



Universidade do Minho
Escola de Engenharia

Comunicações por Computador TP2

Alexandre Fernandes a94154 Délio Alves a94557
Henrique Vaz a95533

24 Novembro 2022

Licenciatura em Engenharia Informática

Conteúdo

1	Introdução	3
2	Planeamento das tarefas da primeira fase	4
2.1	Planeamento do servidor primário	4
2.2	Planeamento do servidor secundário	4
2.3	Planeamento do servidor de resolução	4
2.4	Planeamento do cliente	4
3	Decisões importantes	5
3.1	Escolha da linguagem de programação	5
3.2	Criação dos módulos	5
3.3	Escolha de domínios e sub-domínios	5
4	Arquitetura do sistema	6
5	Funcionalidades de cada elemento	7
5.1	Servidores Principais e Secundários	7
5.2	Servidores DNS (Resolver)	7
5.3	Clientes	7
6	Modelo Comunicativo	8
6.1	PDUs (criação e codificação)	8
6.2	Interações entre os elementos da arquitetura	10
6.3	Erros	10
7	Estratégias de implementação	11
7.1	Servidores primários e secundários	11
7.2	Cliente	11
7.3	Cache	11
7.4	Opções de execução	12
7.5	Atividades reportadas no log	13
8	Planeamento do ambiente de teste	14
8.1	Ficheiros de configuração e base de dados	15
9	Como testar o sistema da fase 1	23
10	Atividades Desenvolvidas e Participação	24

1 Introdução

Este trabalho consiste na implementação de um sistema de DNS (nslookup), o sistema será constituído por: dois servidores de topo, dois servidores de domínios de topo, um domínio de DNS reverso, cada servidor de domínio de topo terá por sua vez dois servidores secundários. Existirão também dois sub-dominios que terão um servidor primário e dois secundários cada. Será também necessário implementar um servidor de resolução (resolver) e um cliente de onde serão enviadas as queries DNS.

Os servidores de topo, servidores de domínios de topo e servidores principais têm basicamente o mesmo funcionamento, logo a diferença entre eles será a forma como serão configurados através dos respetivos ficheiros de configuração. Nestes servidores os dados dos domínios serão carregados através de ficheiros de base de dados, ou seja, os servidores têm sempre a cópia mais atual dos dados. No caso dos servidores secundários a maior diferença será que o carregamento dos dados será efetuado a partir de um servidor primário, ou seja, o servidor secundário irá pedir ao primário os dados de um dado domínio e, caso haja autorização para tal, o servidor primário envia os dados (transferência de zona). No caso do servidor de resolução (resolver) não haverá nem carregamento de dados por ficheiro nem por transferência de zona. Em todos estes servidores existirá um sistema de caching positivo, isto é, sempre que houverem queries para dados que não tínhamos guardados eles serão então armazenados em cache para uso futuro.

No caso do cliente não existe cache, o cliente apenas fará perguntas e esperará por respostas (afirmativas ou negativas).

Nesta primeira fase do trabalho serão implementadas as partes mais importantes de todo o sistema. Nesta fase o nosso sistema consiste apenas num CL (que pode enviar queries e receber resultados), num SS que pode responder a queries do CL e também pedir informações ao SP (transferência de zona) e um SP (que pode receber queries e responder a pedidos dos SS).

2 Planeamento das tarefas da primeira fase

2.1 Planeamento do servidor primário

Para o servidor primário a ideia do grupo será criar um programa capaz de receber alguns parametros para a configuração, por exemplo, qual a porta por defeito a usar, escolher entre modo shy e modo debug, escolher o valor para o timeout, escolher o path para o ficheiro de configuração e para o ficheiro de log.

O SP terá também de ser capaz de fazer parse aos ficheiros de configuração e de base de dados, escrevendo todos os eventos no seu log. Terá também de ser capaz de "escutar" tanto pacotes TCP para transferência de zona como datagramas UDP para responder a queries.

2.2 Planeamento do servidor secundário

Para o servidor secundário será possível receber os mesmo parametros do que nos servidores primários.

No entanto, não será necessário termos um socket TCP a correr continuamente, visto que, este só será utilizado quando quisermos fazer transferência de zona. Tal como no caso anterior será necessário escutar constantemente no socket UDP pois a capacidade de resposta a queries será igual à dos SP.

