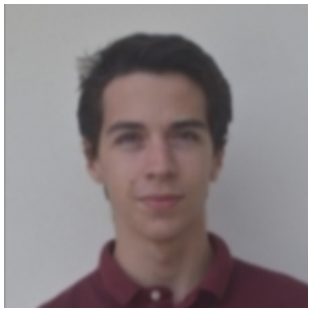
Universidade do Minho
Escola de Engenharia

Comunicações por Computador

Implementação de sistema DNS

Ricardo Lopes Lucena A97746
Xavier Santos Mota A88220
Daniel José Silva Furtado A97327



A97746

A88220

A97327

2 de janeiro de 2023

Conteúdo

1 Introdução

Este relatório é relativo ao 2º trabalho prático da Unidade Curricular de *Comunicações por Computador*, onde foi desenvolvido um projeto alusivo ao DNS (Domain Name System).

O objetivo, numa fase inicial, foi a implementação de um sistema rudimentar de DNS (com um servidor primário e dois servidores secundários), que permita fazer e responder a queries de DNS, bem como atualizar bases de dados entre servidores de DNS.

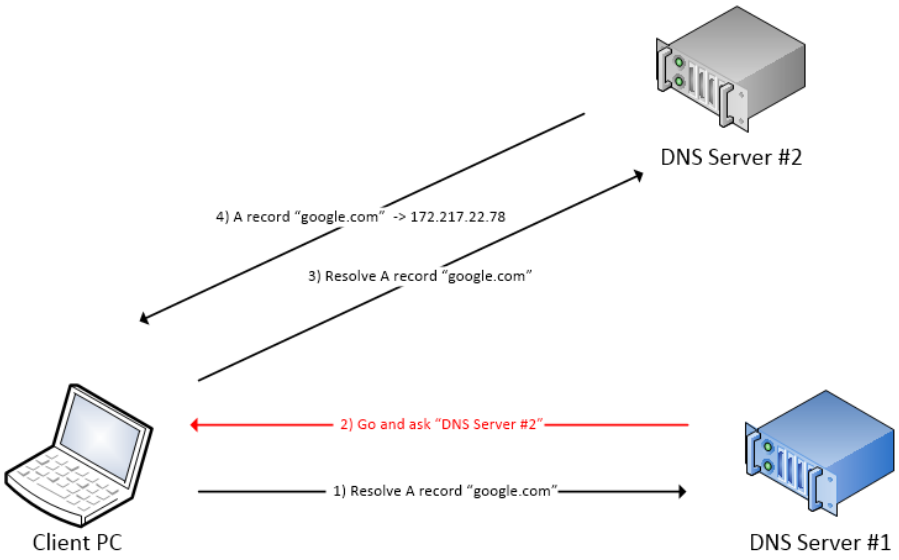


Figura 1: Exemplo de DNS do trabalho

A imagem acima apresentada, representa um esquema básico do sistema de DNS criado para esta fase. Onde o *DNS Server 1* representa o SP (servidor primário) e o *DNS Server 2* representa um dos servidores secundários.

No entanto, com a segunda fase do trabalho concluida, apresentamos a implementação de um sistema de DNS completa.

Nesta segunda fase do projeto, particularmente, corrigimos alguns aspetos da fase anterior, desenvolvemos a comunicação entre os servidores de forma hierárquica e com camadas, e também concluímos a realização do DNS reverse. Explicaremos todas os pensamentos e estratégias adotadas consoante progredimos no trabalho.

2 O que é o DNS

O DNS (*Domain Name System*) é um sistema descentralizado e hierárquico de servidores, com o objetivo de indicar a qualquer máquina o ip de um servidor apenas recebendo o seu url.

A necessidade do DNS surgiu quase que após a invenção da internet, isto porque para o ser humano é muito mais intuitivo e de fácil uso identificar o servidor que queremos aceder por um nome (endereço url) do que pela maneira usada pelos computadores (ipv4 / ipv6). Assim, foi necessário criar o DNS para fazer a relação entre endereços de ip e urls.

