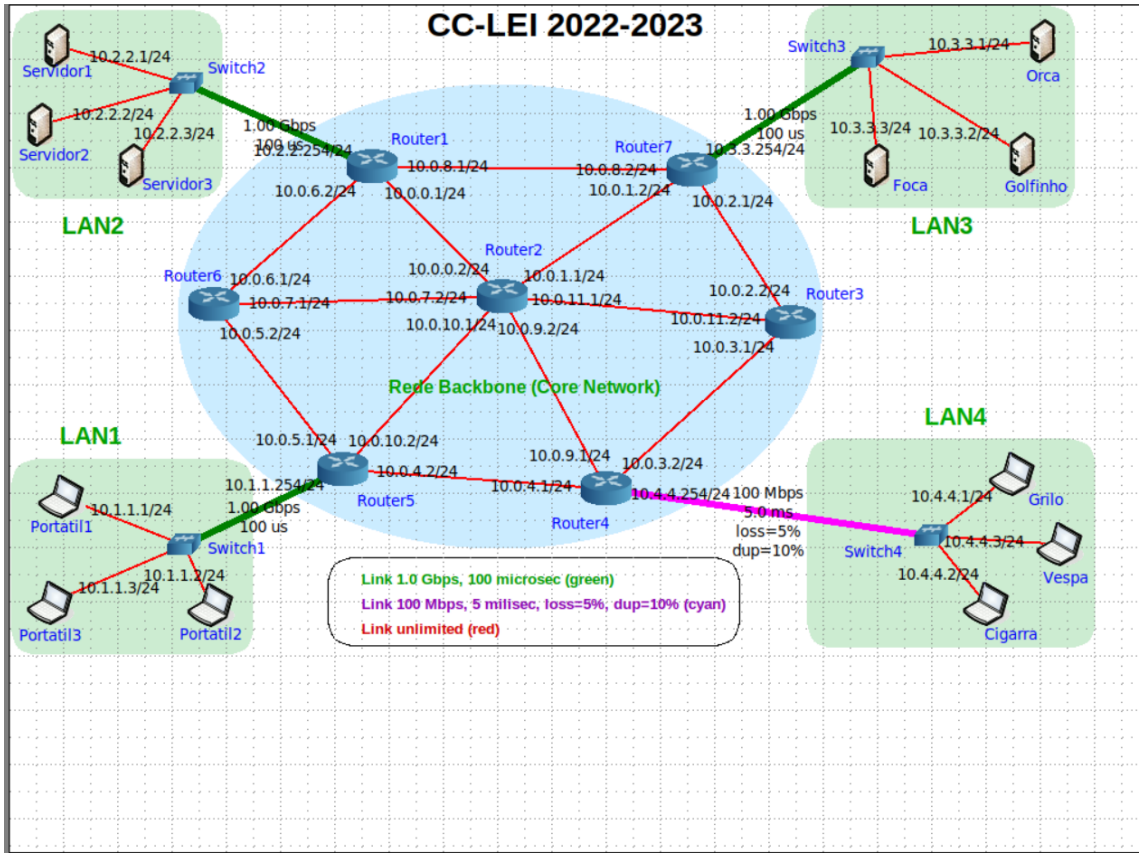

Figura 11 - Ambiente core do TP1

Neste ambiente podemos ver que existem quatro *LAN's*, cujas duas deles possuíam apenas utilizadores e as outras duas apenas *hosts*. O círculo apresentado no meio é a rede *back bone*.

Podemos alterar ligeiramente esta network com as condições que foram pedidas no enunciado, como a presença de dois servidores de topo, utilização de cliente, e também os Servidores primários e secundários de cada domínio.

Mas antes de tudo será relevante representarmos a arquitetura dos domínios criados pelo grupo. O domínio de topo criado chama-se **lei.** e tem 2 SP's, os seus subdomínios são **vilas.lei., joao.lei.** e **sequeira.lei.**. Por sua vez os subdomínios deste são **luis.vilas.lei.**, **silva.joao.lei.** e **pedro.sequeira.lei.** respetivamente (Figura 12).