Tal como os servidores primários, também os SS terão de ser capazes de realizar o parse dos ficheiros de configuração e proceder ao eventual registo de eventos no seu ficheiro de log.

Como é suposto que um SP possa ser também SS de um outro domínio então o código para os SP e SS será exatamente o mesmo, a diferença no seu funcionamento virá do ficheiro de configuração.

2.3 Planeamento do servidor de resolução

Para o resolver o funcionamento será ainda mais simples, não irá carregar dados de ficheiros nem pedir transferência de zona, apenas vai guardando na sua cache os dados das queries que passam por ele.

Como este tipo de servidor também é parecido com os SP e SS o grupo tentará fazer com que o mesmo código possa funcionar para qualquer tipo de servidor de acordo com o ficheiro de configuração.

2.4 Planeamento do cliente

Para o cliente apenas será necessário implementar o mecanismo de envio e receção queries, os parametros que o programa receberá serão apenas o endereço IP[:porta] do servidor que quer contactar, qual o domínio que pretende procurar, qual o tipo do endereço (MX ou A) e se é modo interativo (I) ou recursivo (R) (nesta fase apenas existirá o modo interativo).

3 Decisões importantes

3.1 Escolha da linguagem de programação

Embora houvessem linguagens que proporcionassem uma maior abstração geral do que o C também iríamos perder algum do controlo absoluto sobre a gestão de memória, o que na nossa opinião iria provocar uma perda de eficiência, o C é também a linguagem em que todos os elementos do nosso grupo têm mais experiência, por isso a escolha foi óbvia.

3.2 Criação dos módulos

Os componentes deste trabalho podem ser divididos em 2 tipos principais: Servidores e Clientes. Foi então criado um módulo para cada um destes componentes, de forma a tornar mais fácil a distinção entre si. Apesar de todos os tipos de servidores disponíveis neste projeto apresentarem características um pouco distintas uns dos outros, a sua base acaba por ser, fundamentalmente, a mesma, não se justificando, por isso, a criação de módulos individuais para cada um deles. Uma das vantagens deste tipo de design adotado pelo nosso grupo de trabalho é não só a reutilização de muito do código comum a este tipo de componentes, assim como a redução da quantidade de informação duplicada (em termos de código) entre os diferentes módulos.

Como todos os servidores irão possuir um sistema de caching com a mesma implementação, decidimos então criar um módulo dedicado à construção e implementação do sistema de Cache. Foi também criado um módulo com funções utilitárias, que serão usadas onde necessário. Com todos estes ficheiros foi necessário criar uma Makefile, a qual é responsável por gerar os 2 executáveis distintos para a execução do nosso programa: "dnsServer", "dnsCL".

3.3 Escolha de domínios e sub-domínios

Para domínios de topo escolhemos: animalia. e plantae. Para sub-domínios escolhemos: cao.animalia. e embryophyta.plantae.

4 Arquitetura do sistema

Na primeira fase a arquitetura do sistema será a da figura 1.