O DNS é um sistema descentralizado, pois isto permite maior segurança e eficiência, pois permite evitar um *central point of failure*, que seria o caso se o DNS fosse centralizado, isto é, se todos os servidores de DNS se encontrassem na mesma rede local e essa rede fosse comprometida, todo o DNS deixaria de funcionar. A descentralização dos servidores também permite um mais rápido acesso, em qualquer região do mundo, isto porque haverá sempre algum servidor perto de qualquer utilizador.

O DNS também é organizado de forma hierarquica, isto porque permite uma maior eficiencia ao lidar com pedidos. Cada servidor pertence ao seu domínio/subdomínio, o que permite separar a informação ordenadamente, permitindo o seu mais fácil acesso. Se um servidor não souber um ip pedido, aponta o cliente para um servidor que ache que possua a resposta ou saiba que servidor possui a resposta.

3 Arquitetura do Sistema

Relativamente à arquitetura do sistema, tal como na fase anterior, implementamos um servidor primário, dois servidores secundários e um cliente por domínio/subdomínio. A comunicação entre servidores de DNS e cliente é feita através de UDP, enquanto que comunicação entre servidores de DNS é feita através de TCP. Para além disso, na segunda fase foram acrescentados dois Servidores de Resolução, um por domínio, e três Servidores de Topo, sendo um destes necessário para a implementação do Domínio Reverse e os outros dois são divididos pelos dois diferentes domínios, sendo cada um representado por um servidor primário (SP) em cada domínio.

3.1 Servidores Primários(SP)

O servidor primário é a componente capaz de efetuar queries DNS e de responder às mesmas. Para iniciar, o servidor primário necessita de vários elementos nomeadamente, os **domínios** para os quais ele atua como servidor primário(example.com), identificação do ficheiro onde é suposto ele ir buscar a **base de dados** e identificação do ficheiro onde é suposto serem escritos os **logs**, **portas de atendimento**, identificação dos **Servidores Secundários** (SS) respetivos e dos SP dos subdomínios, **servidores de topo** (ST).

Para funcionar, são necessários 3 **inputs**, sendo estes o ficheiro de base de dados a ler, o ficheiro de configuração do servidor primário, e o ficheiro com a lista de servidores de topo. O **output** é o ficheiro de logs, que é escrito na diretoria dada no ficheiro de configuração anteriormente mencionado.

3.2 Servidores Secundários (SS)

Em semelhança com um Servidor Primário, o **Servidor Secundário** é capaz de responder a queries e efetuá-las. No entanto a diferença reside no facto do Servidor Secundário **não receber diretamente a base de dados** original do SP, mas sim uma réplica da mesma, que tenta manter atualizada.

Um SS tem de ter acesso a informação de configuração específica (domínios para os quais é SS, portas de atendimento, identificação dos SP dos domínios para os quais é SS, identificação do ficheiro de log, informação de segurança para acesso aos SP e endereços dos servidores de topo. Os ficheiros de input são os mesmos de um Servidor Primário, **com exceção** do ficheiro de base de dados Já os ficheiros de output, apenas há um, que é o **ficheiro de logs**.

3.3 Servidores de Resolução (SR)

Um servidor de resolução pode responder a, e efetuar queries. No entanto, **não tem qualquer autoridade** sobre algum domínio específico sendo que o seu único propósito é servir como **intermediário na resolução de queries**. Na primeira fase do trabalho não trabalhamos com um Servidor de Resolução. Já na segunda fase do trabalho necessitamos de implementar este servidor, juntamente com os servidores de Topo e outros elementos do Sistema.

No nosso projeto, um SR tem como input um ficheiro de configuração e o ficheiro com a lista de servidores de topo é obtido através da base de dados do mesmo; como output tem um ficheiro de log

3.4 Cliente (CL)

Uma aplicação cliente de DNS é o processo que precisa da informação da base de dados de DNS dum determinado domínio. A informação é obtida através da **realização de queries a um SR**.

O input e output de um CL será através da **linha de comando** sem necessidade de um ficheiro de configuração.

3.5 Servidores de Topo (ST)

Os servidores de topo têm conhecimento sobre os respetivos Servidores de Domínio de Topo. Já estes últimos atuam como servidores primários do seu domínio. Devido a disso, os ST são os servidores mais elevados na hierarquia do DNS.

