Universidade do Minho
Escola de Engenharia

Comunicações por Computador

Licenciatura em Engenharia Informática

Grupo **6.12**

2022/2023

TP2 - Implementação de Sistema DNS
2ª fase

Alexandra Santos (A94523)

Inês Ferreira (A97372)

Joana Branco (A96584)

Braga, 2 de janeiro de 2023

Índice

Introdução

Este relatório foi desenvolvido no âmbito da Unidade Curricular de Comunicações por Computador com o objetivo de implementar um sistema DNS apto para efetuar serviços entre os diversos componentes.

Numa primeira fase é descrito e especificado de forma teórica os conhecimentos sobre o serviço DNS da arquitetura TCP e protocolos de transporte UDP e TCP, enunciando todos os componentes a utilizar. São caracterizados todos os ficheiros e os comportamentos que devem ser adotados na sua leitura e, todas as interações possíveis dos mesmos.

Já numa segunda fase é suposto apresentar as diversas operações que têm como objetivo a implementação de dois servidores que comunicam entre si e, adicionalmente, um cliente em modo debug.

1. Arquitetura do Sistema

Arquitetura da Aplicação

O DNS utiliza um grande número de servidores, que estão organizados de forma hierárquica e distribuídos pelo mundo. Nenhum servidor DNS possui todos os mapeamentos para todos os hosts na Internet.

O DNS foi criado com o objetivo de providenciar um mecanismo que permitisse a utilização de nomes como referência para dispositivos e serviços em rede. Desta forma é eliminada a necessidade do utilizador memorizar o endereço lógico (IP) associado a um determinado dispositivo, com o qual pretenda estabelecer uma comunicação. Por isso, grande parte das comunicações IP estabelecidas são executadas por uma consulta (query) DNS.

Para entender como estas três classes de servidores interagem, suponhamos que um cliente DNS queria determinar o IP endereço para o nome do host www.amazon.com. Primeiramente, ocorrem os seguintes eventos. O cliente primeiro contacta um dos servidores raiz, que retorna o endereço IP dos servidores TLD (Top para o domínio de nível superior .com. De seguida, o cliente contacta um dos servidores TLD, que irá retornar o endereço IP de um servidor autorizado para amazon.com. Finalmente, o cliente entra em contacto um dos servidores autorizados de amazon.com, que irá retornar o endereço IP do nome dos host www.amazon.com.

Numa primeira análise, existem três classes de servidores DNS (Root, Domínios de Topo e Servidores Autoritativos) que podem ser organizados numa hierarquia tal como identificado na Figura 1.

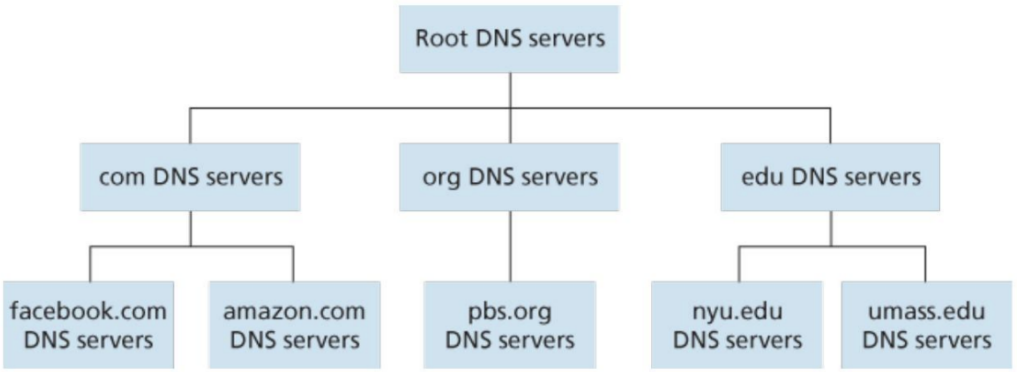


Figura 1: Parte da hierarquia dos servidores DNS (*Domain Name Space)*

Tal como referido anteriormente, existem três classes de servidores DNS:

- **Root DNS Servers (ST)**: Contactam o servidor DNS autoritativo se o mapeamento do nome não for conhecido. De seguida, obtém o mapeamento - nome, Servidor, DNS - e retorna esse mapeamento ao servidor de nomes locais. São responsáveis por fornecer os endereços IP dos servidores TLD;

- **Domínios de topo (SDT)**: Para cada um dos domínios de nível superior, como .com, .org, .net, .edu e .gov, e todos os domínios de nível superior do país, como pt, uk, fr, ca e jp, existe um servidor TLD (ou cluster de servidor);
- **Servidores DNS autoritativos**: São servidores DNS das organizações, com autoridade sobre um domínio de nomes local e sobre os mapeamentos nome/endereço dessa organização. Pode ser gerido pela própria organização ou pelo seu ISP (Internet Service Provider).

Requisitos Funcionais

Tendo em conta o sistema DNS da Internet, podemos identificar quatro tipos de elementos fundamentais que podem interagir: Servidor Primário (SP), Servidor Secundário (SS), Servidor de Resolução (SR) e Cliente (CL) que podem ser descritos da seguinte forma.

1. Servidor Primário

O **Servidor Primário** (SP) é o único servidor que possui acesso direto ao ficheiro de base de dados, portanto é o único capaz de oferecer uma cópia da mesma. Este servidor tem acesso à sua configuração em específico onde estão especificadas várias informações sobre o seu domínio como, a localização do seu ficheiro de base de dados, identificação dos restantes servidores do domínio em questão, localização dos ficheiros log e endereço dos servidores de topo.