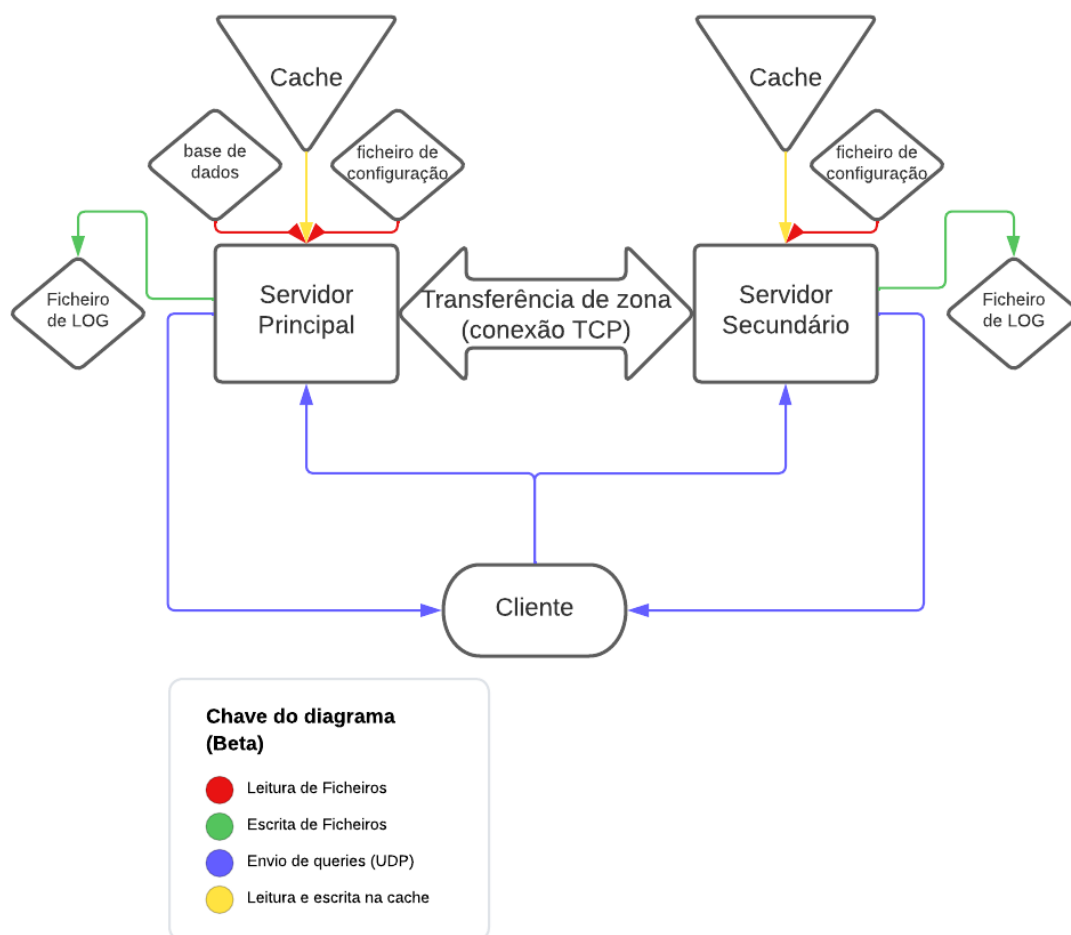


Figura 1: Arquitetura do sistema

5 Funcionalidades de cada elemento

5.1 Servidores Principais e Secundários

Antes que um servidor primário/secundário possa iniciar a sua execução tem primeiro que processar um conjunto de dados de modo a adotar o comportamento desejado. Para isso, estes tipos de servidor possuem um mecanismo que lhes permite analisar e armazenar informação contida não só em ficheiros de configuração como também de base de dados, sendo que estes últimos irão contribuir para a inicialização da cache do servidor em questão.

Após este processo inicial de configuração são então criados dois sockets (um TCP e outro UDP) de modo a ser possível a receção/envio de queries DNS (tanto num contexto de Cliente - Servidor, como num contexto de Servidor - Servidor) e também realizar operações de transferência de zona. De modo a que um servidor deste tipo seja capaz de responder a pedidos desta natureza foi necessário permitir a realização de uma procura de dados no sistema de caching, o qual vai sendo alterado ao longo do tempo em função dos pedidos recebidos/efetuados.

5.2 Servidores DNS (Resolver)

Ao contrário do que acontece nos outros tipos de servidores do projeto, os servidores DNS (ou Resolvers) não necessitam de um ficheiro de base de dados, visto que estes servem apenas como intermediário na resposta/efetuação de queries DNS relativas a um dado domínio em que estes não têm autoridade. Tendo em conta esta informação, todas as funcionalidades referentes à análise e processamento de dados deste tipo de ficheiros não será então necessária para o bom funcionamento destes servidores.

No entanto, com exceção das ligações TCP que possibilitam o processo de transferência de zona, todas as funcionalidades apresentadas na secção anterior estão também presentes nos Servidores DNS. É também importante referir que, como não será efetuada a leitura de um ficheiro de base de dados neste tipo de servidores, a sua cache vai então encontrar-se vazia no momento em que o Resolver inicia o seu processo de execução (a cache vai depois sendo preenchida com informação/dados referentes às diferentes queries recebidas durante toda o tempo de vida do programa).

5.3 Clientes

No que diz respeito aos clientes, estes são capazes de receber e verificar uma query fornecida através da linha de comandos, sendo depois esta enviada, através de um datagrama UDP, para o servidor indicado no pedido e, posteriormente, aguarda a resposta para essa mesma query.

