



UNIVERSIDADE DO MINHO
LICENCIATURA EM ENGENHARIA INFORMÁTICA

Comunicação por Computadores
Trabalho Prático 2 - Fase 2 - LEI 2022/23

Mário Nelson Neto Santos (a70697)
Diogo Viana Ramos Casal Novo (a88276)
Alexandra Dias Candeias (a89521)

Janeiro 2023

Conteúdo

1	Sistema DNS	4
1.1	Introdução	4
2	Modelo comunicacional	5
2.1	Visão geral	5
2.2	Querys DNS	5
2.3	Transferência de Zona	6
3	Componentes do sistema	8
3.1	Introdução	8
3.2	Servidor Primário	8
3.3	Servidor Secundário	10
3.4	Servidor de Resolução	10
3.5	Caching	11
3.6	Cliente	11
4	Ambiente de Testes	12
4.1	testes	13
4.1.1	Teste 1:	13
4.1.2	Teste 2:	14
4.1.3	Teste 3:	15
4.1.4	Teste 4:	17
4.1.5	Teste 5:	19
4.1.6	Teste 6:	20

Capítulo 1

Sistema DNS

1.1 Introdução

O Sistema de Nomes de Domínio (DNS) é uma rede distribuída de servidores que são responsáveis por resolver os endereços de nomes de domínio para endereços IP. Isso permite que os utilizadores acedam a websites e outros serviços de Internet de forma mais fácil, através de endereços mnemônicos em vez de ter que memorizar endereços IP numéricos. Este trabalho prático foi desenvolvido com o objetivo de criar um sistema DNS com os seus componentes básicos: servidor primário, servidor secundário, servidor de resolução e cliente. Foi escolhida a linguagem Python devido à sua sintaxe simples e à necessidade de respostas rápidas aos requisitos apresentados. Embora o Python possa não ser a melhor opção em termos de performance para um sistema DNS, consideramos que se adequou melhor ao nosso plano de ação entre as linguagens viáveis. Neste relatório, tentaremos explicar de forma clara e objetiva o nosso processo de pensamento, decisão e execução dos vários componentes.

Capítulo 2

Modelo comunicacional

2.1 Visão geral

Neste projeto lidaremos maioritariamente com dois intuitos de comunicação: o envio de uma querye DNS ou uma transferência de zona. As queries DNS são enviadas utilizando o protocolo UDP, que é um protocolo de camada de transporte de alta velocidade e não confiável, isto significa que podem ser perdidas ou danificadas durante o envio, mas isso não é um problema, pois a resposta a uma query é geralmente pequena e pode ser facilmente refeita. A transferência de zona, por outro lado, é um processo mais lento e importante, pois envolve o envio de toda a informação de um domínio para outro servidor DNS. Para garantir a integridade e confiabilidade do processo, a transferência de zona é realizada através de um protocolo de camada de transporte mais confiável, o TCP.

2.2 Querys DNS

Todas as nossas comunicações são efetuadas através de mensagens encapsuladas no protocolo UDP, na figura 2.1 podemos ver a estrutura de uma dessas mensagens.

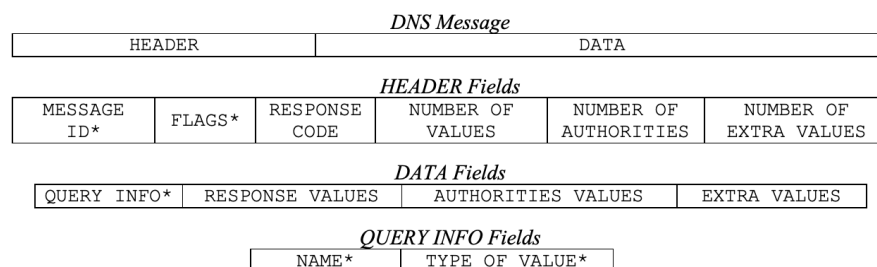


Figura 2.1: Representação das mensagens DNS utilizadas no sistema.

Esta estrutura é baseada na estrutura proposta pelo enunciado.

Uma mensagem é constituída pelos campos HEADER e DATA, subdivididos nos seguintes:

- **MessageID:** identificador de mensagem (número inteiro entre 1 e 65535, gerado aleatoriamente pelo CL ou servidor que faz a query original) que irá ser usado para relacionar as respostas recebidas com a query original;
- **Flags:** As flags são usadas para definir como o servidor ao qual foi pedido a informação, vai operar. Existem três possíveis valores:
 - **A:** esta flag é usada para que indicar que o servidor que respondeu tem autoridade sobre a query em questão;
 - **Q:** indica que a query é um pedido, sendo a sua ausência interpretada como a indicação de que esta é uma resposta a um pedido;
 - **R:** indica que deve ser executada recursivamente
- **Response Code:** indica o código de erro da resposta, se for 1 o domínio existe mas não foi encontrado correspondência para o tipo de registo, se for 2 o domínio não existe, se for 3 a mensagem DNS não foi codificada corretamente.
- **Number of Values:** número de entradas com a resposta exata pedida na query.
- **Number of authorities:** número de entradas de registos sobre servidores autoritativos.
- **Number of extra values:** número de entradas adicionais.
- **Response Values:** lista de entradas que fazem match em NAME e TYPE OF VALUE com o requisitado
- **Authorities Values:** lista de entradas de TYPE OF VALUE NS que representam os servidores com autoridade sobre o domínio.
- **Extra Values:** lista de entradas de TYPE OF VALUE A para servidores cujo nome foi referido num dos campos da query mas que o seu endereço não foi pedido
- **Name:** Nome do domínio sobre o qual se pretende obter informação
- **Type of Value:** O TYPE OF VALUE corresponde a um dos tipos de valor suportados na base de dados

2.3 Transferência de Zona

A transferência de zona consiste em replicar a base de dados a que o SP tem acesso, nos restantes SS que estão autorizados no domínio. O método escolhido é idêntico ao do enunciado sendo a única diferença o segundo passo é feito juntamente com o terceiro. Ou seja, o número de linhas é enviado juntamente com o número de série da base de dados.

