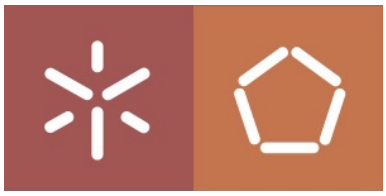

Comunicações por Computador

22/23

TP2: Implementação de Sistema DNS

Grupo 6.02

23 de dezembro de 2022

Patrícia Pereira (A89578) Meriem Khammassi (A85829)

Índice

Lista de Figuras

1 Introdução

Domain Name System é um registo que contém nomes de sites e respetivos a endereços IP associados.

No âmbito da UC de Comunicação por Computadores, foi-nos proposto conceber um Sistema DNS da arquitetura TCP/IP, utilizando os protocolos do nível de transporte UDP e TCP.

Para tal, especificar um PDU e as primitivas de serviço dum protocolo aplicacional utilizando um protocolo de transporte orientado à conexão e um protocolo de transporte não orientado à conexão, seria o nosso segundo objetivo à atingir para uma implementação eficiente.

Assim, o nosso foco inicial foi pensar e desenvolver os componentes fundamentais e ficheiros de configuração do sistema para poderem interagir e obter um modelo de comunicação eficiente que permite o envio de mensagens DNS.

Na perspetiva de atingir todos os objetivos anteriormente mencionados, decidimos desenvolver o nosso projeto em **Java** sendo o desenvolvimento e as decisões tomadas explicados nos restantes tópicos deste relatório.

2 Arquitetura do sistema

O DNS utiliza um esquema de nomes hierárquicos conhecidos como *domain names* sendo o nível mais alto da hierarquia é o *Root Server* que pode responder diretamente a querys sobre registos armazenados em cache na zona da raíz e encaminhar outros pedidos ao servidor *Top Level Domain* apropriado.

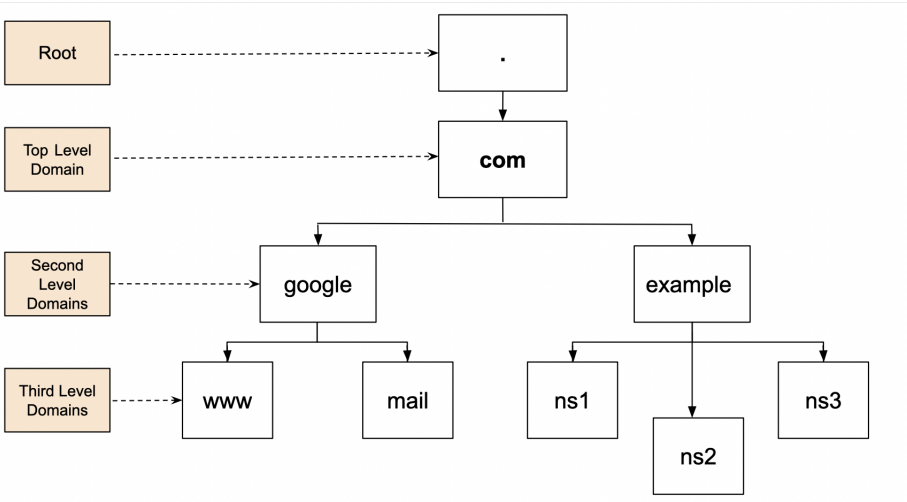

Figura 1: Arquitetura da solução

2.1 Funcionalidades dos elementos do Sistema

Tendo em consideração as normas que especificam o Sistema DNS, podemos identificar quatro primeiros tipos fundamentais e duas variantes especiais de servidores.

- **Cliente** - Aplicação que efetua pedido de informação sobre um domínio e espera pela resposta do servidor.
- **Servidor Resolução** - Intermediário entre o cliente e os restantes Servidores. Explicaremos melhor o funcionamento deste na próxima subsecção.
- **Servidor de Topo** - Reencaminha pedidos para o Domínio de topo apropriado.
- **Servidor de Domínio de topo** - Reencaminha para o Servidor Primário apropriado.
- **Servidor Primário** - Tem a informação sobre o domínio pedido pelo cliente.
- **Servidor Secundário** - Possui uma cópia dos ficheiros de um Servidor Primário, e pode pedir uma atualização da mesma.