3.6 Domínio Reverse

O domínio reverse do DNS é um mapeamento inverso de endereços IP para nomes de domínio. Noutras palavras, é usado para determinar o nome de domínio a partir de um endereço IP. Ele é usado principalmente para verificar a identidade de um servidor ou para encontrar o servidor responsável por um determinado endereço IP.

4 Modelo Comunicativo

Na figura baixo é possível observar a composição de uma mensagem padrão do nosso programa.

Uma mensagem DNS deve ter um cabeçalho de tamanho fixo(Header) e uma parte de dados que deve ocupar até 1 KByte. A parte dos dados contém 4 partes:

- Os dados de uma query

- Resultados dessa query

- Informação sobre os servidores que têm informação autoritativa sobre os dados da resposta

- Informação adicional indiretamente ligada aos resultados da query original

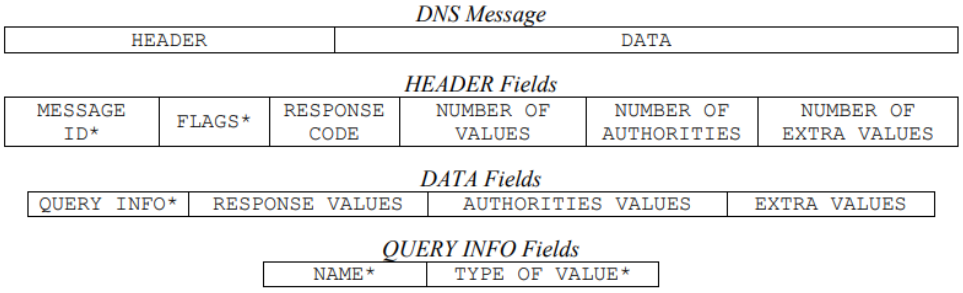


Figura 2: Modelo de Mensagens

A seguir explicaremos o significado de todos os campos do cabeçalho representados na figura acima.

- **Message ID** - Responsável por identificar uma mensagem com um número inteiro entre 1 e 65535.

- **Flags** - Só há 3 tipos de flags, Q, A, e R, que identificam, respetivamente, que a mensagem é uma query ou uma resposta a uma query, que o processo ocorre de forma recursiva, e não de forma iterativo que é a forma normal, e por último, a flag A que indica que a resposta é autoritativa.

- **Response Code** - É responsável por devolver o código de erro relativo a uma query. Este parâmetro tem 4 outputs possíveis. Caso seja 0, significa que não houve erros. No caso de ser 1 significa que o dominio incluido em "NAME"existe, mas não foi encontrada informação direta em relação ao respetivo Type of Value. Quando é 2, significa que o

dominio incluido em "Name"não existe, e quando é 3 significa que a mensagem DNS não foi descodificada corretamente.

- **Number of Values** - Responsável por indicar o número de entradas de resposta contidas numa resposta a uma query com um número inteiro entre 1 e 255
- **Number of Authorities** - Responsável por identificar o número de entradas de autoridade presentes na resposta com um número inteiro entre 1 e 255
- **Number of Extra Values** - Responsável por indicar quantas entradas de resposta adicional estão presentes na resposta com um número inteiro entre 1 e 255
- **Query Info** - Responsável por identificar a informação e o tipo de valor associado ao parâmetro da query
- **Response Values** - Responsável por indicar quantas entradas de resposta estão presentes na resposta à query
- **Authorities Values** - Responsável por indicar as entradas de autoridade contidas na resposta à query
- **Extra Values** - Responsável por indicar informações adicionais sobre o domínio ou o recurso para o qual a query foi feita

5 A nossa abordagem ao problema

Primeiro de tudo, decidimos usar como linguagem de programação o *python*, pois sentimos que é uma linguagem com a qual temos um certo à vontade, e que é relativamente fácil para resolver maior parte dos problemas.

5.1 Criação dos ficheiros de parse

Começamos por fazer os ficheiros responsáveis pelo parse de informações quer do Servidor Principal, quer da Base de Dados. Em relação ao Servidor Principal, criamos uma classe que tem todos os parâmetros necessários como variáveis da classe (base de dados, servidor secundário, diretoria dos logs, etc). Já o parse da Base de Dados faz a mesma coisa (atribuir os respetivos parâmetros às variáveis da classe) com a exceção que no fim adiciona a linha à cache.

