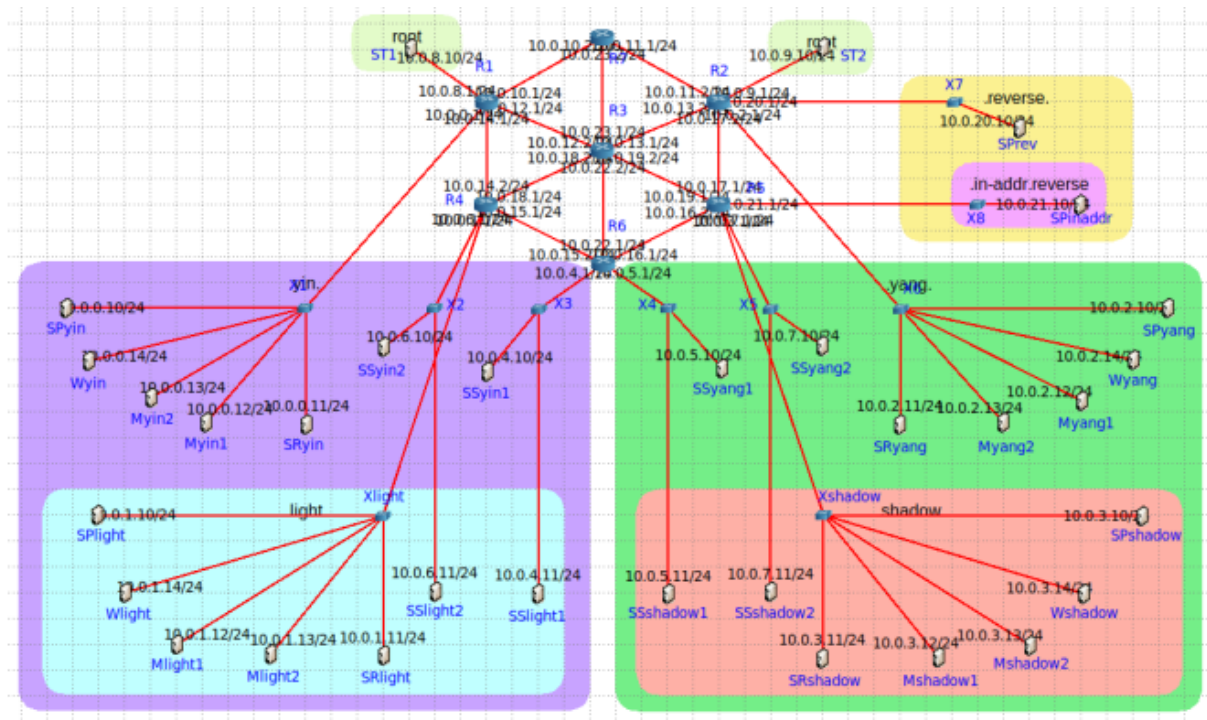


Universidade do Minho

Comunicações por Computador

Trabalho Prático 2 - PL7 - Grupo 1

Implementação de Sistema DNS



Ana Rita Moreira Vaz (a97540)

Robert Benjamin Szabo (a91682)

22/11/2022

Índice

	Página
Introdução	3
Arquitetura do Sistema	4
Modelo de Comunicação.....	8
Modelo de Informação.....	21
Ambiente de Teste.....	30
Possíveis Situações de Erro	33
Tabela de Atividades.....	35
Conclusão	36
Anexos.....	37

Introdução

Tal como os humanos podem ser identificados de diversas formas, os hosts podem ser identificados através de hostnames, como por exemplo `www.google.com` ou `www.facebook.com`. Hostnames são mnemônicas apreciadas pelos humanos, no entanto, fornecem pouca informação da localização do host na internet. Este problema é resolvido através da identificação dos hosts com endereços IP.

Assim, pode-se diferenciar duas preferências distintas: os humanos preferem hostnames enquanto os routers preferem endereços IP.

Assim surge o DNS (Domain Name System). O DNS é uma base de dados distribuída que implementa uma hierarquia de servidores DNS e um protocolo de camada de aplicação que permite que os hosts consultem os bancos de dados distribuídos.

Arquitetura do Sistema

Para solucionar o problema de escala, os servidores DNS são organizados da forma como é possível observar na Figura 1.

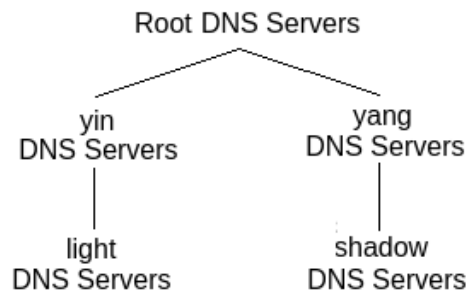


Fig 1. Porção da hierarquia dos servidores DNS

A seguir iremos exemplificar o seguimento de um pedido de um cliente. Primeiro, o cliente contacta um dos servidores root (existem milhares de servidores root distribuídos mundialmente configurados na zona raiz do DNS como 13 autoridades nomeadas). O root server retorna ao cliente o endereço IP dos servidores SDT (explicitado mais à frente). O cliente contacta um dos servidores autoritativos, que retorna um endereço IP para o hostname final.