2. Servidor Secundário

O **Servidor Secundário** (SS) é semelhante ao servidor primário, a única diferença é que apresenta uma cópia da base de dados do SP correspondente, que é guardado em memória volátil. Este servidor é o único que faz o pedido da base de dados, caracterizado como transferência de zona, ao servidor primário. Para além disso, o SS tem acesso à sua configuração em específico onde estão especificadas várias informações sobre o seu domínio, isto é, portas de atendimento, identificação dos SP dos domínios para os quais é SS, identificação do ficheiro log, o endereço dos servidores de topo.

3. Servidor Primário versus Servidor Secundário

Numa configuração de DNS, o SP é quem vai conter o arquivo onde está contida toda a informação relevante para um domínio, como por exemplo o número de IP. Ou seja, quando é colocado o domínio, o SP é quem vai buscar as informações para se fazer a tradução diretamente no arquivo local.

Já o SS contém uma cópia da base de dados, que é fornecida pelo SP através da operação denominada por Transferência de Zona.

4. Transferência de Zona (SP e SS)

A transferência de zona é uma interação do tipo TCP, que consiste na comunicação entre um Servidor Secundário e Servidor Primário para verificar se a base de dados está atualizada, ou até mesmo, criar a cópia da base de dados em questão para o SS. Todo o processo é explicado mais em detalhe no tópico 3. Modelo de Comunicação.

5. Servidor de Resolução

O **Servidor de Resolução** (SR) faz papel de intermediário, logo não possui nenhuma autoridade sobre nenhum do domínio. Apenas serve a rede local e as aplicações da rede local (ex: máquina local ou até dos processos da máquina local). Este tipo de servidor tem apenas na sua constituição uma cache. Ele utiliza as mesmas mensagens e responde como se fosse um primário e/ou secundário em termos de PDU, ou seja, só responde quando lhe pedirem. Este servidor tem que ter acesso à sua configuração em específico onde são apresentadas várias informações sobre o seu domínio, como restantes servidores do domínio em questão, localização dos ficheiros log e dos servidores de topo.

6. Cliente

O **Cliente** (CL) é o processo que necessita da informação da base de dados de DNS de um determinado domínio - por exemplo, a aplicação de browser, um cliente de e-mail - mas apenas para pedir informação, isto é, vai processar a resposta mas não vai conseguir enviar. O cliente obtém a informação realizando as queries DNS a um SR que está contida num ficheiro de configuração - endereço IP e portas de atendimento - ou então usa o SR do sistema operativo.

> **Diferença entre Cliente e Servidor:** os servidores não tem nenhuma interface com o utilizador, a única forma de arrancar um SP ou SS é indicando qual é o seu respetivo ficheiro de configuração.

7. Todos os Componentes

	SP	SS	SR	CL
Input	- Ficheiro de configuração - Ficheiro de base de dados para cada domínio gerido - Ficheiro com a lista de ST	- Ficheiro de configuração -Ficheiro com a lista de ST	- Ficheiro de configuração - Ficheiro com a lista de ST	Linha de comando (não há necessidade de ficheiro de configuração)
Output	- Ficheiro log			

Tabela 1: Resumo dos componentes utilizados

2. Modelo de Informação

No contexto do projeto são necessários ficheiros de configuração, para que cada um dos componentes do sistema saiba qual o comportamento que deve adotar, de base de dados, onde é incluída toda a informação dos seus dados, de log, em que deve ser registrada toda a atividade do elemento em questão, e com a lista de ST. A sua representação e descrição é apresentada de seguida.

Em contexto do nosso trabalho, podem ser observados exemplos de todos os ficheiros abaixo explicados nos Anexos.

Ficheiro de configuração dos servidores SP, SS e SR

Estes ficheiros apresentam todas as regras que os diversos servidores devem aplicar na execução deste sistema. A estrutura de cada uma das entradas é:

```
{parâmetro} {tipo de valor} {valor associado ao parâmetro}
```

O parâmetro diz respeito ao domínio em questão, para o tipo de valor existem diversas possibilidades (enumeradas de seguida) e, os valor associados ao parâmetro variam conforme o valor que se trata, podendo ser caminhos de localização de ficheiros de base de dados, log ou de root, tal como endereços associados a um servidor, etc.

Podem ser aceites os seguintes tipos de valor:

Tipo	Descrição
DB	Indica o ficheiro da base de dados.
SP	Indica o endereço IP[:porta] do SP - qual é o endereço do Servidor Primário para aquele mesmo domínio. Não aparece esta linha no ficheiro de configuração de um SP.
SS	Indica ao SP quais são os Servidores Secundários que têm acesso - sendo um secundário por cada linha. Não aparece esta linha no ficheiro de configuração de um SS.
DD	Indica o endereço de um dos servidores que deve ser tomado como servidor por defeito desse mesmo domínio em específico.
ST	Indica a localização do ficheiro com uma lista dos Servidores de Topo.

LG	Indica a localização do ficheiro log, onde é registrada toda a atividade do servidor associado ao seu domínio.

Tabela 2: Tipos de valores aceites num ficheiro de configuração

Na figura 3,4 e 5 presente nos anexos, é possível observar uma imagem do ficheiro de configuração de um SP, SS e SR, respectivamente.

Ficheiro com a lista de ST (Servidores de Topo)

Estes ficheiros devem ser acedidos quando se requer informação adicional sobre os domínios acima do domínio em questão e, estes dados não constam em cache. Apenas é apresentada uma lista com os Servidores de Topo em que as suas entradas são simplesmente um endereço IP/porta um por linha de um ficheiro. A estrutura de cada uma das entradas é a seguinte:

```
{endereço IP[:porta]}
```

Na figura 7 presente nos anexos, é possível observar uma imagem do ficheiro com a lista dos ST, que apresenta o endereço IP do STFairy.