2.2 Modelo de Comunicação

O Sistema DNS é um sistema hierárquico e distribuído de gestão de nomes de domínio e subdomínio, funciona em uma infraestrutura de subsistemas, com diferentes servidores a processar informações e transmiti-las uns para os outros, trabalhando na primeira camada com o objetivo de responder aos pedidos recebidos dos clientes.

Por outro lado, é necessária a utilização do protocolo UDP. Este é um protocolo de transporte em tempo real sem conexão que oferece um processo de comunicação mais rápido, o que significa que através da utilização do UDP, pode-se enviar datagramas de uma máquina a outra, mas sem garantia de que os dados enviados chegarão intactos e na ordem correta. Por essa razão, este protocolo é usado para trocar pequenas informações.

A comunicação entre os elementos do sistema pode ser feita de forma iterativa ou recursiva.

2.2.1 Modelo Iterativo

1. O sistema é iniciado quando um cliente pretende aceder a um domínio, i.e. quando saber o endereço IP correspondente ao nome. É assim criada uma query e enviada para o servidor resolução.

   Quando o Servidor de Resolução recebe a query ele acede à sua cache para verificar se tem a resposta:

2. (a) Caso não contenha, reencaminha a query para o próximo nível, o Servidor de Topo (ST), o topo da hierarquia.

(b) Este reencaminha o Servidor de Resolução (SR) para o Domínio de Topo que contem informação sobre o domínio pretendido.

(c) O Servidor de Resolução (SR) envia a mensagem para o Servidor de Domínio de Topo (SDT).

(d) O Servidor de Domínio de Topo (SDT) acede à sua base de dados e, responde ao Servidor de Resolução (SR) qual o servidor autoritativo que contém o IP.

(e) O Servidor de Resolução (SR) envia a query para este servidor e,

(f) Este responde qual o endereço pedido.
Neste momento o Servidor de Resolução (SR) armazena a resposta na sua cache.

3. Tendo a resposta, o Servidor de Resolução (SR) responde ao Cliente.

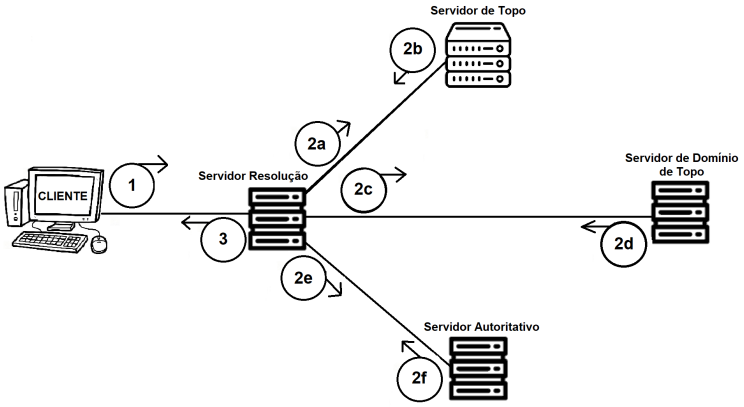

Figura 2: Representação da comunicação Iterativa

2.2.2 Modelo Recursivo

1. Como no modelo anterior, a comunicação inicia-se no cliente. É enviada uma query com a indicação que a comunicação deve ser recursiva.

   Ao receber a query, o Servidor de resolução verifica se tem a resposta em cache:

2. (a) Caso não contenha, reencaminha a query para o próximo nível, o Servidor de Topo (ST).

   (b) Este reencaminha a query para o Servidor de Domínio de Topo (SDT) que contém informação sobre o domínio pretendido.

   (c) O Servidor de Domínio de Topo (SDT) acede à sua base de dados para saber qual o servidor autoritativo que contém o IP e, por sua vez, reencaminha para esse a query.

(d) O Servidor autoritativo responde ao Servidor de Domínio de Topo (SDT) com o IP pedido e esse é reencaminhado pelo caminho inverso até chegar ao Servidor de Resolução (SR).