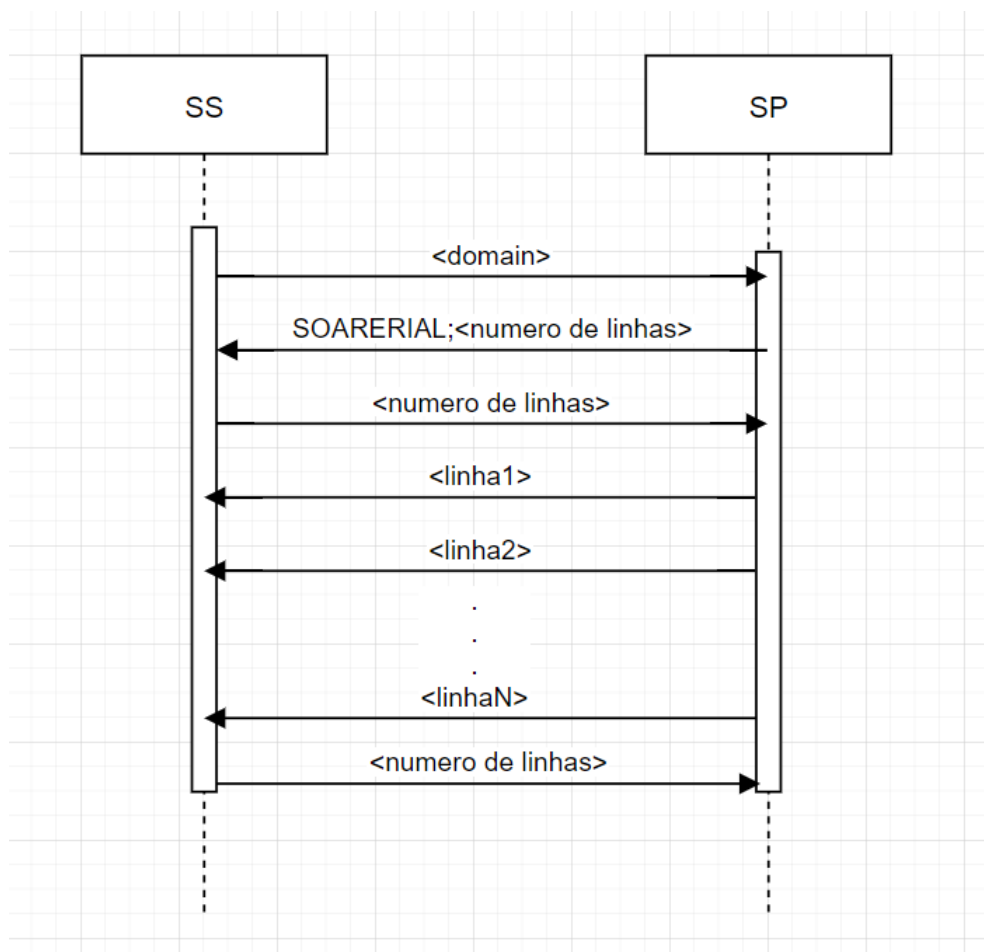


Figura 2.2: Diagrama comunicacional de transferência de zona

Capítulo 3

Componentes do sistema

3.1 Introdução

Foram implementados três tipos de servidor: servidor DNS primário, servidor DNS secundário e servidor de resolução

Um único servidor pode ser servidor DNS primário, servidor DNS secundário e servidor de resolução simultaneamente desde que os domínios de cada um deles sejam independentes, isto quer dizer que nenhum domínio pode pertencer a um subdomínio do próprio servidor.

Um servidor DNS primário é responsável por manter a base de dados principal de registos DNS para um determinado domínio. Ele atualiza as bases de dados dos servidores DNS secundários e também pode receber queries (consultas) do servidor de resolução relativamente a pedidos de clientes sobre os registos DNS de um domínio específico.

Um servidor DNS secundário é responsável por manter uma cópia da base de dados do servidor DNS primário para o mesmo domínio. Ele serve como uma segunda fonte de informação para que, na eventualidade do servidor primário não estar ativo ou não conseguir responder, a informação possa ser na mesma consultada. Um servidor de resolução pode encaminhar as queries dos clientes para o servidor secundário em caso de o servidor primário não estar disponível.

O servidor de resolução é responsável por receber pedidos de clientes e encaminhá-los para o servidor DNS apropriado para obter a resposta. Ele pode encaminhar as queries para um servidor DNS primário ou secundário, dependendo da disponibilidade e da configuração dos servidores.

Todos os servidores deveram ter no mínimo uma entrada DD associada a cada domínio para que este saiba aonde ir procurar a informação

3.2 Servidor Primário

O servidor primário (SP) tem acesso a toda a informação relativa aos domínios por ele geridos e tem a função de a fornecer a entidades permitidas. No caso do nosso sistema o SP mantém os registos dos domínios para os quais é primário em ficheiros de base de dados independentes.

O Servidor, ao ser iniciado, efetua parse de um ficheiro de configuração que contém a in-

formação de todos os domínios que gere. Quando esta informação possui entradas com identificação dos ficheiros das bases de dados, identificação de endereços de servidores DNS secundários (não obrigatório), uma única entrada DD com o ip *127.0.0.1* e identificação do ficheiro de logs (não obrigatório), este identifica que o servidor será SP para o domínio em questão e inicia um novo parse desta vez do ficheiro de base de dados supracitado.

Após leitura e interpretação da informação de todos os ficheiros de base de dados, dos domínios do servidor primário, este lança uma thread para que esta fique à espera de pedidos de atualização das bases de dados (Transferências de Zona) por parte do servidor DNS secundários que tenham autoridade sobre os domínios em questão.