6 Modelo Comunicativo

Tal como consta do enunciado do projeto, a comunicação entre os diversos componentes do sistema irá ser efetuada através do envio de queries DNS, as quais seguem um formato bem estruturado para que exista uma maior facilidade na leitura e retenção da informação pretendida.

6.1 PDUs (criação e codificação)

No momento em que nos preparamos para o envio de uma dada query, procedemos à criação da mensagem a ser transmitida, a qual segue o seguinte formato:

- message id → identificador de mensagem que irá ser usado para relacionar as respostas recebidas com a query original
- flags → suporta as flags *Q* (indica que é uma query e não uma resposta), *A* (indica que a resposta é autoritativa) e *R* (indica que o modo a utilizar é o recursivo, que difere do modo por defeito, que é o iterativo / se estiver presente na resposta de uma query indica que o servidor suporta o modo recursivo)
- response code → indica o código de erro na resposta a uma query (se for 0 significa que não ocorreu nenhum tipo de erro)
- n values → número de entradas relevantes que respondem diretamente à query e que fazem parte da lista de entradas incluídas no campo *RESPONSE VALUES*
- n authorities → número de entradas que identificam os servidores autoritativos para o domínio incluído no *RESULT VALUES*
- n extra values → número de entradas com informação adicional relacionada com os resultados da query ou com os servidores da lista de autoridades
- info name → parâmetro da query
- info type of name → tipo de valor associado ao parâmetro da query
- response values → lista de entradas que fazem match com o *NAME* e *TYPE OF VALUE* incluídos na cache ou na base de dados do servidor autoritativo
- 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
- extra values → lista das entradas do tipo A (incluídos na cache ou na base de dados do servidor autoritativo) que fazem match no parâmetro com todos os valores no campo *RESPONSE VALUES* e no campo *AUTHORITIES VALUES* de forma a que o elemento CL ou servidor recebe resposta não tenha que fazer novas queries para saber os endereços IP dos parâmetros que vêm como valores nos outros dois campos

De modo a respeitar esta estrutura o grupo de trabalho resolveu utilizar o seguinte método de codificação:

```
typedef struct query{
    int message_id;
    char *flags;
    int response_code;
    int n_values;
    int n_authorities;
    int n_extra_values;
    char* info_name;
    char* info_type_of_name;
    char** response_values;
    char** authorities_values;
    char** extra_values;
}*QUERY;
```

Figura 2: PDU de uma QUERY

De seguida apresentamos também um exemplo da criação desta estrutura na interface do cliente:

```
QUERY n_query=malloc(sizeof (struct query));

if(argc!=5){
    printf("Numero de argumentos invalido\n");
    usage();
    exit(1);
}

if (!IP_parser(argv[1],dns_server_ip, &dns_server_port, DEFAULT_PORT)){
    printf("Numero IP do servidor DNS invalido\n"); //ss\n", argv[1]);
}
// iniciar geracao de numeros aleatorios
srand(time(NULL));

// socket UDP
udpfd = socket(AF_INET, SOCK_DGRAM, 0);
if(udpfd<0){
    printf("Unable to create UDP socket");
    exit(1);
}

// minha informacao
bzero(&servaddr, sizeof(servaddr)); //inicializar a estrutura a zeros
servaddr.sin_family = AF_INET; // IP
servaddr.sin_addr.s_addr = htonl(INADDR_ANY); // aceita conexoes de todos
servaddr.sin_port = htons(0); // porta aleatoria para mim (primeira porta)

// binding server addr structure to udp sockfd
bind(udpfd, (struct sockaddr*)&servaddr, sizeof(servaddr));

len = sizeof(servaddr);

// informacao do servidor
inet_aton(dns_server_ip,&(servaddr.sin_addr));
servaddr.sin_port = htons(dns_server_port); // porta do servidor para onde

// constroi query a partir dos argumentos de entrada
build_query(n_query,argv);
```

Figura 3: Exemplo da criação de uma QUERY

6.2 Interações entre os elementos da arquitetura

Existem diversas interações possíveis dentro do nosso sistema DNS, pelo que, nesta secção do documento, iremos proceder à sua descrição detalhada.

Primeiramente, a interação principal (pelo menos nesta primeira fase) do nosso programa ocorre no envio de queries DNS entre um Cliente e um Servidor (quer este seja Primário/Secundário ou um Resolver numa segunda fase do projeto). Para que esta interação seja possível, são criadas (em ambos os componentes) um socket UDP no qual será transmitido os diversos pedidos/respostas entre os intervenientes da ligação. Este tipo de interação está também presente no envio de queries entre os servidores dos diferentes tipos.