5.2 Cache

A Cache é uma matriz, que é responsável pela organização de toda a informação do sistema, separada corretamente dentro das suas características. As linhas são acrescentadas à cache à medida que vai acontecendo o parsing da base de dados. Depois, à medida que o programa vai correndo, e à medida do valor associado ao "SOARETRY", a cache vai verificando se há alterações, e caso haja, atualiza.

5.3 Servidor Principal e Servidor Secundário

Nos ficheiros sp.py, ss.py, e respetivas threads(threadSP.py, threadSS.py) ocorre a transferência de zona. Na transferência de zona, tal como nos foi pedido, o Servidor Secundário conecta-se pelo protocolo TCP, e envia uma primeira mensagem com o domínio respetivo ao qual pretende fazer a transferência de zona. Depois, o Servidor Primário, envia o número de entries que existem na cache, e de seguida o servidor secundário responde ao SP, aceitando receber as entradas da cache.

5.4 Ficheiro de Logs

Foi-nos pedido para criar um ficheiro que ao longo do programa fosse registando o que vai acontecendo, com uma data e hora do que aconteceu. Para isso, utilizamos a diretoria presente no ficheiro de configuração do Servidor Principal, e criamos lá um ficheiro. Existe um ficheiro de logs para cada servidor. À medida que as queries vão sendo recebidas/respondidas/enviadas, definimos funções para escrever no respetivo ficheiro.

Na segunda fase do trabalho também corrigimos o que tinhamos de forma a criar um ficheiro logs de todas as atividades feitas ao longo da execução do trabalho. Mais uma vez, o programa sabe a localização deste ficheiro através dos ficheiros de configuração ("all LG"seguido da diretoria do ficheiro). Aqui estão exemplos dos ficheiros logs criados no nosso trabalho.

```
30:12:2022.01:44:48:175 EV conf-file-read configSR.txt
30:12:2022.01:54:55:724 EV conf-file-read configSR.txt
30:12:2022.01:55:43:562 EV conf-file-read configSR.txt
30:12:2022.02:02:39:777 EV conf-file-read guaxinim.txt
30:12:2022.02:02:39:777 EV log-file-create logs/guaxinim.log
30:12:2022.02:02:39:777 EV db-file-read dbs/guaxinim.db
30:12:2022.02:02:44:380 EV conf-file-read configSR.txt
30:12:2022.02:02:48:772 QR 127.0.0.1:45233 48556,Q,0,0,0,0;guaxinim.,MX;
30:12:2022.02:02:48:772 RE 127.0.0.1:45233 ['48556,A,0,2,3,5;', 'guaxinim. MX mx1.
30:12:2022.02:03:58:672 EV conf-file-read configSR.txt
30:12:2022.02:04:01:683 QR 127.0.0.1:49401 53998,Q,0,0,0,0;guaxinim.,MX;
30:12:2022.02:04:01:683 RE 127.0.0.1:49401 ['53998,A,0,2,3,5;', 'guaxinim. MX mx1.
30:12:2022.02:15:56:151 EV conf-file-read configSR.txt
30:12:2022.02:16:07:528 QR 127.0.0.1:37513 16977,Q,0,0,0,0;grande.guaxinim.,MX;
30:12:2022.02:16:07:528 RE 127.0.0.1:37513 ['16977,A,1,0,1,1;', ';', 'grande.guaxi
30:12:2022.02:17:30:001 EV conf-file-read configSR.txt
30:12:2022.02:17:33:547 QR 127.0.0.1:41233 59878,Q,0,0,0,0;grande.guaxinim.,MX;
30:12:2022.02:17:33:547 RE 127.0.0.1:41233 ['59878,A,1,0,1,1;', ';', 'grande.guaxi
30:12:2022.02:18:16:966 EV conf-file-read guaxinim.txt
30:12:2022.02:18:16:966 EV log-file-create logs/guaxinim.log
30:12:2022.02:18:16:966 EV db-file-read dbs/guaxinim.db
30:12:2022.02:18:21:370 EV conf-file-read configSR.txt
30:12:2022.02:18:24:480 QR 127.0.0.1:55805 11327,Q,0,0,0,0;grande.guaxinim.,MX;
30:12:2022.02:18:24:480 RE 127.0.0.1:55805 ['11327,A,1,0,1,1;', ';', 'grande.guaxi
30:12:2022.02:19:40:682 EV conf-file-read configSR.txt
30:12:2022.02:28:46:541 EV conf-file-read configSR.txt
30:12:2022.02:29:06:399 EV conf-file-read configSR.txt
30:12:2022.03:20:51:903 EV conf-file-read guaxinim.txt
```