3. Tendo a resposta o Servidor de Resolução (SR) responde ao Cliente.

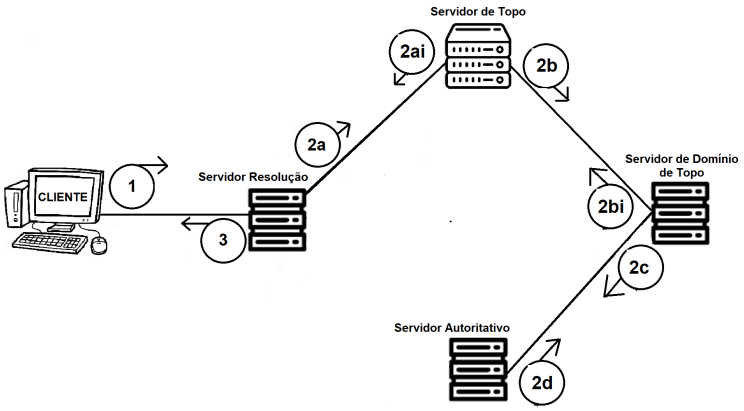

Figura 3: Representação da comunicação Recursiva

2.3 Transferência de Zona

A transferência de zona DNS, é um tipo de transação de DNS. É um dos muitos mecanismos disponíveis para os administradores replicarem bancos de dados DNS em um conjunto de servidores DNS.

O sistema DNS usa o TCP para transferir dados de zona e assume a forma de uma transação cliente-servidor.

O pedido de transferência de zona é feito pelo servidor secundário, solicitando dados ao servidor primário. A parte da base de dados que é replicada é uma zona.

1. As operações de transferência de zona devem ser feitas através de conexão TCP. O Servidor Secundário efetua uma query para saber se a versão da base de dados que possui é a mais recente. Caso não seja, o SS inicia uma tentativa de transferência de zona.

   O pedido contém o nome completo do domínio correspondente à base de dados da qual pretende receber uma cópia.

2. O SP verifica se o domínio é válido e se o SS tem permissão para receber a cópia.

   Se isso se verificar, o SP envia para o SS a quantidade de entradas do ficheiro da base de dados a enviar.

3. O SS aceita a quantidade de entradas e responde ao SP com o mesmo número.

4. Depois de aceite pelo SS, o SP envia as entradas em formato texto, enumerando-as por ordem crescente.

   O SS vai confirmando se recebeu todas as entradas esperadas até um tempo inicialmente definido se esgotar.

5. Quando esse tempo terminar o SS termina a conexão TCP e desiste da transferência de zona.

   Este deve tentar novamente a transferência de zona após um intervalo de tempo igual a SOARETRY. Um SP também não deve aceitar pedidos consecutivos de transferência de zona do mesmo SS com intervalo menor de SOARETRY segundos.

O pedido contém o nome completo do domínio correspondente à base de dados da qual pretende receber uma cópia.

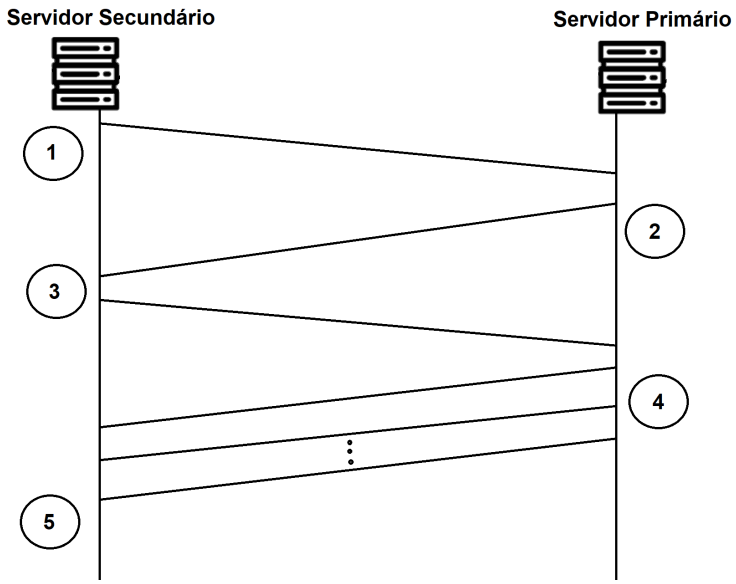

Figura 4: Representação da comunicação Recursiva

3 Modelo comunicativo

Nesta secção iremos apresentar o modelo comunicativo previsto para a fase final deste trabalho prático. Assim iremos apresentar todas as componentes que devem existir, para além das que criamos para esta fase inicial.