A seguir são apresentados exemplos de ficheiros de base de dados e configuração válidos para um SP:

```
17 @ DEFAULT azurem.uni.
18 TTL DEFAULT 86400
19
20 @ SOASP ns1.azurem.uni. TTL
21 @ SOADMIN dns\admin.azurem.uni. TTL
22 @ SOASERIAL 0130122022 TTL
23 @ SOAREFRESH 14400 TTL
24 @ SOARETRY 3600 TTL
25 @ SOAEXPIRE 604800 TTL
26
27 @ NS ns1.azurem.uni. TTL
28 @ NS ns2.azurem.uni. TTL
29 @ NS ns3.azurem.uni. TTL
30
31 mec.@ NS ns1.mec.azurem.uni. TTL
32 mec.@ NS ns2.mec.azurem.uni. TTL
33 mec.@ NS ns3.mec.azurem.uni. TTL
34
35 @ MX mx1.azurem.uni. TTL 10
36 @ MX mx2.azurem.uni. TTL 20
37
38 ns1 A 10.0.16.10 TTL
39 ns2 A 10.0.17.10 TTL
40 ns3 A 10.0.16.16 TTL
41 ns1.mec A 10.0.18.10 TTL
42 ns2.mec A 10.0.19.10 TTL
43 ns3.mec A 10.0.18.16 TTL
44 mx1 A 10.0.16.11 TTL
45 mx2 A 10.0.16.12 TTL
46 www1 A 10.0.16.14 TTL 200
47 www2 A 10.0.16.15 TTL 200
48 ftp A 10.0.16.13 TTL
49
50 sp CNAME ns1 TTL
51 ss1 CNAME ns2 TTL
52 ss2 CNAME ns3 TTL
53 mail1 CNAME mx1 TTL
54 mail2 CNAME mx2 TTL
```

Figura 3.1: Ficheiro de base de dados do SP

```
7 azurem.uni SP 10.0.16.10
8 azurem.uni DD 127.0.0.1
9 azurem.uni LG log/azurem-ss.log
10 all LG log/all.log
11 root ST dnsFiles/rootservers.db
```

Figura 3.2: Ficheiro de configuração do servidor primário

3.3 Servidor Secundário

O servidor DNS secundário (SS) funciona como uma salvaguarda para os casos em que o servidor DNS primário falha.

Como já referido anteriormente, o Servidor, ao ser iniciado, efetua parse de um ficheiro de configuração. Quando esta informação possui entradas de identificação do endereço do servidor DNS primário, uma única entrada DD com o ip *127.0.0.1* e identificação do ficheiro de logs (não obrigatório), este identifica que o servidor será SS para o domínio em questão e inicia uma transferência de zona para servidor DNS primário referido.

Para cada SS são efetuados regularmente pedidos de atualização da base de dados ao servidor primário, requisitando, inicialmente, o número de série do ficheiro de um determinado domínio, sendo este usado para perceber se a versão da base de dados no SP é diferente da versão usada no SS. Para esse caso será necessário fazer a transferência de zona.

A seguir é apresentado um exemplo de um ficheiro de configuração válido para um SS:

```
7 azurem.uni SP 10.0.16.10
8 azurem.uni DD 127.0.0.1
9 azurem.uni LG log/azurem-ss.log
10 all LG log/all.log
11 root ST dnsFiles/rootservers.db
```

Figura 3.3: Ficheiro de configuração do SS

3.4 Servidor de Resolução

O servidor de resolução (SR) funciona como um intermediário entre um cliente e os vários servidores. Deverá ser o local para o qual os clientes comunicam, pois, este servidor está responsável por comunicar com os restantes servidores de forma a responder ao que foi pedido.

Como já referido anteriormente, o Servidor, ao ser iniciado, efetua parse de um ficheiro de configuração. Quando esta informação possui apenas entradas do tipo *DD* e identificação do ficheiro de logs (não obrigatório) associados ao domínio, este identifica que o servidor será SR para o tal domínio

A seguir é apresentado um exemplo de um ficheiro de configuração válido para um SS:

```
4 azurem.uni DD 10.0.16.10
5 azurem.uni DD 10.0.17.10
6 azurem.uni DD 10.0.16.16
7
8 tp LG log/azurem.log
9 all LG log/all.log
10 root ST dnsFiles/rootservers.db
```

Figura 3.4: Ficheiro de configuração do SR

3.5 Caching

Foram implementados dois tipos de caching: caching de base de dados e caching de consulta (queries). O primeiro armazena os registos de uma base de dados em cache para acelerar o acesso aos dados. A cache de consulta armazena os resultados de consultas de nomes de domínio (DNS) em cache para acelerar a resposta a consultas iguais futuras.

Ambas as caches são implementadas como dicionários em Python e são preenchidas com entradas de diferentes fontes, como arquivos de configuração ou novas consultas DNS. As entradas na cache de base de dados incluem o domínio, o tipo de consulta, o valor de resposta, o tempo de vida (TTL) e a prioridade. As entradas na cache de consulta incluem o domínio, o tipo de consulta, o código de resposta, os valores de resposta, os valores de autoridade, os valores extra, o número de valores, o número de autoridades e o número de valores extra, além de um timestamp e se a entrada é ou não autoritativa. As entradas em ambas as caches são atualizadas quando novas entradas são adicionadas ou quando os valores de TTL das entradas já existentes expiram. As funções "get" e "getQueryCache" são usadas para retornar entradas de cada uma das caches. A função "put" é usada para adicionar entradas à cache de base de dados e a função "putQueryCache" é usada para adicionar entradas à cache de consulta. Além disso, a classe Cache também inclui funções para retornar entradas de base de dados a partir de ficheiros de configuração e para carregar novas consultas em ambas as caches. As funções "debug" e "incTimeStamp" também são fornecidas para depuração e controlo do tempo de expiração das entradas de cache, correspondentemente.

3.6 Cliente

O cliente é capaz de enviar uma query DNS a um servidor específico, identificado pelo endereço IP e porta fornecidos. A query é criada com base no domínio, tipo de consulta e flags fornecidas pelo usuário. Depois de enviar a query para o servidor, o cliente aguarda a resposta e, em seguida, imprime-a.

Capítulo 4

Ambiente de Testes

O ambiente de testes para o sistema DNS é uma parte fundamental do desenvolvimento e manutenção de um sistema de nomes de domínio. Neste ambiente, podemos construir e configurar diferentes tipologias de redes, de modo a simular diferentes cenários e testar o nosso sistema de DNS de forma a garantir o seu correto funcionamento.