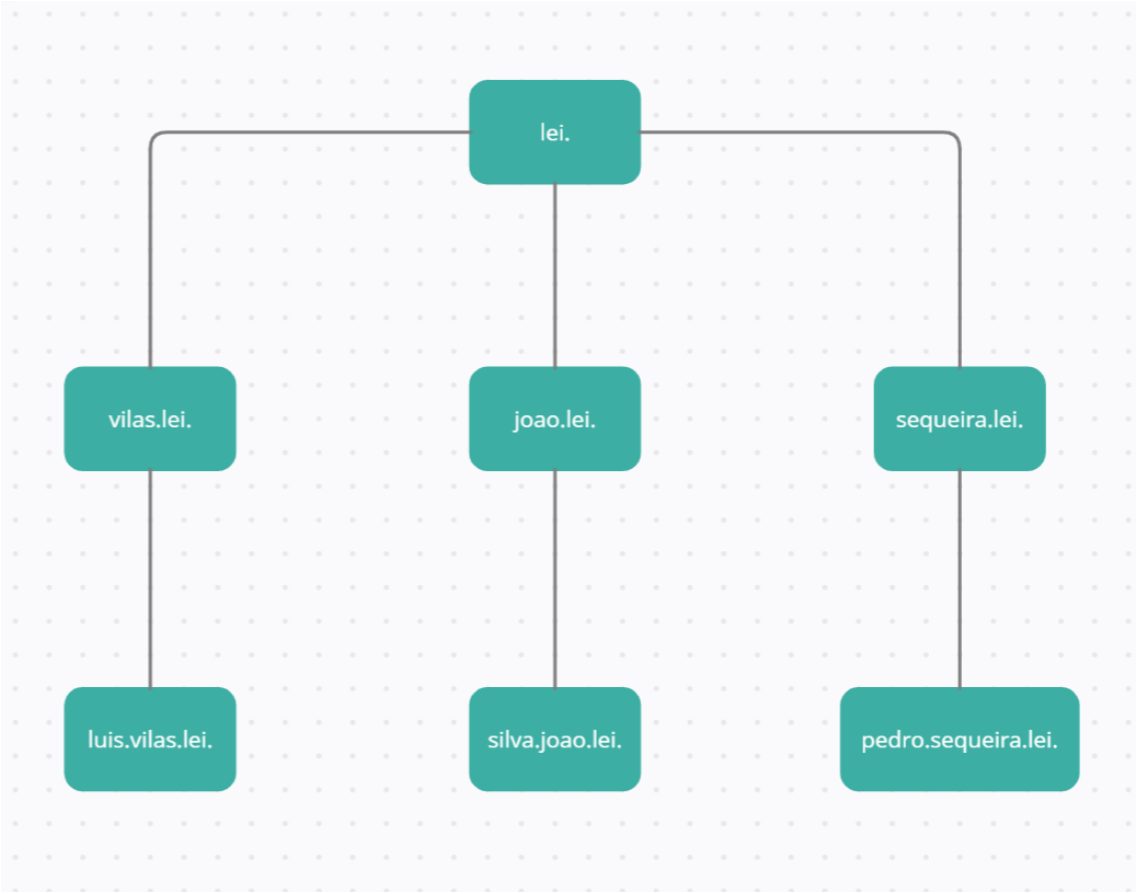

Figura 12 - Arquitetura dos domínios

Além do que já está definido na figura 11 e descrito posteriormente também falta adicionar um conjunto de máquinas pertencentes a cada um dos domínios. Estas máquinas não estarão representadas na topologia core pois não vão ser "executadas", mas estas, nomeadamente, servidores de email, estão declaradas nos ficheiros de base de dados. Também é importante explicar a maneira como foram distribuídos os servidores pela rede. É necessário não ter todos os servidores de um dado domínio na mesma LAN, uma vez que a falha desta resultará na não funcionalidade do domínio e também da perda de dados.