Em resumo, nesta arquitetura é possível identificar 4 tipos de elementos fundamentais que podem interagir: Servidor Primário (SP), Servidor Secundário (SS), Servidor de Resolução (SR) e Cliente (CL). De realçar que um componente de software pode executar em simultâneo mais do que um tipo de servidor.

Para além destes tipos fundamentais de elementos, existem duas variantes especiais de servidores: os Servidores de Topo (ST, ou DNS root servers) e os Servidores de Domínios de Topo (SDT, ou DNS Top-Level Domain servers).

Neste projeto, o comportamento dos SDT é considerado como sendo igual aos SP ou aos SS. Assim, um SP ou SS autoritativos para um domínio de topo é um SDT, ainda que não tenham domínios hierarquicamente acima na árvore DNS.

Por outro lado, os ST são considerados como sendo SP, mas têm apenas uma base de dados em que, para cada domínio de topo, inclui informação dos SDT respetivos. Portanto, só podem responder com estes dois tipos de informação. No contexto deste projeto, tanto os ST como os SDT são implementados com o mesmo componente que implementa um SP ou um SS.

A seguir estão referidas as funcionalidades de cada elemento neste trabalho, começando pelas globais:

- Qualquer componente de software desenvolvido pode ter parâmetros de funcionamento que sejam passados por argumentos na altura de arranque. Assim,

os componentes têm, no mínimo, três parâmetros de funcionamento: um para indicar a porta de atendimento dos servidores quando esta for diferente da porta normalizada (53), outro que indica o valor de timeout quando se espera pela resposta a uma query e um terceiro que especifica se funciona em modo debug ou não.

- Um componente que funcione em modo debug deve enviar toda a atividade registrada nos logs também para o standard output.
- Todos os tipos de servidores devem possuir a funcionalidade de cache.
- Todos os componentes do sistema implementam processos idênticos de leitura de ficheiros de configuração e de processos comunicacionais assíncronos utilizando queries e respostas a queries.
- Todos os elementos limitam o seu input a ficheiros de configuração ou de dados e o seu output a ficheiros de log ou stdout.

SP

- Os servidores SP respondem a, e efetuam, queries DNS.
- Têm acesso direto à base de dados dum domínio DNS, sendo a autoridade que o gere.
- Qualquer atualização necessária à informação dum domínio DNS é realizada diretamente na base de dados do SP.
- Para além dos dados respeitantes diretamente ao domínio DNS, um SP tem acesso a informação de configuração específica (domínios para os quais é SP, portas de atendimento, identificação dos ficheiros das bases de dados, identificação do ficheiro de log, informação de segurança para acesso às bases de dados, identificação dos SS respetivos e dos SP dos subdomínios, endereços dos servidores de topo, etc.).
- Um servidor SP tem como input um ficheiro de configuração, um ficheiro de base de dados por cada domínio gerido, e um ficheiro com a lista de servidores de topo.
- Como output tem um ficheiro de log ou, se funcionar em modo debug, o stdout.
- Um servidor ST tem apenas uma base de dados em que, para cada domínio de topo, inclui informação dos SDT respetivos.

SS

- SS responde a, e efetua, queries DNS.
- Tem autorização e autoridade para possuir (e manter atualizada) uma réplica da base de dados original do SP autoritativo dum domínio DNS, armazenada apenas em memória volátil.
- Um SS tem 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, endereços dos servidores de topo, etc.).
- Um SS tem como input um ficheiro de configuração e um ficheiro com a lista de servidores de topo.
- Como output tem um ficheiro de log ou, se funcionar em modo debug, o stdout.

SR

- Um servidor SR responde a, e efetua, queries DNS sobre qualquer domínio, mas não tem autoridade sobre nenhum pois serve apenas de intermediário.
- Um SR também é conhecido como local DNS server, DNS resolver, DNS cache-only server e DNS forwarder.
- Os SR funcionam em cascata, dependendo da gestão da configuração dos clientes, dos provedores de serviços e das instituições.
- Um SR tem de ter acesso a informação de configuração específica (eventuais domínios por defeito e lista dos servidores DNS que deve contactar, porta de atendimento, identificação do ficheiro de log, endereços dos servidores de topo, etc.).
- Um SR tem como input um ficheiro de configuração e um ficheiro com a lista de servidores de topo.
- Como output tem um ficheiro de log ou, se funcionar em modo debug, o stdout.