Figura 3: Ficheiro logs "all"

```
27:12:2022.00:25:33:895 EV conf-file-read configs/grandeguaxinim.txt
27:12:2022.00:25:33:895 EV log-file-create logs/grandeguaxinim.log
27:12:2022.00:25:33:896 EV db-file-read dbs/grandeguaxinim.db
27:12:2022.00:25:35:641 QR 127.0.0.1:43412 51293,Q,0,0,0,0;grande.guaxinim,MX;
27:12:2022.00:25:35:642 RE 127.0.0.1:43412 ['51293,A,1,0,3,3;', ';', 'grande.guaxi
27:12:2022.02:10:10:980 EV conf-file-read configs/grandeguaxinim.txt
27:12:2022.02:10:10:980 EV log-file-create logs/grandeguaxinim.log
27:12:2022.02:10:10:980 EV db-file-read dbs/grandeguaxinim.db
27:12:2022.02:13:30:592 EV conf-file-read grandeguaxinim.txt
27:12:2022.02:13:30:592 EV log-file-create logs/grandeguaxinim.log
27:12:2022.02:13:30:592 EV db-file-read dbs/grandeguaxinim.db
27:12:2022.02:13:37:014 EV conf-file-read grandeguaxinim.txt
27:12:2022.02:13:37:014 EV log-file-create logs/grandeguaxinim.log
27:12:2022.02:13:37:014 EV db-file-read dbs/grandeguaxinim.db
27:12:2022.02:13:42:085 EV conf-file-read grandeguaxinim.txt
27:12:2022.02:13:42:085 EV log-file-create logs/grandeguaxinim.log
27:12:2022.02:13:42:085 EV db-file-read dbs/grandeguaxinim.db
27:12:2022.02:44:02:888 EV conf-file-read grandeguaxinim.txt
27:12:2022.02:44:02:888 EV log-file-create logs/grandeguaxinim.log
27:12:2022.02:44:02:888 EV db-file-read dbs/grandeguaxinim.db
27:12:2022.03:27:44:443 EV conf-file-read grandeguaxinim.txt
27:12:2022.03:27:44:443 EV log-file-create logs/grandeguaxinim.log
27:12:2022.03:27:44:444 EV db-file-read dbs/grandeguaxinim.db
```

Figura 4: Ficheiro logs de um SP

Segue uma lista dos registos efetuados nos logs, e do que cada um significa:

- **QR/QE** - Informação de que uma query é recebida/enviada, e do respetivo endereço de onde é recebida/ de onde foi enviado. Para além disso escreve os dados associados à query.
- **RP/RR** - Informação de que uma resposta a uma query é recebida/enviada, juntamente com os dados relevantes à resposta da query.
- **ZT** - Iniciado e Concluida a Transferência de Zona juntamente com o endereço do servidor na outra ponta.
- **EV** - Sempre que foi detetado um evento/atividade interna num componente, por exemplo, ficheiro de configuração/dados/ST lido, criado ficheiro de log, etc...
- **ER** - Significa que foi recebido um PDU do endereço indicado que não foi possível descodificar corretamente.
- **EZ** - É reportado sempre que o processo de transferência de zona não é executado corretamente.
- **FL** - Sempre que é reportado um erro num componente interno
- **TO** - Quando é detetado um timeout na comunicação com o servidor.
- **SP** - Execução de um componente foi parada
- **ST** - Execução de um componente foi iniciada

6 A Hierarquia

Quando um cliente envia uma query para o Servidor de Resolução, existe toda uma ordem de tentativas efetuadas pelo SR para conseguir enviar a resposta ao cliente.