Ficheiros de log

Os ficheiros de log são responsáveis por armazenar toda a atividade envolvente no servidor em questão. Isto é, sempre que um componente arranca, este deve verificar a existência dos ficheiros log que encontram-se indicados no seu ficheiro de configuração. Se não existirem ficheiros log, este deve ser criado, e se já existirem novas entradas, estas devem ser registadas a partir da última entrada existente no ficheiro.

Para cada linha de ficheiro, deve existir uma entrada de log, de modo a que, sempre que um componente arrancar, este poder verificar a existência

As entradas destes ficheiros devem seguir a seguinte sintaxe:

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

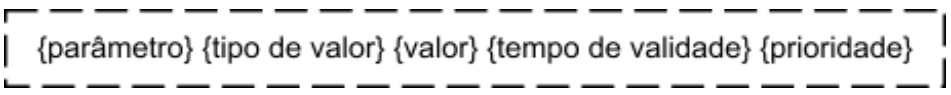
A etiqueta temporal é usada para sinalizar a data do sistema de quando houve atividade, as diversas opções para tipo de entrada, que estão enumeradas maia abaixo, o endereço IP - serve para localizar a operação que foi efetuada e, por fim, os dados de entrada que variam consoante o tipo de entrada.

Tipo	Descrição
QR/QE	É 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	Indica a informação que é apresentada na resposta à query.
ZT	Indica quando é feita uma transferência de zona e de que servidor se trata, SP ou SS - registrando se foi bem ou mal sucedido.
EV	Indica se houve algum evento interno no componente - nos dados de entrada é indicada informação adicional sobre o acontecimento.
ER	Indica que foi recebido um PDU do endereço indicado que não foi possível decodificar corretamente.
EZ	Indica que foi detectado um erro num processo de transferência de zona que não foi concluída corretamente - os dados de entrada mencionam qual o servidor em questão.
FL	Foi detectado um erro no funcionamento interno do componente.
TO	Quando foi detectada um timeout na interação com o servidor no endereço indicado - é indicada qual o tipo de timeout, por exemplo, se é resposta a query, (início de uma transferência de zona).
SP	Indica que a execução do componente foi parada.
ST	Indica que a execução do componente foi iniciada.

Tabela 3: Tipos de entrada aceites num ficheiro de log

Ficheiro de dados do SP (Servidor Primário)

O ficheiro de dados apresenta toda a informação relativa aos dados. De notar uma sintaxe na qual o SP deve saber reportar:



O parâmetro diz respeito ao domínio em questão e os tipos de valor são os que estão representados na tabela x. O campo do tempo de validade (TTL) especifica o tempo máximo em segundos que os dados podem existir dentro da cache de um servidor. Quando o TTL não é suportado num determinado tipo, o seu valor é igual a zero.

O campo da prioridade define uma ordem de prioridade, quanto menor o valor maior a prioridade. Para os parâmetros que possuem um único valor ou para parâmetros nos quais os valores têm a mesma prioridade, o campo não deve existir.

Os nomes completos de emails, dominios, servidores e hosts, todos devem terminar com um '.' (por exemplo, o nome completo do domínio presente na Figura 6 : *winx.*). Contudo, quando os nomes não terminam com '.' admite-se que são concatenados com um prefixo definido através do parâmetro '@' do tipo *DEFAULT.*

Na seguinte tabela podemos observar os diferentes tipos de valores a suportar (os tipos marcados com '*' são de implementação opcional e os tipos de valores que devem suportar o campo da prioridade são indicados explicitamente):

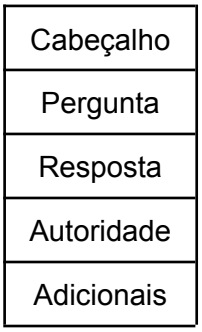
Tipo	Descrição
DEFAULT*	Identifica um prefixo por defeito que é acrescentado sempre que um nome não apareça na forma completa.
SOASP	Indica o nome completo do SP do domínio indicado no parâmetro.
SOAADMIN	Indica o endereço de e-mail completo do administrador do domínio. Por exemplo, '@' deve ser substituído por um '.'
SOASERIAL	Número de série para esta zona - sempre que a base de dados é alterada este número deve ser incrementado.
SOAREFRESH	Intervalo de tempo em segundos para um SS perguntar ao SP do domínio indicado, qual o nº de série da base de dados dessa zona.
SOARETRY	Número de segundos após o qual o SS deve perguntar novamente a um SP, o número de séries da base de dados dessa zona, se o SP não responder - houver um *timeout*.
SOAEXPIRE	Valor que indica por quanto tempo um SS pode continuar a dar respostas autoritativas depois de não ter mais notícias do SP.
NS	Authoritative Name Server - Servidor original responsável por resolver esta zona e por manter os registos originais da mesma.
A	Endereço de uma máquina (IPv4).
CNAME	Indica um nome canónico (ou alias) que está associado ao nome indicado no parâmetro.
MX	Devolve o nome de um servidor de e-mail para o domínio indicado no parâmetro.
PTR	Utilizado em resoluções recursivas (a que domínio está associado este IP).

Tabela 4: Tipos de valores aceites num ficheiro de dados

Na figura 6, presente nos anexos, é possível observar o Ficheiro de Base de Dados do Servidor Primário do domínio .winx.

PDU: Formato das mensagens de DNS

Para todas as operações mencionadas na imagem abaixo, foi definido um protocolo de comunicação que permite a transferência de informação. O DNS usa o mesmo formato de mensagem para todos os tipos de consultas a um cliente e para a resposta a um servidor. O servidor DNS armazena diferentes tipos de registros de recurso utilizados para resolver nomes. Esses registros contêm o nome, endereço e tipo de registro, sendo alguns desses tipo, *A*, *NS*, *CNAME*, que iremos falar mais abaixo.