Uma das ferramentas que podemos utilizar neste contexto é o Core network emulator. Através do Core podemos construir uma tipologia de redes e simular diferentes cenários, como por exemplo a interconexão de redes locais ou a utilização de servidores de nomes de diferentes zonas.

Na figura abaixo mostramos a representação da nossa tipologia no Core GUI.

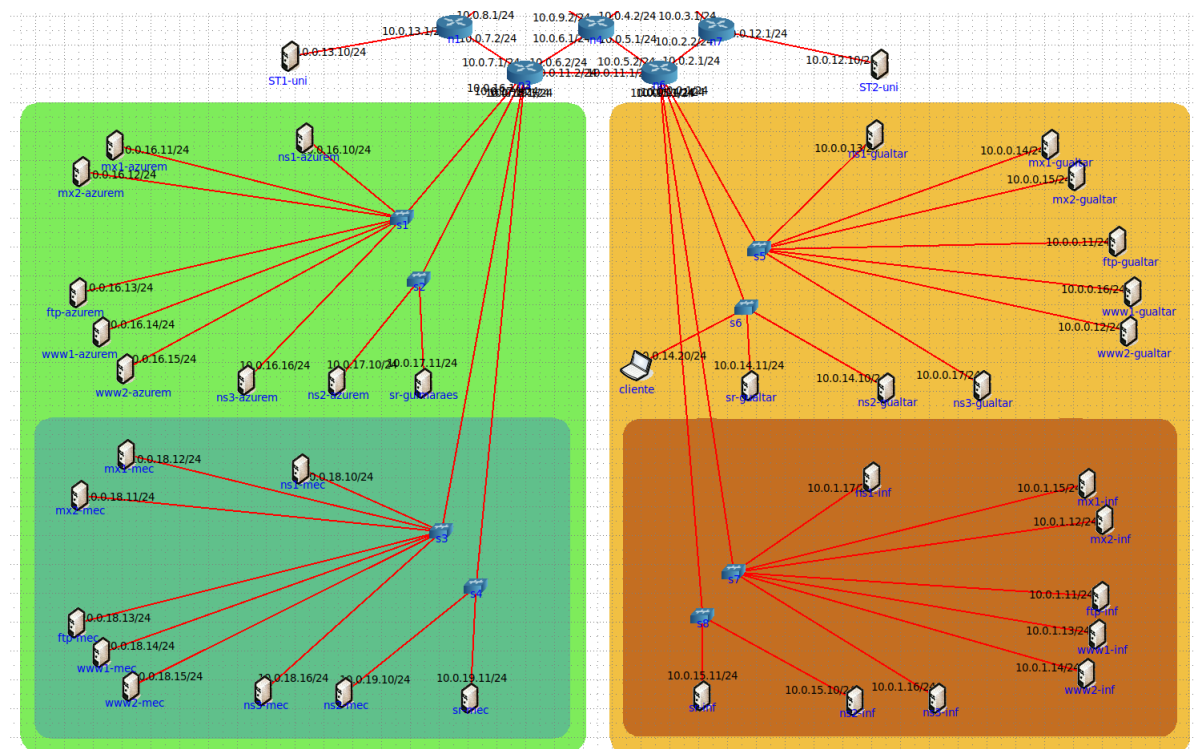


Figura 4.1: Topologia de rede

Na topologia apresentada é possível verificar que existem dois servidores de topo (ST) e seis servidores de domínio de topo (SDT), sendo dois destes servidores primários e os restantes servidores secundários. Para além destes temos acesso a seis servidores associados a sub-domínios, dos quais dois são SP e dois são SS. Para cada um dos domínios encontramos ainda mais seis servidores associados, dois deles de e-mail, um deles servidor de resposta e os restantes (fictícios) apenas servem para definir endereços nas nossas bases de dados de teste. Por fim, temos um cliente associado ao Switch S6 que efetuará as queries.

4.1 testes

Abaixo explicitamos alguns dos testes efetuados no ambiente descrito:

4.1.1 Teste 1:

Pedido realizado ao servidor de resolução.

- **IP** 10.0.14.11 (sr-gualtar)
- **dominio:** azurem.uni.
- **type:** MX
- **flags:** A, Q

```
core@mx2-gualtar:~/Documents/cc/dns$ python3 run.py c 10.0.14.11 azurem.uni. MX A+Q
-----
# Header
MESSAGE-ID = 3001, FLAGS = A, RESPONSE-CODE = 0,
N-VALUES = 2, N-AUTHORITIES = 3, N-EXTRA-VALUES = 5;
# Data: Query Info
QUERY-INFO.NAME = azurem.uni., QUERY-INFO.TYPE = MX;
# Data: List of Response, Authorities and Extra Values
RESPONSE-VALUES = azurem.uni. MX mx1.azurem.uni. 86400 10,
RESPONSE-VALUES = azurem.uni. MX mx2.azurem.uni. 86400 20;
AUTHORITIES-VALUES = azurem.uni. NS ns1.azurem.uni. 86400,
AUTHORITIES-VALUES = azurem.uni. NS ns2.azurem.uni. 86400,
AUTHORITIES-VALUES = azurem.uni. NS ns3.azurem.uni. 86400;
EXTRA-VALUES = ns1.azurem.uni. A 10.0.16.10 86400,
EXTRA-VALUES = ns2.azurem.uni. A 10.0.17.10 86400,
EXTRA-VALUES = ns3.azurem.uni. A 10.0.16.16 86400,
EXTRA-VALUES = mx1.azurem.uni. A 10.0.16.11 86400,
EXTRA-VALUES = mx2.azurem.uni. A 10.0.16.12 86400;
-----
```