Para que o sistema flua como pretendido todos os servidores devem estar em execução e à escuta em pelo menos uma porta.

3.1 Cliente

Quando um cliente é iniciado, é criada uma *DNS message* com o os campos correspondentes à query preenchidos, e as flags, caso existam. Espera ativamente que lhe seja enviada uma resposta e, assim que receba é terminado.

3.2 Servidores

Uma vez que base a todos os servidores uma vez que todos tem o mesmo principio: recebem uma query, interpretam-a, acedem à sua base de dados e enviam outra query, por esta razão temos uma classe *Server*.

As funcionalidades especificas dos outros servidores irão ser implementadas noutras classes reaproveitando o que já foi implementado.

3.3 Parsers

O trabalho prático exige dois ficheiros iniciais, o de configuração e o de base de dados. Por esta razão, foram criadas duas classes para poder processar a informação contida em cada ficheiro.

4 Estratégias de implementação

Para a primeira fase criamos as classes referentes a:

- Definição de todos os campos pertencentes a uma mensagem DNS: É nesta classe onde convertemos a mensagem para um array de Bytes bem como o inverso.
- Um Cliente onde é criada uma primeira query e enviada para um ip recebido por argumento.
- Um Servidor que recebe, interpreta e responde.
- Uma classe que cria uma mensagem DNS formatada como solicitada no enunciado.
- Uma classe que lê o ficheiros de configuração e cria variáveis com as informações relevantes.
- Uma classe semelhante à anterior que lê um ficheiro de base de dados.
- Uma classe que cria um ficheiro de logs e escreve para o mesmo.

4.1 Opções de execução e funcionalidades

Para executar um cliente deve-se executar o ficheiro *.class* e dar como argumento o ip do servidor a quem se deve enviar a query e a porta, o nome do domínio, o tipo de valor e, opcionalmente flags.

> java cliente 192.168.56.102:1234 example.com XP A

Para executar um servidor apenas precisamos de passar como argumento o ficheiro de configuração.

> java server configFile

4.1.1 Requisitos Funcionais

Para esta primeira implementamos as seguintes funcionalidades:
- Criar uma mensagem DNS com todos os campos que esta contém.
- Um cliente eficiente que cria e envia e recebe querys.
- Um servidor que recebe querys as interpreta e responde.

4.2 Paralelização

A necessidade de paralelizar é crucial. Na nossa implementação pensamos necessitar da utilização de threads. A comunicação entre as threads Sender e Receiver, deverá ser assegurada por uma queue, quando o Receiver recebe uma DNS message descompacta-a, processa a informação e cria uma mensagem de resposta para enviar. Esta mensagem será adicionada a uma queue e posteriormente removida pelo Sender, quando este estiver disponível para a enviar.

4.3 Encriptação

No modo normal do funcionamento dos componentes, as mensagens DNS têm que ser codificadas em binário de forma eficiente, precisando de implementar encriptação nas nossas mensagens DNS. Para isso, criamos duas funções, uma para encriptar e outra para desencriptar. Nestas funções utilizamos as classes de java Cipher e SecretKeySpec para nos auxiliar na encriptação. Ambas as funções recebem e devolvem um array de bytes.

Após a criação das mesmas, no fim da execução da função ToByteArray invocaremos o método responsável por encriptar. Enquanto que no início do construtor DNSmessage, invocaremos o método responsável pela desencriptação. Garantindo que as mensagens convertidas em arrays de bytes para transmissão contêm a mensagem encriptada.

4.4 Ficheiros de Log

Para o ficheiro de Logs criamos uma classe onde existe um método que deverá ser usado pelos servidores. Este método recebe o nome do ficheiro e uma string que escreve na próxima linha.