A mensagem DNS possui um cabeçalho fixo, as restantes secções possuem um comprimento variável.

Mecanismo de Codificação Binária

Na codificação binária de uma mensagem DNS, apenas é necessário a codificação de três tipos de campos: números inteiros positivos, (identificadores, TTL, prioridades, etc.), strings (nomes de domínios, máquinas, serviços) e flags.

Os inteiros são facilmente codificáveis num número certo de bytes, dependendo do intervalo de valores possíveis. As strings são codificadas diretamente numa sequência de símbolos ASCII (todos os nomes que é possível usar nas bases de dados do DNS têm de ser representáveis por códigos ASCII). As flags podem ser facilmente codificadas em bits específicos dum byte. Os separadores dos campos na codificação binária são desnecessários. Também não é preciso qualquer mecanismo de compressão, encriptação ou verificação da integridade das mensagens de DNS.

3. Modelo de Comunicação

As interações possíveis deste sistema são efetuadas por mensagens do tipo DNS encapsuladas no protocolo UDP, ou seja, toda a comunicação que envolve um servidor e um cliente tem por base uma ligação do tipo UDP. A unidade de dados protocolar a ser usada por este tipo de mensagens é o PDU, de modo a que seja possível descodificar as mensagens em causa.

As mensagens DNS têm uma representação lógica que inclui o cabeçalho, este com tamanho fixo, e uma parte de dados limitada até 1KByte. A constituição do cabeçalho é apresentada na tabela 1.

	Representação	**Descrição**
Message ID	entre 1 e 65535	Identificador da mensagem
Flags	Q, R ou A	Identifica se é uma query ou uma resposta, processo recursivo ou iterativo e resposta autoritativa
Response Code	0, 1, 2 ou 3	Indica se há erro na resposta a uma query e qual o erro em específico **(*)**
Number of Values	entre 0 e 255	Identifica o número de entradas relevantes que estão classificadas no campo Response Values
Number of Authorities	entre 0 e 255	Caracteriza o número de entradas para servidores autoritativos
Number of Extra Values	entre 0 e 255	Define o número de entradas com dados adicionais relativos a resultados da query ou servidores autoritativos

Tabela 5: Campos do cabeçalho da mensagem DNS

(*) Situações de erro:

- → 0: não existe erro e a resposta tem informação direta à query, a resposta deve ser guardada na cache;
- → 1: o domínio em NAME existe mas não foi identificada informação sobre ele que corresponde ao campo TYPE OF VALUE, a resposta é negativa e pode ser guardada em cache;
- → 2: o domínio em NAME não existe, a resposta é negativa e pode ser guardada em cache;
- → 3: a mensagem DNS não foi descodificada corretamente.

Já o campo de dados pode ser caracterizado como apresentado na tabela 2.

	Descrição
Query Info	Associado aos dados originais da query **(**)**
Response Values	Identifica as entrada correspondentes aos parâmetros da query guardados na cache ou base de dados do servidor autoritativo
Authorities Values	Identifica as entrada correspondentes aos parâmetros da query guardados na cache ou base de dados do servidor autoritativo
Extra Values	Caracteriza as entradas do servidor autoritativos

Tabela 6: Campos de dados da mensagem DNS

Em suma, pode-se verificar que o campo de dados é representado pelos dados duma query original, pelas respostas dadas a essa mesma query, informação sobre os servidores autoritativos e, por fim, informação extra indiretamente relacionada com a query original.

★ **Como é efetuada a comunicação?**

O sistema faz a sua comunicação através do envio de queries numa mensagem DNS tendo como destino, sempre, um certo Servidor. A origem desta mesma query pode ser qualquer um dos Servidores presentes neste contexto ou até mesmo um Cliente.

Considerando as queries em questão, pode se representar a sua informação pelos seguintes campos **(**)**:

- ➔ Name;
- ➔ Type of Value.

Na mensagem da query vão ser usados apenas alguns campos referentes ao cabeçalho e aos dados da mensagem de DNS e ao campo relativo à informação da query sendo eles, **Message ID, Flags, Query Info, Name, Type of Value**. Os restantes campos não são considerados ou sequer incluídos.

Quando a query é recebida por um Servidor, este deve descodificar a sua informação, verificando a sua exatidão, e, posteriormente, procurar a resposta necessária. A informação para a resposta pode estar armazenada na cache ou na base de dados. Se for iniciada a procura na cache e esta não estiver registada então, o Servidor envia uma query para um Servidor de Domínio de Topo em que esteja incluída a entrada que procurava inicialmente.

Transferência de Zona

Nestes casos, a conexão da interação é do tipo TCP. Um exemplo destas interações é a comunicação entre um SS e um SP para verificar se a sua base de dados está atualizada e caso não esteja poder atualizar essa mesma.

Em concreto, o SS envia ao SP correspondente o nome do domínio para onde espera receber a cópia da base de dados do outro. Depois de estabelecida a conexão e de confirmada a veracidade do domínio e a permissão do SS para efetuar a cópia do ficheiro de base de dados, o SP envia o número de entradas do seu ficheiro de base de dados, com um valor máximo de 65535.

Caso o SS envie o mesmo número de entradas retornando ao SP, significando que concordou, já é possível o envio dessas mesmas entradas no formato texto, por ordem crescente.

Até o tempo terminar, o SS vai analisando todas as entradas recebidas. Quando o tempo predefinido chegar ao fim, o SS tem que acabar com a conexão TCP e com a transferência de zona. Só após um tempo igual a SOARETRY é que pode voltar a tentar estabelecer uma outra conexão.