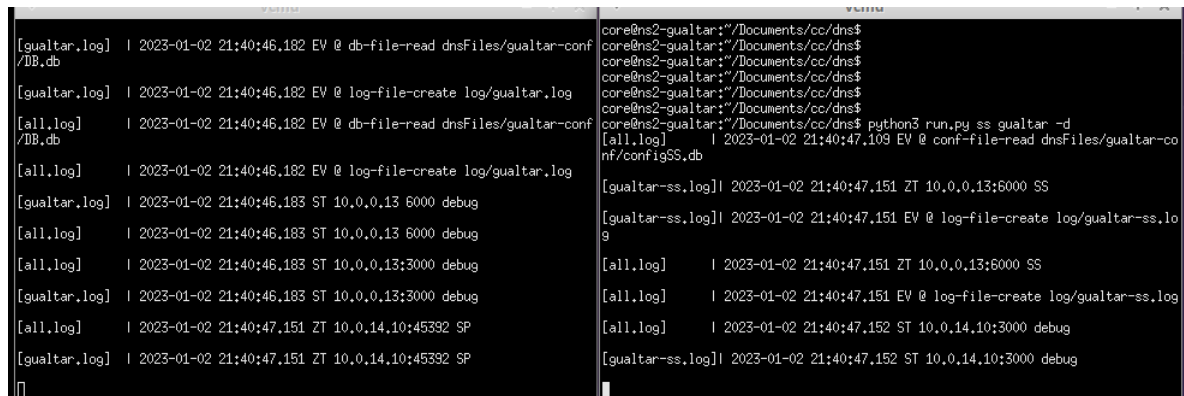
Figura 4.2: Resposta no cliente

A resposta corresponde ao esperado sendo que nos *Response Values* os registos *MX* do domínio "azurem.uni.", nas *Authorities Values* vem os valores do tipo *NS* correspondentes aos servidores com autoridade sobre o domínio e nos *Extra Values* vem os registos do tipo *A* que correspondem aos diversos domínios mencionados.

4.1.2 Teste 2:

Transferência de zona entre NS1gualtar e NS2gualtar (modo debug)

- IP ns1- 10.0.0.13
- IP ns2- 10.0.14.10
- domínio: gualtar.uni.



```
[gualtar.log] | 2023-01-02 21:40:46.182 EV @ db-file-read dnsFiles/gualtar-conf/DB.db
[gualtar.log] | 2023-01-02 21:40:46.182 EV @ log-file-create log/gualtar.log
[all.log] | 2023-01-02 21:40:46.182 EV @ db-file-read dnsFiles/gualtar-conf/DB.db
[all.log] | 2023-01-02 21:40:46.182 EV @ log-file-create log/gualtar.log
[gualtar.log] | 2023-01-02 21:40:46.183 ST 10.0.0.13 6000 debug
[all.log] | 2023-01-02 21:40:46.183 ST 10.0.0.13 6000 debug
[all.log] | 2023-01-02 21:40:46.183 ST 10.0.0.13:3000 debug
[gualtar.log] | 2023-01-02 21:40:46.183 ST 10.0.0.13:3000 debug
[all.log] | 2023-01-02 21:40:47.151 ZT 10.0.14.10:45392 SP
[gualtar.log] | 2023-01-02 21:40:47.151 ZT 10.0.14.10:45392 SP

core@ns2-gualtar:~/Documents/cc/dns$
core@ns2-gualtar:~/Documents/cc/dns$
core@ns2-gualtar:~/Documents/cc/dns$
core@ns2-gualtar:~/Documents/cc/dns$
core@ns2-gualtar:~/Documents/cc/dns$
core@ns2-gualtar:~/Documents/cc/dns$ python3 run.py ss gualtar -d
[all.log] | 2023-01-02 21:40:47.109 EV @ conf-file-read dnsFiles/gualtar-conf/configs.db
[gualtar-ss.log] | 2023-01-02 21:40:47.151 ZT 10.0.0.13:6000 SS
[gualtar-ss.log] | 2023-01-02 21:40:47.151 EV @ log-file-create log/gualtar-ss.log
[all.log] | 2023-01-02 21:40:47.151 ZT 10.0.0.13:6000 SS
[all.log] | 2023-01-02 21:40:47.151 EV @ log-file-create log/gualtar-ss.log
[all.log] | 2023-01-02 21:40:47.152 ST 10.0.14.10:3000 debug
[gualtar-ss.log] | 2023-01-02 21:40:47.152 ST 10.0.14.10:3000 debug
```

Figura 4.3: Logs transferência de zona entre NS1gualtar e NS2gualtar

Como podemos verificar pelos logs a transferência de zona foi efetuada com sucesso entre o servidor NS1gualtar e NS2gualtar. Esta ocorreu na inicialização dos servidores referidos.

4.1.3 Teste 3:

Comportamento dos diversos componentes para responder a um pedido iterativo (modo debug)