A primeira coisa que o SR faz é verificar se a query já foi previamente executada. Caso esse seja o caso, a mesma vai estar presente na cache do SR, enviando assim o conteúdo presente na mesma para o cliente.

No entanto, pode acontecer de não haver nada na cache ou a query desejada não estar na mesma. Nesse caso, o SR comunica com o DD (*Default Domain*), que no caso do nosso trabalho prático é o Servidor Primário do domínio ao qual é direcionado o pedido. Caso o Servidor associado ao Default Domain não seja capaz de responder à query, o SR tem de "subir"ainda mais na "hierarquia", procurando a resposta no ST(Root-a). Na base de dados do ST, está presente a informação necessária para aceder aos SDT (Servidores de Dominio de Topo).

Todo este percurso é feito com base ao feedback que vai sendo enviado pelo servidor, mais especificamente, pelo parâmetro "Response Code".

Caso o Response Code seja igual a 0, significa que a query foi efetuada com sucesso.

Caso seja 1, o domínio da query pode existir, mas não foi encontrada a resposta para a query. Se devolver 2, significa que o domínio não existe, e no caso de ser 3, a mensagem DNS foi mal descodificada.

7 Ambiente de Testes

Relativamente à topologia usada para esta fase final do projeto, continuamos a ter a base da fase anterior, isto é, dois domínios, cada um com o seu subdomínio. Cada um destes tem um servidor primário, dois servidores secundários e um cliente. Temos também dois servidores primários do domínio root, acompanhados do respetivo reverse. Para além disso, acrescentamos nesta fase mail servers (MXs) e servidores resolvers, bem como uma variedade de clientes e servidores.

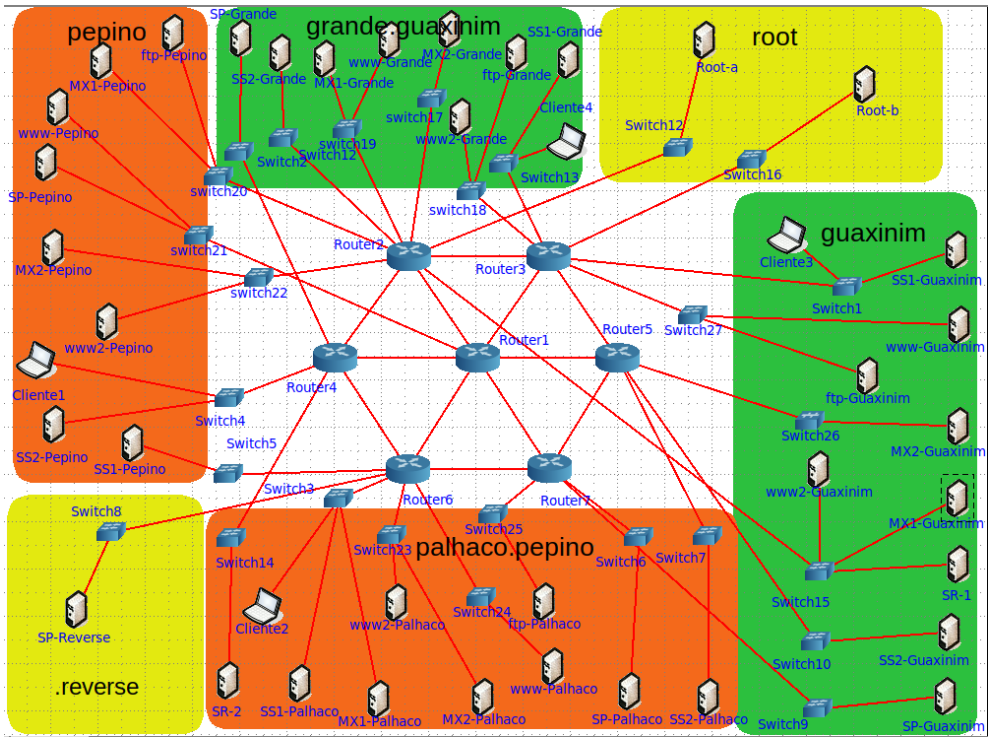


Figura 5: Topologia do Core

8 Execução do Programa

Ao longo dos próximos subtópicos, vamos mostrar as funcionalidades suportadas pelo nosso programa e o comportamente do mesmo em diversas situações.