Agora com as descrições definidas anteriormente podemos começar por falar na topologia que foi pensada e criada pelo grupo.

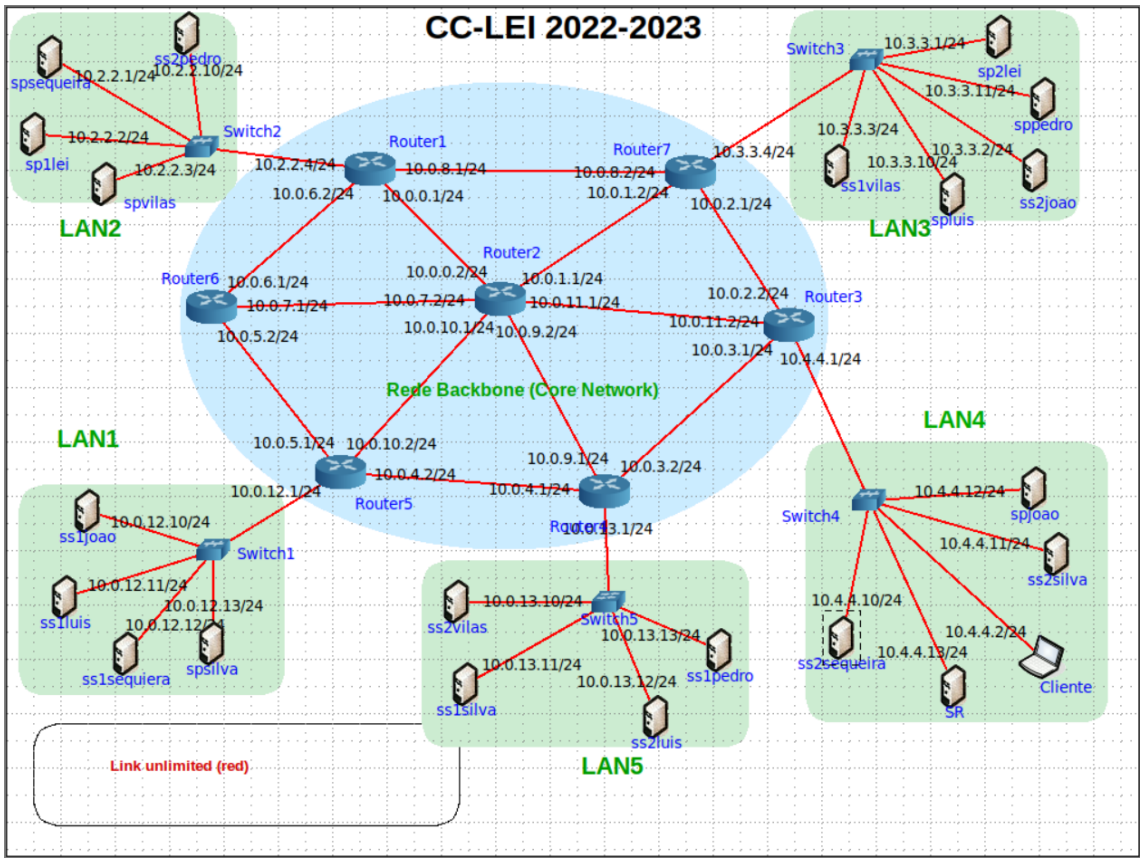

Figura 13 - Topologia core

Na figura 13 apresentasse a topologia construída pelo grupo, como foi já referido esta tem em base a topologia do primeiro trabalho prático com algumas adições, nomeadamente, todos os servidores dos domínios necessários, 1 SP e 2 SS por cada domínio, à exclusão do domínio de topo que tem apenas 2 SP's.