No entanto, servidores primários e secundários possuem um tipo de interação somente a eles reservado, sendo ela a transferência de zona. A transferência de zona, ao contrário do que acontece com o envio de queries, faz uso de uma conexão TCP entre um SP e um SS e possui um conjunto diverso de etapas, sendo elas o envio do número de linhas relevantes (não inclui linhas em branco ou comentários) no ficheiro de base de dados do servidor primário, que é depois seguido pelo envio do mesmo valor numérico por parte do SS (de modo a confirmar a validade da informação enviada pelo SP). Após este processo inicial de validação e confirmação começa-se então o envio de cada uma das linhas presentes no ficheiro da base de dados do SP, que serão depois recebidas e armazenadas pelo SS no outro lado da comunicação.

6.3 Erros

Tal como acontece em muitas das etapas deste projeto, também aqui no Modelo Comunicativo estamos sujeitos à ocorrência de erros e/ou situações inesperadas.

Tendo isto em mente, foram então colocadas em funcionamento operações que visam detetar/relatar os casos em que estes supostos erros possam acontecer. Alguns dos erros mais propensos a ocorrer serão a formação/criação de queries cujo PDU não siga as normas extipuladas no enunciado e também os casos em que existe a tentativa da criação de uma ligação com um componente e/ou serviço que não se encontra disponível (nos casos em que um dado servidor se encontra desligado) ou até mesmo situações onde são introduzidos endereços/portas inválidas .

7 Estratégias de implementação

Como já foi referido anteriormente nesta fase foram implementados os servidores principais, secundários e o cliente.

7.1 Servidores primários e secundários

A implementação tanto dos servidores primários como dos secundários está junta no mesmo código, ao lermos o ficheiro de configuração sempre que tivermos uma entrada DB o servidor irá ler o respetivo ficheiro carregando-o para a cache e assumirá um papel de Servidor primário para esse domínio. Quando no ficheiro de configuração aparece um campo SP então para o domínio apresentado o servidor funcionará como servidor secundário e terá de pedir transferência de zona para obter os dados que necessita.

Este módulo contém então obviamente uma função que faz parse ao ficheiro de configuração guardando os dados em estruturas adequadas que têm por exemplo as informações de quem são os SPs e SSs dos domínios e quais os respetivos endereços. Posteriormente temos funções que percorrem as estruturas referidas anteriormente e para o caso dos SS fazem o pedido de transferência de zona e no caso dos SP fazem a leitura do ficheiro.

Após termos os dados que necessitamos carregados na cache então estamos prontos para receber queries no socket UDP.

De forma a termos sempre um socket UDP pronto a receber queries e um socket TCP sempre pronto a receber pedidos de transferências de zona foram usados dois threads.

7.2 Cliente

No cliente a implementação é mais simples, apenas temos um mecanismo que transforma as queries escritas pelo utilizador no formato pretendido para as comunicações e envia para um qualquer servidor (o utilizador introduz o endereço) e espera a resposta, verifica se têm um formato válido e imprime o conteúdo.

7.3 Cache

Uma parte vital de todo o sistema é a cache, é usada em todos os servidores, logo o código é usado em vários sítios. Para a cache usamos um Array de estruturas (as estruturas contém os campos referidos no enunciado), o tamanho da cache é calculado de acordo com o tamanho do primeiro ficheiro de base de dados lido e têm um extra de 300 entradas.

7.4 Opções de execução

De forma aos nossos servidores poderem ter configurações ligeiramente diferentes ao serem iniciados (mesmo antes de lerem os ficheiros de configuração) o nosso programa "dnsServer" é capaz de receber algumas flags, estas são:

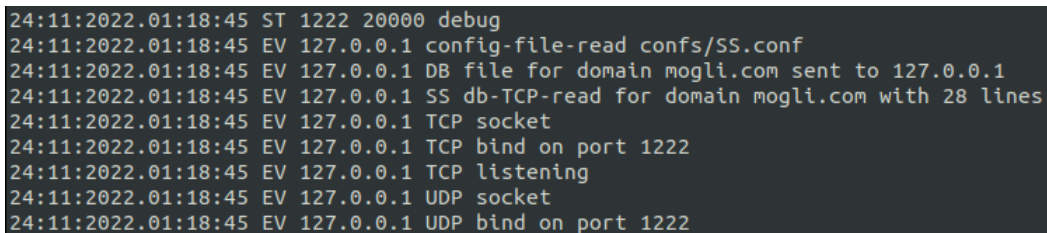
- - -debug; Este modo é o modo verboso em que tudo o que é escrito no log também é impresso no terminal.
- - -shy; Este modo é o modo não verboso, nada é impresso no terminal.
- - -port + porta; Esta flag permite alterar a porta em que o servidor irá correr.
- - -timeout + tempo; Esta flag permite alterar o valor do timeout;
- - -conf + path; Esta flag permite alterar a localização do ficheiro de configuração.
- - -log + path; Esta flag permite alterar a localização do ficheiro de configuração.

O ficheiro de configuração por defeito está em "confs/SP.conf" O ficheiro de log por defeito está em "logs/SP.log"

7.5 Atividades reportadas no log

Nesta fase apenas foram implementados os logs gerais para os servidores, ou seja, ainda não existem logs para os dominios. As mensagens possiveis no log são:

- Sucesso ou falha na leitura de ficheiro de configuração
- Sucesso ou falha na leitura de ficheiro de base de dados
- Aviso de erros nos ficheiros de conf e bd
- Sucesso ou falha no envio de um dado ficheiro de base de dados em transferência de zona
- Anotação de query inválida
- Sucesso ou falha a criar o socket TCP
- Sucesso ou falha a criar o socket UDP
- Sucesso ou falha na recessão do ficheiro na transferência de zona
- Aviso de endereços IP inválidos
- Aviso de encerramento incorreto do sistema



```
24:11:2022.01:18:45 ST 1222 20000 debug
24:11:2022.01:18:45 EV 127.0.0.1 config-file-read confs/SS.conf
24:11:2022.01:18:45 EV 127.0.0.1 DB file for domain mogli.com sent to 127.0.0.1
24:11:2022.01:18:45 EV 127.0.0.1 SS db-TCP-read for domain mogli.com with 28 lines
24:11:2022.01:18:45 EV 127.0.0.1 TCP socket
24:11:2022.01:18:45 EV 127.0.0.1 TCP bind on port 1222
24:11:2022.01:18:45 EV 127.0.0.1 TCP listening
24:11:2022.01:18:45 EV 127.0.0.1 UDP socket
24:11:2022.01:18:45 EV 127.0.0.1 UDP bind on port 1222
```

Figura 4: Exemplo de entradas de log num SS

8 Planeamento do ambiente de teste

O ambiente de teste consistirá em: 2 servidores de topo, 2 servidores de domínios de topo, um domonio de DNS reverso, 1 sP e 2 SS para cada dominio e subdominio do sistema, um SR e um CL.

A topologia no core será parecida à do TP1 (poderemos ter necessidade de adicionar novos hosts).

- Servidor de Topo 1 → Servidor1
- Servidor de Topo 2 → Orca
- Servidor Primário do domínio de topo animalia → Servidor3
- Servidor Secundário do domínio de topo animalia → Grilo
- Servidor Secundário do domínio de topo animalia → Portatil3
- Servidor Primário do domínio de topo plantae → Golfinho
- Servidor Secundário do domínio de topo plantae → Grilo
- Servidor Secundário do domínio de topo plantae → Portatil1
- Servidor Primário do sub-domínio cao.animalia → Cigarra
- Servidor Secundário do sub-domínio cao.animalia → Foca
- Servidor Secundário do sub-domínio cao.animalia → Servidor2
- Servidor Primário do sub-domínio embryophyta.plantae → Portatil2
- Servidor Secundário do sub-domínio embryophyta.plantae → Foca
- Servidor Secundário do sub-domínio embryophyta.plantae → Servidor2
- Servidor Resolver → Vespa

A forma como dividimos os servidores tenta minimizar o número de hosts necessários e maximizar a redundancia.