8.1 Execução do Servidor

Sendo que nós não temos um ficheiro capaz de dar parse ao input e decifrar se ele é um SP, um SR, ou um ST, temos vários tipos de inputs possíveis para iniciar o servidor:

- **SR:** python3 sr.py configSR.txt (Porta) 200 D
- **ST:** python3 st.py configST.txt (Porta) 200 D
- **SP:** python3 sp.py (nome do dominio/subdominio).txt (Porta) 200 D

8.2 Execução do Cliente

Depois de termos o servidor inicializado, normalmente com o SR, podemos começar a usar clientes para enviarem queries para o servidor.

- **CL:** python3 cliente.py 127.0.0.1:(Porta) (dominio da query:) MX A

Aqui está uma imagem do que pode aparecer após executar o comando acima representado:

```
root@Cliente3:/home/core/Desktop/CC-Project# python3 cliente.py 10.0.28.10:53 pepino. MX A
Pedido de Query :  54665,Q,0,0,0,0;pepino.,MX;

        #-------------#         Resultado da Query         #-------------#
54665,A,0,2,3,5;
pepino. MX mx1.pepino. 86400 10,pepino. MX mx2.pepino. 86400 20;
pepino. NS ns1.pepino. 86400,pepino. NS ns2.pepino. 86400,pepino. NS ns3.pepino. 86400;
ns1.pepino. A 10.0.18.10 86400,ns2.pepino. A 10.0.16.11 86400,ns3.pepino. A 10.0.7.10 86400,mx1.pepi
no. A 10.0.35.10 86400,mx2.pepino. A 10.0.34.11 86400;
        #-------------------------------------------------------------#
```

Figura 6:

9 Contribuição do grupo para o trabalho

A figura, abaixo apresentada, representa como foi feita a divisão das tarefas para a primeira fase do trabalho, bem como o contributo de cada membro para cada tarefa.

	Cache	parse SP	parse DB	Query	thread SP thread SS	thread UDP	sp / ss	cliente	logs	topologia	relatório
Xavier	X				X	X	X	X		X	X
Daniel	X		X	X	X	X	X	X	X		X
Ricardo		X	X		X	X	X		X		X

Figura 7:

Já a figura abaixo representada representa a divisão das tarefas para a segunda fase do trabalho.

	SR	ST	DD	Ajustes Cache	Ficheiros de Configuração	Ajustes Queries	Reverse	Ajustes Topologia	Ajustes Logs	Relatório
Xavier	X	X	X	X	X			X	X	X
Daniel	X	X	X	X	X	X	X		X	
Ricardo	X	X	X	X	X	X			X	X

Figura 8:

10 Conclusão

A primeira parte deste trabalho prático revelou-se um grande desafio, no sentido em que tivemos de enfrentar um "Universo"completamente novo para nós, o que, inicialmente, nos deixou um pouco desorientados. No entanto, após finalizado, sentimo-nos bastante mais confiantes e seguros neste módulo.

Durante a implementação deparamo-nos com vários problemas e dificuldades que, com mais ou menos dificuldade, fomos resolvendo e tomando decisões na orientação do estado final do projeto. A primeira fase foi também essencial para desenvolver as nossas skills de *Python*, assim como ganhar um à vontade com o *Core*.

Na segunda fase do trabalho, com uma base sólida previamente feita, apenas tivemos de adicionar e ajustar algumas partes do trabalho que após uma particular atenção decidimos mudar. No entanto, no final de contas achamos que o sistema desenvolvido foi testado com sucesso e apresentou um bom desempenho nas operações realizadas.

Contudo, é importante ressaltar que a implementação de um sistema DNS completo envolve uma série de aspectos técnicos e de segurança que não foram abordados neste trabalho, mas que devem ser levados em consideração em aplicações reais.

Em resumo, a implementação de um sistema DNS foi um projeto enriquecedor que permitiu a aplicação dos conceitos teóricos estudados e o desenvolvimento de habilidades práticas na área de redes de computadores.

11 Bibliografia

Referências

[1] Enunciado do Trabalho Prático

[2] Material fornecido pela equipa docente na plataforma *BlackBoard*