- IP 10.0.14.11 (sr-gualtar)
- dominio: inf.gualtar.uni.
- type: MX
- flags: A, Q

```
core@sr-gualtar:~/Documents/cc/dns$  
core@sr-gualtar:~/Documents/cc/dns$  
core@sr-gualtar:~/Documents/cc/dns$  
core@sr-gualtar:~/Documents/cc/dns$  
core@sr-gualtar:~/Documents/cc/dns$  
core@sr-gualtar:~/Documents/cc/dns$ python3 run.py sr gualtar -d  
[all.log] | 2023-01-02 22:02:21.777 EV @ conf-file-read dnsFiles/gualtar-co  
nf/configSR.db  
[all.log] | 2023-01-02 22:02:21.783 ST 10.0.14.11:3000 debug  
[all.log] | 2023-01-02 22:02:32.531 QR 10.0.14.20:3000  
3001:A+Q,0.0.0.0;inf.gualtar.uni.,MX;  
ns2  
[all.log] | 2023-01-02 22:02:32.531 EV @ new-cache-entry query-cache  
[all.log] | 2023-01-02 22:02:32.531 QE 10.0.14.10:3000  
3001:Q,0.0.0.0;inf.gualtar.uni.,MX;  
[all.log] | 2023-01-02 22:02:33.032 TO 10.0.14.10:3000 query  
[all.log] | 2023-01-02 22:02:33.032 QE 10.0.0.17:3000  
3001:Q,0.0.0.0;inf.gualtar.uni.,MX;  
[all.log] | 2023-01-02 22:02:33.532 TO 10.0.0.17:3000 query  
[all.log] | 2023-01-02 22:02:33.533 QE 10.0.0.13:3000  
3001:Q,0.0.0.0;inf.gualtar.uni.,MX;  
[all.log] | 2023-01-02 22:02:33.534 RR 10.0.0.13:33516  
3001:A,1.0.3.3;inf.gualtar.uni.,MX;inf.gualtar.uni. NS ns1.inf.gualtar.uni. 8640  
0;inf.gualtar.uni. NS ns2.inf.gualtar.uni. 86400;inf.gualtar.uni. NS ns3.inf.gua  
ltar.uni. 86400;ns1.inf.gualtar.uni. A 10.0.1.17 86400;ns2.inf.gualtar.uni. A 10  
.0.15.10 86400;ns3.inf.gualtar.uni. A 10.0.1.15 86400;  
[all.log] | 2023-01-02 22:02:33.534 QE 10.0.1.17:3000  
3001:Q,0.0.0.0;inf.gualtar.uni.,MX;  
[all.log] | 2023-01-02 22:02:33.535 RR 10.0.1.17:44123  
3001:A,0.2.3.5;inf.gualtar.uni.,MX;inf.gualtar.uni. MX mx1.inf.gualtar.uni. 8640  
0 10;inf.gualtar.uni. MX mx2.inf.gualtar.uni. 86400 20;inf.gualtar.uni. NS ns1.i  
nf.gualtar.uni. 86400;inf.gualtar.uni. NS ns2.inf.gualtar.uni. 86400;inf.gualtar  
uni. NS ns3.inf.gualtar.uni. 86400;ns1.inf.gualtar.uni. A 10.0.1.17 86400;mx2.i  
nf.gualtar.uni. A 10.0.15.10 86400;ns3.inf.gualtar.uni. A 10.0.1.16 86400;mx1.in  
f.gualtar.uni. A 10.0.1.15 86400;mx2.inf.gualtar.uni. A 10.0.1.12 86400;  
[all.log] | 2023-01-02 22:02:33.535 EV @ new-cache-entry server-cache  
[all.log] | 2023-01-02 22:02:33.535 EV @ new-cache-entry server-cache  
[all.log] | 2023-01-02 22:02:33.535 EV @ new-cache-entry server-cache  
[all.log] | 2023-01-02 22:02:33.535 EV @ new-cache-entry server-cache  
[all.log] | 2023-01-02 22:02:33.535 EV @ new-cache-entry server-cache  
[all.log] | 2023-01-02 22:02:33.535 EV @ new-cache-entry server-cache  
[all.log] | 2023-01-02 22:02:33.535 EV @ new-cache-entry server-cache  
[all.log] | 2023-01-02 22:02:33.535 EV @ new-cache-entry server-cache  
[all.log] | 2023-01-02 22:02:33.535 EV @ new-cache-entry server-cache  
[all.log] | 2023-01-02 22:02:33.535 RP 10.0.14.20:55764  
3001:A,0.2.3.5;inf.gualtar.uni.,MX;inf.gualtar.uni. MX mx1.inf.gualtar.uni. 8640  
0 10;inf.gualtar.uni. MX mx2.inf.gualtar.uni. 86400 20;inf.gualtar.uni. NS ns1.i  
nf.gualtar.uni. 86400;inf.gualtar.uni. NS ns2.inf.gualtar.uni. 86400;inf.gualtar  
uni. NS ns3.inf.gualtar.uni. 86400;ns1.inf.gualtar.uni. A 10.0.1.17 86400;ns2.i  
nf.gualtar.uni. A 10.0.15.10 86400;ns3.inf.gualtar.uni. A 10.0.1.16 86400;mx1.in  
f.gualtar.uni. A 10.0.1.15 86400;mx2.inf.gualtar.uni. A 10.0.1.12 86400;  
[all.log] | 2023-01-02 22:02:33.535 RP 10.0.14.11:46508  
3001:A,1.0.3.3;inf.gualtar.uni.,MX;inf.gualtar.uni. NS ns1.inf.gualtar.uni. 86400;inf.gualt  
ar.uni. NS ns2.inf.gualtar.uni. 86400;inf.gualtar.uni. NS ns3.inf.gualtar.uni. 86400;ns1.in  
f.gualtar.uni. A 10.0.1.17 86400;ns2.inf.gualtar.uni. A 10.0.15.10 86400;ns3.inf.gualtar.un  
i. A 10.0.1.15 86400;  
[all.log] | 2023-01-02 22:02:33.533 RP 10.0.14.11:46508  
3001:A,1.0.3.3;inf.gualtar.uni.,MX;inf.gualtar.uni. NS ns1.inf.gualtar.uni. 86400;inf.gualt  
ar.uni. NS ns2.inf.gualtar.uni. 86400;inf.gualtar.uni. NS ns3.inf.gualtar.uni. 86400;ns1.in  
f.gualtar.uni. A 10.0.1.17 86400;ns2.inf.gualtar.uni. A 10.0.15.10 86400;ns3.inf.gualtar.un  
i. A 10.0.1.15 86400;  
[all.log] | 2023-01-02 22:02:33.534 QR 10.0.14.11:3000  
3001:Q,0.0.0.0;inf.gualtar.uni.,MX;  
[inf.log] | 2023-01-02 22:02:33.534 EV @ new-cache-entry query-cache  
[all.log] | 2023-01-02 22:02:33.534 EV @ new-cache-entry query-cache  
[inf.log] | 2023-01-02 22:02:33.534 RP 10.0.14.11:46508  
3001:A,0.2.3.5;inf.gualtar.uni.,MX;inf.gualtar.uni. MX mx1.inf.gualtar.uni. 86400 10;inf.gua  
tar.uni. MX mx2.inf.gualtar.uni. 86400 20;inf.gualtar.uni. NS ns1.inf.gualtar.uni. 86400;inf  
gualtar.uni. NS ns2.inf.gualtar.uni. 86400;inf.gualtar.uni. NS ns3.inf.gualtar.uni. 86400;ns  
inf.gualtar.uni. A 10.0.1.17 86400;ns2.inf.gualtar.uni. A 10.0.15.10 86400;ns3.inf.gualtar  
ni. A 10.0.1.16 86400;mx1.inf.gualtar.uni. A 10.0.1.15 86400;mx2.inf.gualtar.uni. A 10.0.1.  
86400;  
[all.log] | 2023-01-02 22:02:33.534 RP 10.0.14.11:46508  
3001:A,0.2.3.5;inf.gualtar.uni.,MX;inf.gualtar.uni. MX mx1.inf.gualtar.uni. 86400 10;inf.gua  
tar.uni. MX mx2.inf.gualtar.uni. 86400 20;inf.gualtar.uni. NS ns1.inf.gualtar.uni. 86400;inf  
gualtar.uni. NS ns2.inf.gualtar.uni. 86400;inf.gualtar.uni. NS ns3.inf.gualtar.uni. 86400;ns  
inf.gualtar.uni. A 10.0.1.17 86400;ns2.inf.gualtar.uni. A 10.0.15.10 86400;ns3.inf.gualtar  
ni. A 10.0.1.16 86400;mx1.inf.gualtar.uni. A 10.0.1.15 86400;mx2.inf.gualtar.uni. A 10.0.1.  
86400;
```