Funcionamento do CL

Quanto ao CL, não é usado nenhum tipo de ficheiro que contribua para o seu funcionamento. Enquanto os outros servidores deste sistema necessitam de ficheiros de configuração, de base de dados e de log para registar as atividade e respostas das suas queries, o input do CL tem origem na linha de comandos ou na interface da aplicação. Já o output é fornecido através da interface de output ou da aplicação.

É neste caso que há uma interação entre o utilizador e o sistema em que o utilizador pode ter acesso à informação sobre o sistema em questão.

A aplicação CL necessita de saber, em qualquer momento, qual o servidor que deve usar como destino para as suas queries. Também vai recorrer à informação, obtida pelo utilizador, no campo Query Info (Name e o Type of Value). Opcionalmente, o utilizador pode dizer se a query é recursiva porque por defeito, o CL utiliza queries em modo iterativo, não ativando a flag R.

Após ter recolhido a informação, o CL pode adotar um comportamento como se fosse uma aplicação normal ou um SR, no entanto não deve implementar cache.

Este exercício deve executar, por defeito, em modo debug sendo que, as mensagens DNS não estão em binário.

Sistemas de Cache

Deve ser implementado um sistema de cache em que as respostas disponibilizam os resultados pedidos, isto é, caching positivo. Há maior necessidade de implementação de um sistema de cache nos servidores não autoritativos, como o SR.

O objetivo é um sistema de caching positivo em memória volátil (igual para todos os servidores) e o registo de informação na cache só deve ser feito depois da resposta recebida pela query estar processada.

A linha correspondente aos campos de entrada de informação num sistema cache estão apresentados na tabela 3.

Name	Parâmetro
Type	Tipo de valor
Value	Valor
TTL	Tempo de Validade
Order	Prioridade
Origin	Origem da entrada (FILE, SP ou OTHERS)
TimeStamp	Tempo desde o arranque do servidor até a entrada ser registada na cache (em segundos)
Index	Número da entrada, incluindo as entradas FREE (1 até N)
Status	Estado da entrada (FREE ou VALID)

Tabela 7: Estrutura de uma entrada de informação na cache

4. Planeamento do Ambiente de Teste

Na Topologia do Core foram implementados dois servidores de topo, **STFairy** que faz referência às fadas e o **STHuman**, que faz referência ao mundo dos Humanos. Existem também três domínios de topo, que fazem referência à sua escola, portanto, dentro de cada um dos domínios é possível identificar de que servidores se trata, pois o nome de cada uma das personagem é inicializado com o tipo de servidor que se trata (SP, SS, SR). Para além disso, a cada domínio de topo, é atribuído o seu respectivo subdomínio, que tal como nos domínios o nome é inicializado com o tipo de servidor que se trata e neste caso, com o nome do poder da personagem. Por exemplo, um dos Servidores Primários presente no domínio .winx, está denominado por **SPBloom**, e dentro do subdomínio .winx, está presente **SPFire,** que é um servidor primário.

A topologia com este componentes pode ser observada em Anexo apresentando, também, os seus detalhes.

Ainda na implementação do Core, houve a necessidade de implementar os Servidores do mesmo domínio em sub-redes diferentes por questões de segurança e fiabilidade. Deste modo, caso haja algum problema num deles, há sempre uma alternativa para aceder à informação que for precisa. De maneira a conectar os Servidores de Topo e os Servidores de Domínio de Topo, foram usados routers que permitiram a partilha de informação. Os switchers também foram necessários para a adição de novas sub-redes do mesmo domínio.

Também se acrescenta que o domínio de topo nomeado .reverse é constituído pelos domínios de DNS reverso. O reverse faz uma consulta de DNS para o domínio que está associado a um determinado endereço de IP. Neste caso, faz-se o oposto da pesquisa de DNS, em que se consulta o sistema de DNS para que este retorne um endereço de IP.

Tendo em conta os componentes no ambiente de teste, foram criados também os seus ficheiros de configuração de cada um dos servidores instalados, os ficheiros de base de dados e, também, os ficheiros com a lista de ST do sistema. O ficheiro de configuração usado no caso para todos os servidores (SP, SS e SR), que fazem input do mesmo, sendo este ficheiro indicado com o próprio nome.

Ao contrário do ficheiro anterior, nos ficheiros de base de dados não foi necessário para todos os servidores, porque apenas o servidor primário é que faz input do mesmo ‘para cada um dos domínios'.

Por fim, o ficheiro com a lista de ST é apenas um, contendo o ip dois Servidores de Topo por cada linha do ficheiro, sendo estes distinguidos/diferenciados através do seu IP.

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

O programa é capaz de executar o servidor através da classe **Server**, tanto o Servidor Primário como o Servidor Secundário. É nesta classe que é inicializado o servidor, após a inserção da informação necessária para o mesmo através da linha de comandos. Os dados essenciais para o arranque do servidor são:

- o valor da porta de atendimento;
- a localização do ficheiro de configuração do mesmo.

Também foi debatido a inclusão do parâmetro de timeout e de debug mas por impossibilidade de implementação desta parte, não foi incluído no seu input.

Desta forma, através do ficheiro de configuração, conseguimos ter acesso às suas especificações. Da maneira que é feita a leitura do ficheiro de configuração, vão sendo guardados todos os dados relativos ao servidor em questão como, o seu domínio, qual o tipo de servidor (SP ou SS), a localização dos ficheiros de base de dados (apenas se for SP) e a localização dos ficheiros log.

Considerando um SP, há a necessidade de criação de uma estrutura de dados que armazene todos os dados, inclusive os que estão presentes no ficheiro de base de dados, a cache. Todos os dados relativos à cache seguem a estrutura apresentada na classe **ResourceRecord**. Assim, podemos definir a cache como uma lista de linhas do tipo **ResourceRecord**.