Com a topologia definida podemos agora acabar por construir os ficheiros de configuração e de base de dados dos diversos servidores além do ficheiro de base de dados do domínio lei. representado na figura 3. Seguem-se alguns exemplos destes:

```
# DNS database file for domain vilas.lei.

@ DEFAULT vilas.lei.
TTL DEFAULT 86400


@ SOASP sp.vilas.lei. TTL
@ SOAADMIN admin.vilas.lei. TTL
@ SOASERIAL 20222023 TTL
@ SOAREFRESH 14400 TTL
@ SOARETRY 3600 TTL
@ SOAEXPIRE 604800 TTL


@ NS sp.vilas.lei. TTL
@ NS ss1.vilas.lei. TTL
@ NS ss2.vilas.lei. TTL

luis.@ NS spluis.luis.vilas.lei. TTL


@ MX mx2.vilas.lei. TTL


sp A 10.2.2.3 TTL
ss1 A 10.3.3.3 TTL
ss2 A 10.0.13.10 TTL
mx2 A 10.0.13.16 TTL


spluis.luis. A 10.3.3.10 TTL


10.2.2.3 PTR sp TTL
10.3.3.3 PTR ss1 TTL
10.0.13.10 PTR ss2 TTL
10.0.13.16 PTR mx2 TTL



# CNAMES
spvilas CNAME sp TTL
ss1vilas CNAME ss1 TTL
ss2vilas CNAME ss2 TTL
mail2 CNAME mx2 TTL
```

Figura 14 - Ficheiro de dados do domínio vilas.lei

Neste ficheiro está definido, como mencionado em cima, o servidor de email, o seu endereço com o campo A e com o campo PTR para facilitar ao DNS reverso. Isto também se verifica com os servidores secundários e primário. Além disso também está descrito o endereço do subdomínio do domínio em questão. Os valores para os campos

SOASERIAL, SOAREFRESH, SOARETRY e SOAEXPIRE são iguais para todos os ficheiros.

```
# Configuration file for primary server for vilas.lei.

vilas.lei. DB ./Database/vilas.lei.db
vilas.lei. SS 10.3.3.3
vilas.lei. SS 10.0.13.10
vilas.lei. DD 127.0.0.1
vilas.lei. LG ./var/dns/spvilas.log
all LG /var/dns/all.log
root ST ./var/dns/rootservers.db
```

Figura 15 - Ficheiro de configuração do sp do domínio vilas.lei.

O ficheiro de configuração, neste caso do SP do domínio vilas.lei., possuí as informações necessárias como a localização da base de dados do domínio, o endereço dos SS's, a localização para onde o ficheiro de log deve ser escrito assim como a lista de servidores de topo.

```
# Configuration file for secondary server nº 1 for vilas.lei.

vilas.lei. SP 10.2.2.3
vilas.lei. DD 127.0.0.1
vilas.lei. LG ./var/dns/ss1vilas.lei.log
all LG /var/dns/all.log
root ST /Database/rootservers.db
```

Figura 16 - Ficheiro de configuração do ss1 do domínio vilas.lei.

A figura 16 apresenta a configuração de um SS. Repare que este não possuí um campo DB uma vez que um SS irá guardar a base de dados em cache que irá obter a partir da transferência de zona do SP. Também temos um campo novo, o campo SP que indica o endereço do SP do domínio em questão.

Estes são os ficheiros de configuração e de dados que serão passados aos servidores para teste. Os ficheiros não variam muito do que foi apresentado sendo que as únicas coisas que mudam são os endereços dos servidores e o nome dos ficheiros que logs.

B.5 Implementação de SP, SS, ST, SDT e SR em modo debug e normal

Após todas as análises necessárias, pudemos iniciar a implementação dos vários tipos de servidores.

Desse modo, iniciamos – nos pelo SP, ou seja, o servidor primário que será usado, sempre que esteja disponível. Quando este não está operacional, recorremos a um dos seus SS para podermos fazer os pedidos.