Figura 4.4: Componentes apos um pedido iterativo

A resposta corresponde ao esperado sendo que nos *Response Values* os registos *MX* do domínio "inf.gualtar.uni.", nas *Authorities Values* vem os valores do tipo *NS* correspondentes aos servidores

com autoridade sobre o domínio e nos *Extra Values* vem os reegistos do tipo *A* que correspondem aos diversos domínios mencionados.

O caso acima pode ser descrito na seguinte sucessão de eventos:

- **1** - O cliente efetua um pedido ao SR
- **2** - O SR encaminha o pedido para o NS1gualtar
- **3** - NS1gualtar responde com os endereços dos servidores para o domínio requisitado
- **4** - SR encaminha pedido para NS1-inf
- **5** - NS1-inf responde com os dados
- **6** - SR encaminha resposta para Cliente

4.1.4 Teste 4:

Comportamento dos diversos componentes para responder a um pedido iterativo (modo debug)

- IP 10.0.14.11 (sr-gualtar)
- dominio: inf.gualtar.uni.
- type: MX
- flags: A, Q, R

The figure displays four terminal windows showing DNS logs for a recursive query. The logs are organized into four quadrants, each representing a different component's activity:

- Top-Left:** Shows the initial query from 3001.R+A.0.2.3.5 to inf.gualtar.uni. MX. It includes the query details and the response from the server, which contains the MX records for inf.gualtar.uni.
- Top-Right:** Shows the query being processed by the core@cliente component. It displays the query details and the response from the server, which contains the MX records for inf.gualtar.uni.
- Bottom-Left:** Shows the query being processed by the gualtar.log component. It displays the query details and the response from the server, which contains the MX records for inf.gualtar.uni.
- Bottom-Right:** Shows the query being processed by the core@ns1 component. It displays the query details and the response from the server, which contains the MX records for inf.gualtar.uni.

The logs show the flow of the query from the client to the server, and then from the server to the various components (gualtar.log, core@ns1, core@cliente) that process the query and return the response.

Figura 4.5: Componentes apos um pedido recursivo

A resposta corresponde ao esperado sendo que nos *Response Values* os registos *MX* do domínio "inf.gualtar.uni.", nas *Authorities Values* vem os valores do tipo *NS* correspondentes aos servidores com autoridade sobre o domínio e nos *Extra Values* vem os reegistos do tipo *A* que correspondem aos diversos domínios mencionados.

O caso acima pode ser descrito na seguinte sucessão de eventos:

- **1** - O cliente efetua um pedido ao SR
- **2** - O SR encaminha o pedido para o NS1gualtar
- **3** - NS1-gualtar encaminha pedido para NS1-inf
- **5** - NS1-inf responde com os dados ao NS1-gualtar
- **4** - NS1-gualtar encaminha resposta para SR
- **6** - SR encaminha resposta para Cliente

4.1.5 Teste 5:

Comportamento dos diversos componentes para responder a um pedido recursivo (modo debug)