Posto isto, o servidor está apto para comunicação com o cliente que apresente o mesmo valor para a porta de atendimento. Esta comunicação é feita através do método comunicUDP em que é estabelecida a conexão entre os dois componentes via UDP. Esta ligação é apresentada com sockets que permitem enviar e receber pacotes de dados.

O servidor cria um datagrama para poder receber o pedido efetuado pelo cliente. Todos os pacotes de dados transportados pelo socket são recebidos com uma formação binário, desta forma há mais facilidade de transmissão dos mesmos.

Na classe **DNSMessage** é possível observar que os diferentes tipos de construtores permitem, por exemplo, a receção de uma query do tipo buffer de bytes e, atribuir-lhe assim os diferentes parâmetros. Este tipo de transformação é explicada mais em detalhe em 6. CL em modo debug.

Durante todo o processo, é feita a escrita dos acontecimentos nos ficheiros de log. Cada linha de um ficheiro de log corresponde a um envio/receção de uma query e estes estão devidamente identificados por uma etiqueta temporal.

Feita assim a receção do pedido do cliente, é necessário construir a resposta ao mesmo, isto pode ser observado no método buildResponse. Este método recebe uma query, já transformada para o tipo de **DNSMessage** e é procurada a resposta a esta query na cache do servidor em questão. O método searchCache faz com que sejam percorridas todas as linhas da cache e sejam armazenados os índices das linhas da cache onde dê match com o pedido. Por exemplo, se o nome da query e o tipo dela foram iguais ao nome e ao tipo da linha cache, então, vai ser armazenado na lista de elementos “RV” o índice da linha de cache em questão. Posto isto, é construída a mensagem a ser enviada ao cliente pelas informações obtidas.

De maneira semelhante à receção de um pedido, a resposta ao mesmo também é enviada em formato de bytes e, posteriormente, o cliente é que trata da sua tradução.

Agora considerando um SS, é necessário pedir a transferência de zona via TCP. Para este fim decidiu-se criar uma classe **SS** que permite a criação de uma conexão entre o SP e o SS.

Esta operação permite ao SS ter a versão mais atualizada do SP do seu domínio. Para ir de encontro a este objetivo, decidiu-se obter o domínio do SS e procurar qual o SP desse mesmo domínio. Desta forma garante-se que estamos a procurar pelos servidores na mesma rede e não noutras. De modo a efetuar a cópia do ficheiro de base de dados do SP, efetuou-se a leitura desse mesmo ficheiro guardando essa informação na cache do SS. Tal como mencionado acima, todas as linhas da cache vão seguir a estrutura da classe **ResourceRecord**. Este componente não foi totalmente implementado tal como explicado no tópico 7. Correções.

As restantes implementações não foram concretizadas tal como é explicado em 7. Correções.

6. CL em modo debug

O programa é capaz de executar o cliente através da classe **Cliente**. É nesta classe que é inicializado o cliente, após a inserção da informação necessária para o mesmo através da linha de comandos. Os dados essenciais para o arranque do cliente são:

- o valor da porta de atendimento;
- o nome da query;
- o tipo da query.

Num ponto inicial, os dados inseridos no input são transformados num pacote de dados através do construtor **DNSMessage** que permite a receção de apenas o nome e tipo da query. A comunicação entre o cliente e o servidor é feita a partir de sockets em que é enviado/recebido o pacote de dados com formação em binário. Tem que ser garantido que o servidor corresponde ao mesmo valor da porta do cliente.

De seguida, o cliente recebe a resposta construída pelo servidor. Esta resposta construída pelo servidor é transformada para a estrutura de dados do construtor da classe **DNSMessage** onde definimos o número de bytes correspondente a cada campo da mensagem.

Considerando a estrutura das queries, definimos os campos da mensagem atribuindo-lhes o número necessário de bits para a sua compreensão, ou seja, o valor à frente apresentado é o número necessário de bits para a sua representação.

Os campos de cabeçalho de uma mensagem possuem os seguintes parâmetros:

- messageId -> 16 bits
- flags -> 3 bits
- responseCode -> 2 bits
- nValues -> 8 bits
- nAuthorities -> 8 bits
- nExtraValues -> 8 bits

Já os campos de dados podem ser divididos em:

- Query Name
- Query Type
- Response Values= List[nValues]
- Authorities Values= List[nAuthorities]
- Extra Values= List[nExtra]

Neste caso, o comprimento da lista onde os valores de Response/Authorities/Extra Values são guardados depende do número de cada um deles apresentado no cabeçalho (nValues/nAuthorities/nExtraValues).

Cada um destes valores anteriormente descrito define as variáveis globais da classe **DNSMessage**.

Por outro lado, também houve a necessidade da criação de um construtor que permita a receção apenas do nome e tipo da query, onde são atribuídos valores “zero” ou “null” aos restantes atributos.

7. Correções

Infelizmente, não conseguimos atingir todos os objetivos propostos para a segunda fase, visto que tivemos bastantes dificuldades na primeira fase devido à complexidade do projeto e conhecimento na linguagem de Java. Devido a essas implicações nesta fase estivemos um pouco limitados pois tivemos que reformular e corrigir partes do código e relatório ainda da primeira parte.

Desde a entrega da 1ª fase conseguimos melhorar em diversos aspectos, alguns deles sendo, a comunicação entre o cliente e o servidor, a transferência de zona entre o servidor primário e o servidor secundário, aperfeiçoamos Arquitetura de Sistemas e o Modelo de Informação no relatório. No entanto, vamos enumerar algumas das coisas que gostaríamos de ter implementado. Tais como o ST, SDT e SR. Gostaríamos de apresentar estratégias para a implementação destes mesmos componentes mas, devido às complicações encontradas, a prioridade focou-se na correção dos aspetos que não funcionavam numa primeira fase.