O SP inicia a sua operação, abrindo uma Thread, em TCP que está sempre à escuta na porta 99 para poder enviar a informação para os SS que se vão iniciando. Dessa forma, consegue se garantir que o SS não disponha de uma base de dados própria, pois recebe a informação através do servidor primário.

Por outro lado, quando o SS é iniciado, através do ficheiro de configurações, consegue obter o IP do seu "dono", o servidor principal. Através desse IP faz um pedido de transferência de zona, ou seja, pede para que sejam enviadas as informações que este tem na sua base de dados.

Após recebidas as informações, este inicia o serviço real de DNS, de igual forma na porta 53.

O ST, SR e SDT não foram desenvolvidos por nós neste trabalho, devido à falta de tempo e aos problemas que foram aparecendo no core, bem como incompatibilidades com as outras cadeiras, ou mesmo desencontros nos horários dos membros do grupo.

B.6 CL em modo debug e modo normal

O nosso cliente, desenvolvido de igual forma, em Java, dispõe de opções para efetuarmos os pedidos aos nossos servidores, tanto ao primário como ao secundário.

Com este podemos efetuar pedidos para obter os vários tipos de informações, ao nível do DNS, que um domínio dispõe.

Divisão de Tarefas

	João Silva 91671	Luís Vilas 91697	Pedro Sequeira 91660
A.1	3	1	1
A.2	3	2	1
A.3	1	2	3
A.4	1	1	3
A.5	2	3	2
A.6	2	3	2

Figura 17 - Tabela de divisão de tarefas. Contribuição para a fase 1: 1: Pouca, 2: Média, 3: Grande

	João Silva 91671	Luís Vilas 91697	Pedro Sequeira 91660
B.1	3	1	1
B.2	3	1	1
B.3	3	1	1
B.4	3	1	1
B.5	1	3	1
B.6	1	3	1
B.7	1	3	1
B.8	3	3	1

Figura 18 - Tabela de divisão de tarefas. Contribuição para a fase 2: 1: Pouca, 2: Média, 3: Grande

Conclusão

Em relação á realização do trabalho, afirmamos que as tarefas A1, A2 e A3 foram triviais de realizar uma vez que o enunciado já apresentava a maior parte da informação necessária. Mas apesar disso foram encontras algumas pequenas dificuldades, nomeadamente no comportamento dos elementos em situação de erro de leitura. Também apresentou alguma dificuldade a construção do código que permitia a transferência de zona.

Relativamente o ponto A.4 apresentou algumas dificuldades definição dos ficheiros de base dados, assim como a distinção dos servidores de topo uma vez que são apresentados dois com um intuito de redundância.

Sobre o ponto A.5 e A.6, foram encontrados alguns problemas no entendimento de como funciona do *sockets* quer para TCP quer para UDP, também não tinhas o conhecimento que era obrigatório fazer a transferência de zona pouco após o arranque do sistema.

Felizmente, com o conjunto das aulas práticas e teóricas conseguimos tirar as dúvidas que tínhamos enquanto grupo e, assim, fazer, a nosso ver, um trabalho prático satisfatório.

O grupo sentiu-se interessado em aprender mais e melhor o sistema DNS lecionado nas aulas teóricas. Uma vez que tivemos de entender a maioria dos pormenores sobre este sistema conseguimos perceber melhor o mundo que nos rodeia devido ao use extensivo do DNS.