8.1 Ficheiros de configuração e base de dados

Nota: Os ficheiros apresentados podem ainda ter de ser alterados numa fase futura.

```
#ficheiro de configuracao para os servidores de topo
# Sintaxe: IP dominio

root ST dns/rootServers.db
```

Figura 5: Ficheiro de configuração para os ST

```
# ficheiro de configuração para o dominio de topo animalia
# funciona na porta por defeito em 10.2.2.3
# é servidor de topo de animalia e de plantae

# base de dados de animalia
animalia DB dns/animalia.db
animalia LG logs/animalia.log
# secundários de animalia
animalia SS 10.4.4.1
animalia SS 10.1.1.3

root ST dns/rootServers.db
all LG logs/dns.log
```

Figura 6: Ficheiro de configuração do SP do dominio de topo animalia

```
# ficheiro de configuração para o dominio de topo plantae
# funciona na porta por defeito em 10.3.3.2
# é servidor de topo de funchi e secundario de topo de animalia e plantae

# base de dados de plantae
plantae DB dns/plantae.db
plantae LG logs/plantae.log
#secundários de plantae
plantae SS 10.4.4.1
plantae SS 10.1.1.1

all LG logs/dns.log
```

Figura 7: Ficheiro de configuração do SP do dominio de topo plantae

```
# ficheiro de configuração para secundario de dominios de topo
# funciona na porta por defeito em 10.4.4.1
# é servidor secundario de animalia e plantae

# sou secundario de animalia e plantae
animalia SP 10.2.2.3
animalia LG logs/animalia.log
plantae SP 10.3.3.2
plantae LG logs/plantae.log

all LG logs/dns.log
```

Figura 8: Ficheiro de configuração do SS1 dos dominios de topo plantae e animalia


```
# ficheiro de configuração para secundario de dominios de topo
# funciona na porta por defeito em 10.1.1.3
# é servidor secundario de topo de funchi, animalia e plantae

# sou secundario de animalia e plantae
animalia SP 10.2.2.3
animalia LG logs/animalia.log
plantae SP 10.3.3.2
plantae LG logs/plantae.log

all LG logs/dns.log
```

Figura 9: Ficheiro de configuração do SS2 dos dominios de topo plantae e animalia

```
# ficheiro de configuração para o sub-dominio cao.animalia
# funciona na porta por defeito em 10.2.2.3
# é servidor de topo de animalia

# base de dados de cao.animalia
cao.animalia DB dns/cao.db
cao.animalia LG logs/cao.log

# secundários de cao.animalia
cao.animalia SS 10.3.3.3
cao.animalia SS 10.2.2.2

all LG logs/dns.log
```

Figura 10: Ficheiro de configuração do SP do sub-dominio cao.animalia

```
# ficheiro de configuração para o sub-domínio embryophyta.plantae. e viridiplantae.plantae.
# funciona na porta por defeito em 10.1.1.2
# é servidor de topo de plantae

# base de dados de embryophyta.plantae.
embryophyta.plantae DB dns/embryophyta.db
embryophyta.plantae LG logs/embryophyta.log

# secundário de embryophyta.plantae
embryophyta.plantae SS 10.3.3.3
embryophyta.plantae SS 10.2.2.2

all LG logs/dns.log
```

Figura 11: Ficheiro de configuração do SP do sub-domínio embryophyta.plantae

```
# ficheiro de configuração para o secundario de todos os sub dominios
# funciona na porta por defeito em 10.3.3.3

cao.animalia SP 10.2.2.3
cao.animalia LG logs/cao.log

embryophyta.plantae SP 10.1.1.2
embryophyta.plantae LG logs/embryophyta.log

all LG logs/dns.log
```

Figura 12: Ficheiro de configuração do SS1 dos sub-dominios cao.animalia e embryophyta.plantae

```
# ficheiro de configuração para o secundario de todos os sub dominios
# funciona na porta por defeito em 10.2.2.2

cao.animalia SP 10.2.2.3
cao.animalia LG logs/cao.log

embryophyta.plantae SP 10.1.1.2
embryophyta.plantae LG logs/embryophyta.log

all LG logs/dns.log
```

Figura 13: Ficheiro de configuração do SS2 dos sub-dominios cao.animalia e embryophyta.plantae

```
#servidor primario / servidor dominio de topo animalia
10.2.2.3 animalia.
#servidor primario plantae / servidor dominio de topo plantae
10.3.3.2 plantae.
```