Também não conseguimos fazer a ligação ao ambiente de teste no Core onde o objetivo era observar o ambiente de teste em funcionamento com os diversos servidores dos diversos domínios a funcionar entre eles. Neste caso, seria necessário atribuir o papel de cliente a um dos servidores, por exemplo, um dos SR, de modo a que conseguisse comunicar com um dos outros servidores. Assim seria possível visualizar a comunicação entre os domínios existentes.

8. Participação

De seguida é apresentada uma tabela com a participação dos diversos elementos do grupo no trabalho, tanto na 1ª fase como na 2ª fase.

Atividade (1ª fase)	Alexandra (A94523)	Inês (A97372)	Joana (A96584)
Arquitetura do Sistema (relatório)	90%	-	10%
Modelo de Informação (relatório)	50%	10%	40%
Modelo de Comunicação (relatório)	-	-	100%
Planeamento do Ambiente de Teste (relatório)	5%	-	95%
Protótipo SP e SS em modo debug (relatório)	100%	-	-
Protótipo CL em modo debug (relatório)	-	100%	-
Ficheiros de Configuração	-	-	100%
Ficheiros de Base de Dados	15%	80%	5%
Código de parse dos diversos ficheiros	-	-	100%
Implementação da cache	-	-	100%
Código relativo à estrutura da Query	-	-	100%
Código relativo à estrutura do Cliente	5%	95%	-
Código relativo à estrutura Servidor	-	50%	50%
Código construção respostas e procura na cache		-	100%
Código comunicação entre Servidor e Cliente	-	60%	40%
Código relativo ao SP e SS	-	100%	-

Atividade (2ª fase)	Alexandra (A94523)	Inês (A97372)	Joana (A96584)
Arquitetura do Sistema (reformulação da 1ª fase no relatório)	100%	-	-
Modelo de Informação (reformulação da 1ª fase no relatório)	100%	-	-
Implementação de SP,SS,ST,SDT e SR em modo debug (relatório)	-	-	100%
CL em modo debug (relatório)	-	-	100%
Correções (relatório)	-	100%	-
Correção de erros de código	-	-	100%
Código comunicação entre Servidor e Cliente	-	-	100%
Código relativo à Transferência de Zona	40%	60%	-

Conclusão

Este projeto ajudou-nos a desenvolver capacidades sobre DNS, envio de dados por datagramas, implementar cache entre outros.

Em relação ao desenvolvimento do projeto, tanto os aspectos positivos como os negativos, assim como as dificuldades sentidas e superadas, estão mencionadas no tópico 7. Correções.

De uma forma geral, tal como já foi referido, não conseguimos obter o resultado atingir todos os objetivos devido a certos erros no código que não nos permitiram implementar todas as funcionalidades previstas para o projeto, impedindo assim de dar continuidade à implementação do código, no entanto algumas das dificuldades foram ultrapassadas, que mais uma vez, estão presentes no tópico 7. Correções.

Anexos

- **Ambiente de teste**

Neste trabalho foi implementado um ambiente de teste que está representado na figura 1.

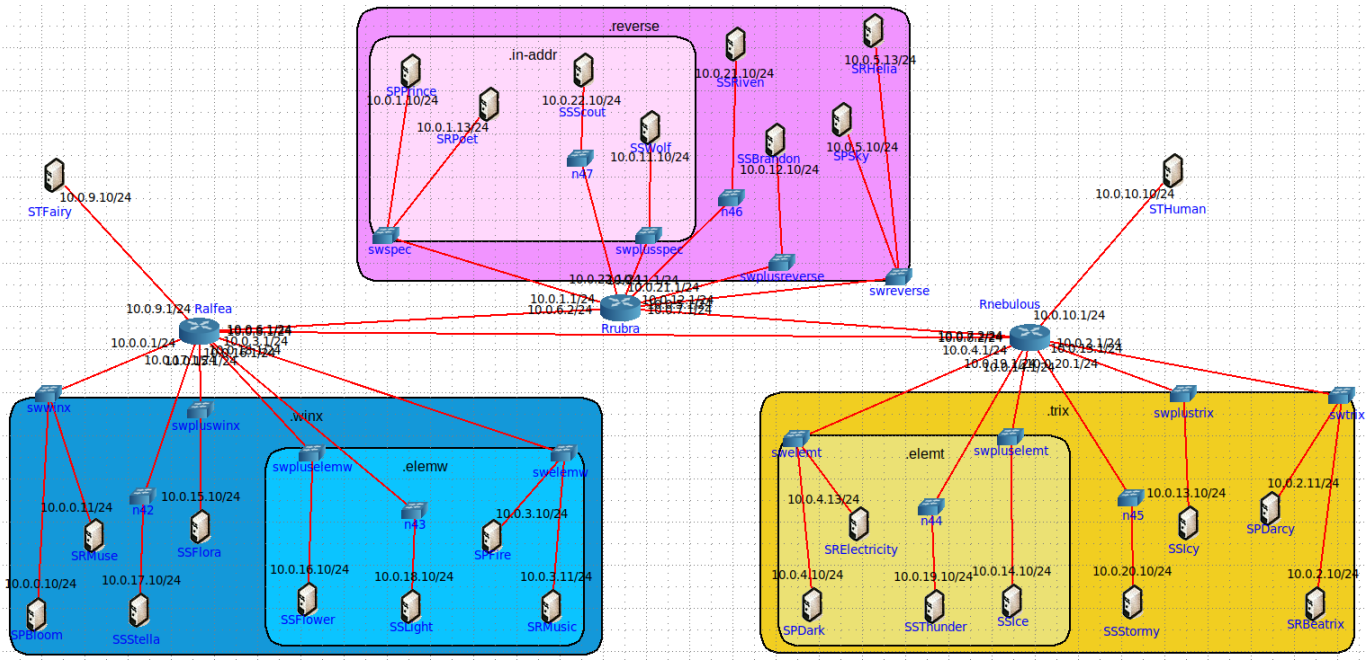


Figura 2: Topologia do Core implementada

No ambiente de teste é possível identificar os dois Servidores de Topo como **STFairy** e **STHuman**.