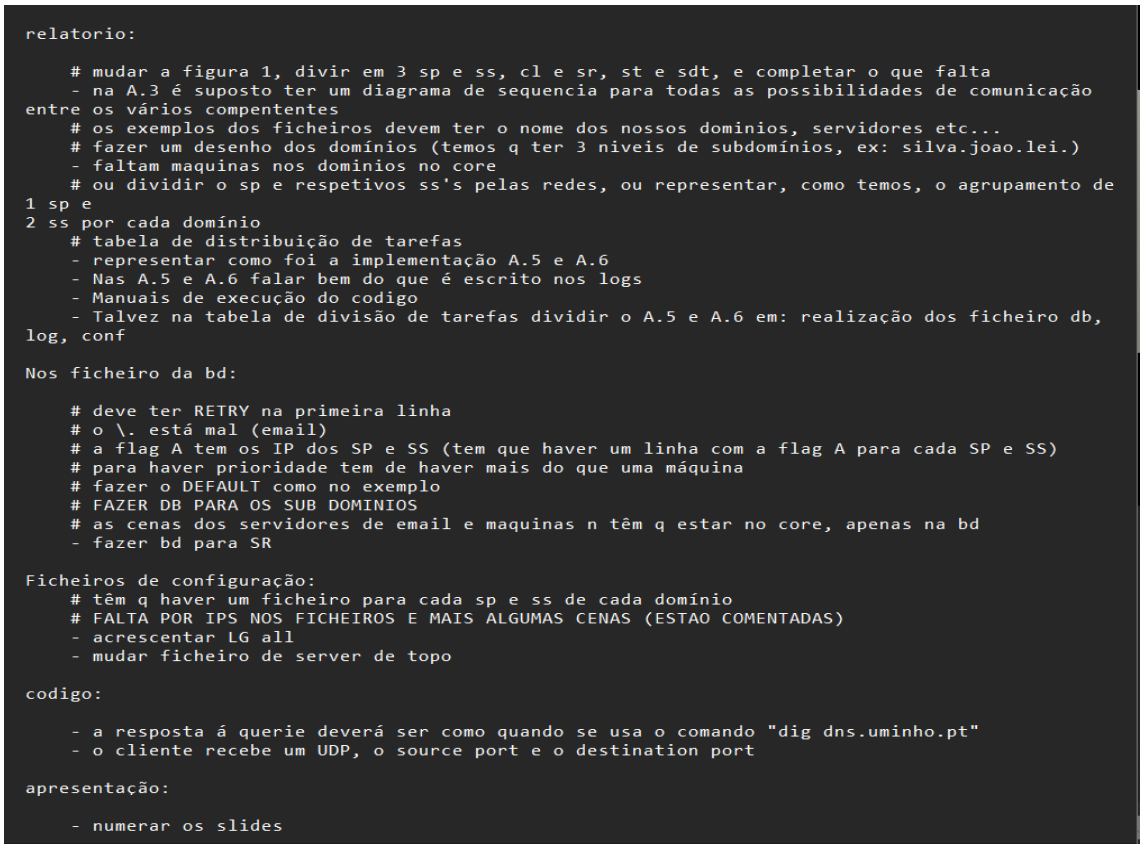

Figura 19 - Tópicos a Rever

Na apresentação da 1ª fase, foram nos descritos alguns problemas que estaríamos a cometer. Através desses comentários, conseguimos corrigir as lacunas presentes no nosso trabalho e melhorar a qualidade do mesmo (Fiigura 19).

Nesta segunda fase, nos tópicos B1-B4 conseguimos completar os objetivos propostos pelo enunciado. Por outro lado, sentimos bastantes dificuldades nos demais tópicos B5-B7, pois eram bastante ambiciosos.

Referências

- **http://www.scom.uminho.pt/Default.aspx?tabid=7&pageid=59&lang=pt-PT**
- **Enunciado do trabalho prático**
- **Slides das aulas práticas**
- **https://creately.com/lp/concept-map-maker/ Para criação dos esboços**
- **Ficheiro core do TP1**
- **https://sequencediagram.org/ Para criação dos diagramas de sequência**
- **Ficheiro de base de dados do domínio lei., vilas.lei., ficheiro de configuração de sp1lei.lei., spvilas.vilas.lei., ss1vilas.vilas.lei.**