Figura 14: Ficheiro root

```
# tabela DNS para o dominio de topo animalia
# com apontadores para os subdominios

@ DEFAULT animalia
TTL DEFAULT 86400

@ SOASP mogli.cao.animalia. TTL
@ SOADMIN a99999\alunos.uminho.pt. TTL
@ SOASERIAL 23434523 TTL
@ SOAREFRESH 14400 TTL
@ SOARETRY 3600 TTL
@ SOAEXPIRE 604800 TTL

@ NS sdt1.animalia. TTL
@ NS sdt2.animalia. TTL

cao.@ NS mogli.cao.animalia.
gato.@ NS mogli.cao.animalia.

@ MX mx1.animalia. TTL
@ MX mx2.animalia. TTL

sdt1 A 10.2.2.1
sdt2 A 10.2.2.2
mx1 A 10.2.2.1

mogli.cao A 10.2.2.5
```

Figura 15: Ficheiro db do dominio de topo animalia

```
# tabela DNS para o dominio de topo plantae
# com apontadores para os subdominios

@ DEFAULT plantae
TTL DEFAULT 86400

@ SOASP carvalho.embryophyta.plantae. TTL
@ SOADMIN a99999\alunos.uminho.pt. TTL
@ SOASERIAL 23434523 TTL
@ SOAREFRESH 14400 TTL
@ SOARETRY 3600 TTL
@ SOAEXPIRE 604800 TTL

@ NS sdt1.plantae. TTL

embryophyta.@ NS carvalho.embryophyta.plantae.
viridiplantae.@ NS carvalho.embryophyta.plantae.

@ MX mx1.plantae. TTL

# mesmo servidor que animalia
sdt1 A 10.2.2.1
# o servidor de email é o sdt2
mx1 A 10.2.2.2

# o carvalho tb responde pelo viridiplantae
carvalho.embryophyta A 10.2.2.7
```

Figura 16: Ficheiro db do dominio de topo plantae

```
# tabela DNS para o dominio de embryophyta.plantae.

@ DEFAULT embryophyta.plantae
TTL DEFAULT 86400

@ SOASP carvalho.embryophyta.plantae. TTL
@ SOADMIN a99999\alunos.uminho.pt. TTL
@ SOASERIAL 23434523 TTL
@ SOAREFRESH 14400 TTL
@ SOARETRY 3600 TTL
@ SOAEXPIRE 604800 TTL

@ NS carvalho.embryophyta.plantae. TTL

@ MX mx1.embryophyta.plantae. TTL

carvalho A 10.2.2.220
eucalipto A 10.2.2.221

mx1 CNAME carvalho
clipe CNAME eucalipto
```

Figura 17: Ficheiro db do sub-dominio embryophyta.plantae

```
# tabela DNS para o dominio de cao.animalia

@ DEFAULT cao.animalia
TTL DEFAULT 86400

@ SOASP mogli.cao.animalia. TTL
@ SOADMIN a99999\alunos.uminho.pt. TTL
@ SOASERIAL 23434523 TTL
@ SOAREFRESH 14400 TTL
@ SOARETRY 3600 TTL
@ SOAEXPIRE 604800 TTL

@ NS mogli.cao.animalia. TTL

@ MX mx1.cao.animalia. TTL

mogli A 10.2.2.5
boby A 10.2.2.100

mx1 CNAME mogli
selva CNAME mogli
```

Figura 18: Ficheiro db do sub-dominio cao.animalia

9 Como testar o sistema da fase 1

Dentro da pasta submetida temos a pasta Src, dentro desta pasta basta usar o comando make para compilar o código necessário.

- Para testar o SP usa-se o comando: `"./dnsServer -port 5000 -conf confs/SP.conf "`
- Para testar o SS usa-se o comando: `"./dnsServer -port 5001 -conf confs/SS.conf "`

Nota: Pode ser necessário alterar os ficheiros de configuração com o endereço em que está a correr o SP e o SS.

Exemplos de queries para testar:

- `./dnsCL 10.2.2.1:1234 www.mogli.animalia. A R`
- `./dnsCL 10.2.2.1:1234 mogli.animalia. MX R`

10 Atividades Desenvolvidas e Participação

Atividade	Henrique	Alexandre	Délio
Arquitetura do Sistema	35	35	30
Modelo de Informação	40	35	25
Modelo de Comunicação	40	30	30
Planeamento do Ambiente de Teste	33.3	33.3	33.3
Protótipo SP e SS em modo debug	35	35	30
Protótipo CL em modo debug	40	30	30
Relatório	35	35	30