- **IP** 10.0.14.11 (sr-gualtar)
- **dominio:** azurem.uni.
- **type:** MX
- **flags:** A, Q

```
-> Query Cache Hit
[all.log] | 2023-01-02 22:19:19,929 RP 10.0.14.20:54798
3001.0+H,0.2,3,5:inf.gualtar.uni.,MX:inf.gualtar.uni. MX mx1,inf.gualtar.uni. 86
400 10,inf.gualtar.uni. MX mx2,inf.gualtar.uni. 86400 20:inf.gualtar.uni. NS ns1
,inf.gualtar.uni. 86400,inf.gualtar.uni. NS ns2,inf.gualtar.uni. 86400,inf.gualt
ar.uni. NS ns3,inf.gualtar.uni. 86400:ns1,inf.gualtar.uni. A 10.0.1,17 86400,ns2
,inf.gualtar.uni. A 10.0.15,10 86400,ns3,inf.gualtar.uni. A 10.0.1,16 86400,mx1.
inf.gualtar.uni. A 10.0.1,15 86400,mx2,inf.gualtar.uni. A 10.0.1,12 86400;

qc

-----

|'inf.gualtar.uni.'| 'MX' | 0 | 86405 |
| 2 |['inf.gualtar.uni. MX mx1,inf.gualtar.uni. 86400 10', 'inf.gualtar.uni. MX mx2,inf.gualtar.uni. 86400 20']
| 5 |['inf.gualtar.uni. NS ns1,inf.gualtar.uni. 86400', 'inf.gualtar.uni. NS ns2,inf.gualtar.uni. 86400', 'inf.gualtar.uni. NS ns3,inf.gualtar.uni. 86400']
| 3 |['ns1,inf.gualtar.uni. A 10.0.1,17 86400', 'ns2,inf.gualtar.uni. A 10.0.15,10 86400', 'ns3,inf.gualtar.uni. A 10.0.1,16 86400', 'mx1,inf.gualtar.uni. A 10.0.1,15 86400', 'mx2,inf.gualtar.uni. A 10.0.1,12 86400']

-----

sc

-----

'inf.gualtar.uni.'|'domainLog'| 'log/all.log' |
'inf.gualtar.uni.'| 'MX' |'mx1,inf.gualtar.uni.'| 86400 | 86405 | 10 | 'OTHERS' |
'inf.gualtar.uni.'| 'MX' |'mx2,inf.gualtar.uni.'| 86400 | 86405 | 20 | 'OTHERS' |
'inf.gualtar.uni.'| 'NS' |'ns1,inf.gualtar.uni.'| 86400 | 86405 | 0 | 'OTHERS' |
'inf.gualtar.uni.'| 'NS' |'ns2,inf.gualtar.uni.'| 86400 | 86405 | 0 | 'OTHERS' |
'inf.gualtar.uni.'| 'NS' |'ns3,inf.gualtar.uni.'| 86400 | 86405 | 0 | 'OTHERS' |
'ns1,inf.gualtar.uni.'|'domainLog'| 'log/all.log' |
'ns1,inf.gualtar.uni.'| 'A' |'10.0.1,17' | 86400 | 86405 | 0 | 'OTHERS' |
'ns2,inf.gualtar.uni.'|'domainLog'| 'log/all.log' |
'ns2,inf.gualtar.uni.'| 'A' |'10.0.15,10' | 86400 | 86405 | 0 | 'OTHERS' |
'ns3,inf.gualtar.uni.'|'domainLog'| 'log/all.log' |
'ns3,inf.gualtar.uni.'| 'A' |'10.0.1,16' | 86400 | 86405 | 0 | 'OTHERS' |
'mx1,inf.gualtar.uni.'|'domainLog'| 'log/all.log' |
'mx1,inf.gualtar.uni.'| 'A' |'10.0.1,15' | 86400 | 86405 | 0 | 'OTHERS' |
'mx2,inf.gualtar.uni.'|'domainLog'| 'log/all.log' |
'mx2,inf.gualtar.uni.'| 'A' |'10.0.1,12' | 86400 | 86405 | 0 | 'OTHERS' |
```

Figura 4.6: Dados na cache do servidor de resposta

Os dados foram armazenados na cache do servidor para que depois possam ser usados posteriormente para responder a novos pedidos semelhantes.

Como podemos observar existem registos a serem guardados em ambas as caches sendo *qc* o pedido para apresentar a cache de consulta e *sc* o pedido para apresentar a cache de base de dados do servidor.

Podemos ainda observar que quando estamos em modo debug é apresentado um *Query Cache Hit* para se saber que a informação foi obtida a partir da cache daquele servidor.

4.1.6 Teste 6:

Pedido realizado ao servidor de resolução sobre um domínio que não existe.

- **IP** 10.0.14.11 (sr-gualtar)
- **domínio:** e.inf.gualtar.uni.
- **type:** MX
- **flags:** A, Q

```
core@cliente:~/Documents/cc/dns$ python3 run.py c 10.0.14.11 e.inf.gualtar.uni. MX A+Q
-----
# Header
MESSAGE-ID = 3001, FLAGS = , RESPONSE-CODE = 2,
N-VALUES = 0, N-AUTHORITIES = 6, N-EXTRA-VALUES = 6;
# Data: Query Info
QUERY-INFO.NAME = e.inf.gualtar.uni., QUERY-INFO.TYPE = MX;
# Data: List of Response, Authorities and Extra Values
AUTHORITIES-VALUES = gualtar.uni. NS ns1.gualtar.uni. 86400,
AUTHORITIES-VALUES = gualtar.uni. NS ns2.gualtar.uni. 86400,
AUTHORITIES-VALUES = gualtar.uni. NS ns3.gualtar.uni. 86400,
AUTHORITIES-VALUES = inf.gualtar.uni. NS ns1.inf.gualtar.uni. 86400,
AUTHORITIES-VALUES = inf.gualtar.uni. NS ns2.inf.gualtar.uni. 86400,
AUTHORITIES-VALUES = inf.gualtar.uni. NS ns3.inf.gualtar.uni. 86400;
EXTRA-VALUES = ns1.gualtar.uni. A 10.0.0.13 86400,
EXTRA-VALUES = ns2.gualtar.uni. A 10.0.14.10 86400,
EXTRA-VALUES = ns3.gualtar.uni. A 10.0.0.17 86400,
EXTRA-VALUES = ns1.inf.gualtar.uni. A 10.0.1.17 86400,
EXTRA-VALUES = ns2.inf.gualtar.uni. A 10.0.15.10 86400,
EXTRA-VALUES = ns3.inf.gualtar.uni. A 10.0.1.16 86400;
-----
core@cliente:~/Documents/cc/dns$
```

Figura 4.7: Resposta recebida com código de erro 2

O pedido continha um domínio inexistente. Como tal, o servidor mais próximo do domínio e com autoridade para responder sobre essa query, enviou uma resposta com o campo *Response Code* igual a 2

Capítulo 5

Conclusão

No final deste trabalho prático foi possível implementar e configurar um sistema de servidores DNS de forma a garantir a resolução eficiente de nomes de domínio para os clientes. Utilizaram-se servidores DNS primários, secundários e de resolução, que trabalharam em conjunto para proporcionar uma solução confiável e de alta disponibilidade.

Ao longo do processo, aprendemos sobre as responsabilidades de cada tipo de servidor DNS e como configurá-los de forma a obter um bom desempenho. Também exploramos diversas técnicas para monitorizar o funcionamento dos servidores e solucionar problemas quando surgiam.

Em resumo, este trabalho prático foi uma oportunidade valiosa para adquirir conhecimentos e habilidades na área da gestão e desenvolvimento de servidores DNS. Agradecemos a orientação e apoio recebidos ao longo do processo, que foram fundamentais para o sucesso deste trabalho.