5 Resultados

Para obter os resultados utilizamos como ficheiro de configuração e base de dados os exemplos fornecidos no enunciado.

```
# Header
MESSAGE-ID = 56135, FLAGS = A, RESPONSE-CODE = 0, N-VALUES = 2, N-AUTHORITIES = 3, N-EXTRA = 9,
# Data: Query Info
QUERY-INFO.NAME = example.com, QUERY-INFO.TYPE = MX,;
# Data: list os Response, Authorities and Extra Values
RESPONSE-VALUES = example.com. MX mx1.example.com 86400,
RESPONSE-VALUES = example.com. MX mx2.example.com 86400,
AUTHORITIES-VALUES = example.com. NS ns1.example.com. 86400,
AUTHORITIES-VALUES = example.com. NS ns2.example.com. 86400,
AUTHORITIES-VALUES = example.com. NS ns3.example.com. 86400,
EXTRA-VALUES = ns1example.com. A 193.136.130.250 86400,
EXTRA-VALUES = ns2example.com. A 193.137.100.250 86400,
EXTRA-VALUES = ns3example.com. A 193.136.130.251 86400,
EXTRA-VALUES = sp.smallerexample.com. A 193.140.90.11 86400,
EXTRA-VALUES = mx1example.com. A 193.136.130.200 86400,
EXTRA-VALUES = mx2example.com. A 193.136.130.201 86400,
EXTRA-VALUES = wwwexample.com. A 193.136.130.80 86400,
EXTRA-VALUES = wwwexample.com. A 193.136.130.81 86400,
EXTRA-VALUES = ftpexample.com. A 193.136.130.20 86400,
```

Figura 5: Resposta recebida por um cliente

6 Tabela de Atividades

Na tabela a baixo está o planeamento para ambas as fases. Na primeira fase pensamos ter conseguido seguir o planeado anteriormente no entanto a implementação dos métodos que transformam uma mensagem DNs em um array de bytes e a função contrária ocuparam-mos mais tempo que o que tínhamos inicialmente previsto.

1ª Fase		2ª Fase	
Planeamento e análise do relatório	20/10 a 3/11	Implementar ficheiros de log	25/11 a 28/11
Definir elementos do sistema	20/10 a 3/11	Implementar Servidor Pirmário Secundário e Resolução	28/11 a 07/12
Definir comunicação entre os sitemas	03/11 a 6/11	Transferência de zona	07/12 a 16/12
Implementação de um Cliente e Servidor	06/11 a 9/11	Implementar sistema Iterativo e Recursivo	07/12 a 16/12
Implementação de uma mesnagem DNS	09/11 a 10/11	Sistema de cache	16/12 a 23/12
Transformar mensagem DNS num array de bytes e voltar a passar para o formato de mensagem	11/10 a 25/11	Implementação de threads, Tempo de validade, prioridade, backup de serviços e hosts	26/12 a 29/12
Parsers dos ficheiros de configuração e base de dados	13/11 a 20/11	Implementação do Ambiente de Teste	29/12 a 2/12
Envio de uma mensagens DNS com o formato correto	20/11 a 25/11	Entrega	23-Feb

Figura 6: Tabela de Atividade

7 Conclusão

Este trabalho prático revelou-se um grande desafio no sentido de termos de enfrentar um "Universo"completamente novo para nós e que nos deixou um pouco desorientados inicialmente.

Durante a implementação deparamo-nos com vários problemas e dificuldades que, fomos resolvendo e tomando decisões na orientação do estado final da primeira fase deste projeto.

Assim sendo, o trabalho prático revelou-se de extrema importância na assimilação e aperfeiçoamento dos conceitos lecionados ao longo das aulas e, ainda, no desenvolvimento de conceitos extra-aula que fomos adquirindo com a pesquisa extensa durante o desenvolvimento. Permitiu-nos, também, um aperfeiçoamento nas skills da linguagem de programação Java e na utilização de um emulador core, ferramenta esta que será útil no futuro com certeza.

Em suma, tendo em conta esta primeira parte do trabalho realizado, estamos satisfeitas e orgulhosas com o resultado obtido, e consideramos ir de encontro aquilo que foi solicitado, procurando melhorar vários aspetos e completar todos os requisitos para obter um trabalho final eficiente.