Para além disto, também foram definidos três domínios de topo (.winx, .trix, .reverse) e os seus respectivos subdomínios (.elemw, .elemt, .in-addr). Em cada um deles era necessária a instalação de Servidores Primários, Servidores Secundários e Servidores de Resolução. Estes elementos podem ser agrupados da seguinte forma:

- **.winx**
  - ★ SPBloom
  - ★ SSStella
  - ★ SSFlora
  - ★ SRMuse

- **.elemw**
  - ★ SSFire
  - ★ SSLight
  - ★ SSFlower
  - ★ SRMusic

- **.trix**
  - ★ SPDarcy
  - ★ SSIcy
  - ★ SSStormy
  - ★ SRBeatrix

  - **.elemt**
    - ★ SPDark
    - ★ SSIce
    - ★ SSThunder
    - ★ SRElectricity

- **.reverse**
  - ★ SPSky
  - ★ SSBrandon
  - ★ SSRiven
  - ★ SRHelia

  - **.in-addr**
    - ★ SPPrince
    - ★ SSScout
    - ★ SSWolf
    - ★ SRPoet

- **Ficheiros**

Todos os SP têm no seu ficheiro de configuração a linha referente à base de dados, os Servidores Secundários, o Domínio por Defeito, a localização dos seus ficheiros de Log e a localização da lista dos ST. Exemplo disto é apresentado na figura 2.

```
# Configuration file for primary server for .winx
.winx DB src/info/data-base-files/Bloom.bd
.winx SS 10.0.17.10
.winx SS 10.0.15.10
.winx DD 10.0.0.10
.winx LG src/logs/winx.log
all LG src/logs/all.log
root ST src/info/root-files/ST.db
```

Figura 3: Ficheiro de Configuração do Servidor Primário do domínio .winx

De seguida, é possível observar a diferença com um ficheiro de configuração de um SS para o mesmo domínio. Neste caso, há a entrada para um SP, sendo este o SP referido anteriormente, tal como observável na figura 3.

```
# Configuration file for secondary server for winx.
winx. SP 10.0.0.10
winx. SS 10.0.17.10
winx. DD 10.0.1.10
.winx LG src/logs/winx.log
all LG src/logs/all.log
root ST src/info/root-files/ST.db
```

Figura 4: Ficheiro de Configuração de um Servidor Secundário do domínio .winx

Em contraste, é possível verificar que os SR apresentam entradas para todos os outros servidores do mesmo domínio. Assim sendo, é apresentado o endereço do SP e dois dois SS do domínio em questão. Isto pode se verificar na figura 5.

```
# Configuration file for resolver server for .winx
.winx SP 10.0.0.10
.winx SS 10.0.17.10
.winx SS 10.0.15.10
.winx DD 10.0.0.10
.winx LG src/logs/winx.log
all LG src/logs/all.log
root ST src/info/root-files/ST.db
```

Figura 5: Ficheiro de Configuração do Servidor de Resolução do domínio .winx

Quanto aos ficheiros de base de dados, apenas os SP os apresentam na sua constituição.

```
# DNS database file for domain .winx
# It also includes a pointer to the primary server
# of the elemw.winx subdomain

@ DEFAULT .winx.
TTL DEFAULT 86400

@ SOASP ns1.winx. TTL
@ SOAADMIN dns\.admin.winx. TTL
@ SOASERIAL 0117102022 TTL
@ SOAREFRESH 14400 TTL
@ SOARETRY 3600 TTL
@ SOAEXPIRE 604800 TTL

@ NS ns1.winx. TTL
@ NS ns2.winx. TTL
@ NS ns3.winx. TTL

elemw.winx. NS sp.elemw.winx. TTL

ns1.winx. A 10.0.0.10 TTL
ns2.winx. A 10.0.0.11 TTL
ns3.winx. A 10.0.0.12 TTL
sp.elemw A 10.0.3.10 TTL

spBloom.winx. CNAME ns1 TTL
ssStella CNAME ns2 TTL
ssFlora CNAME ns3 TTL
```

Figura 6: Ficheiro de Base de Dados do Servidor Primário do domínio .winx

Já os ficheiros com a lista dos ST apresentam apenas uma entrada de identificação do endereço do Servidor em questão, como apresentado na figura 6.

```
#Root file for top server

10.0.9.10
10.0.10.10
```

Figura 7: Ficheiro do Servidor de Topo

Por outro lado, os ficheiros de log podem ser representados da seguinte forma. Neste caso, temos um exemplo de duas linhas (a 2ª linha é incompleta) do nosso ficheiro log para o domínio .winx. Neste caso aplicamos a etiqueta temporal e os dados da query enviada ou recebida pelo servidor.

```
30-12-2022 16:53:33  1 Q 0 0 0 0 winx. NS [] [] []
30-12-2022 16:53:33  1 A 0 3 3 4 .winx. NS [.winx. NS ns1.winx. 86400 6, .winx. NS ns2.winx. 86400 7, .winx. NS ns3.
```

Figura 8: Ficheiro de Log

Bibliografia

Kurose, J. (2016). Computer Networking: A Top-Down Approach (7th edition.)