CL

- Um CL obtém informação realizando queries DNS a um SR (de uma lista de SR predefinida).
- Um CL tem uma lista de SR num ficheiro de configuração (endereços IP e portas de atendimento) ou usa o SR do próprio sistema operativo.
- O input e output do CL será através da linha de comando sem necessidade de um ficheiro de configuração.
- Não tem ficheiros de configuração, de dados ou de logs.
- O input deve vir totalmente da linha de comandos ou da interface da própria aplicação.
- O output deve ser feito totalmente para a interface de standard output ou para a interface da própria aplicação.
- Este componente é o único que suporta uma interação normal com um utilizador humano, permitindo assim a esse utilizador aceder diretamente à informação do sistema DNS desenvolvido, instalado e configurado.
- Uma aplicação CL servirá para indagar um serviço DNS.
- Em primeiro lugar, o CL precisa de saber, assim que arranca e em todos os momentos, que servidor DNS usar como destino das suas queries. Esta informação é o primeiro parâmetro passado como argumento na forma dum endereço IP[:porta].
- Como o CL usa queries DNS normais, tem de obter do utilizador a informação que tem de utilizar no campo QUERY INFO numa query DNS, i.e., o nome completo do parâmetro NAME e o tipo de valor esperado TYPE OF VALUE. Opcionalmente, o utilizador pode indicar se pretende tentar o tipo de query recursiva (correspondente à ativação da flag R).
- Por defeito, o CL utiliza queries no modo iterativo e não ativa a flag R.
- Com as informações recolhidas pelo utilizador, o CL deve executar uma query tal como se fosse uma aplicação cliente normal ou um SR.
- O CL não deve implementar nenhum mecanismo de cache.
- Os resultados da query devem ser mostrados ao utilizador utilizando uma sintaxe/formatação semelhante à utilizada nas entradas dos ficheiros de log.

- O CL deve, por defeito, executar em modo debug, isto é., as mensagens de DNS não são codificadas em binário e são transmitidas no formato de sequência de caracteres.
- Numa implementação opcional, o CL pode funcionar em modo normal e utilizar a codificação binária das mensagens DNS que tenha sido definida para os outros elementos/componentes do sistema.

Na figura seguinte (Fig.2) podemos observar os diferentes componentes e funcionalidades que cada servidor implementa. Tanto o SP, o SS, como o SR têm as funcionalidades de Servidor mas implementam também funcionalidades próprias. Todos os servidores recebem e respondem a perguntas (usando UDP) fazendo uma pesquisa na cache, construindo a resposta e registando toda a atividade nos logs. Todos fazem parsing do ficheiro de configuração mas com pequenas alterações de acordo com as entradas que o ficheiro contém. O SP e o SS possuem a capacidade de estabelecer uma ligação TCP entre eles para efeito da transferência da base de dados do SP para o SS.

O cliente é independente destes e possui apenas a capacidade de fazer uma pergunta e obter uma resposta sobre domínios.

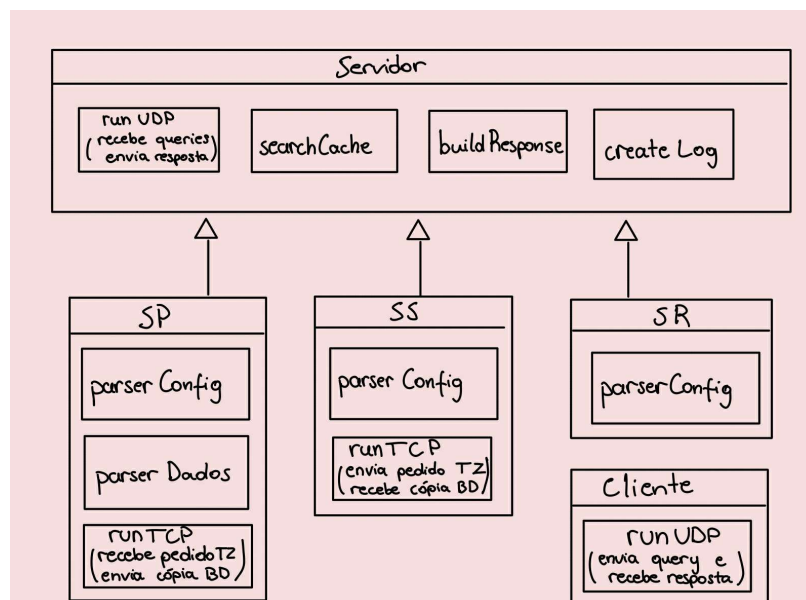


Fig 2. Arquitetura base do sistema

Sistemas de Cache

- Todos os servidores implementam um sistema de cache em memória volátil.
- O caching é positivo quando as respostas disponibilizam os resultados pedidos.
- O caching é negativo quando as respostas informam que uma determinada informação não existe.
- O registo de informação em cache só é efetuado depois de se processar uma resposta recebida a uma query feita anteriormente.
- O envio de queries não está relacionado com nenhuma atividade de caching.
- Os CL não precisam de qualquer tipo de cache.

Modelo de Comunicação

Todas as interações assíncronas (não orientadas à conexão) possíveis neste sistema são realizadas através de mensagens DNS encapsuladas no protocolo UDP. As mensagens DNS são implementadas usando a sintaxe da mesma unidade de dados protocolar (PDU).

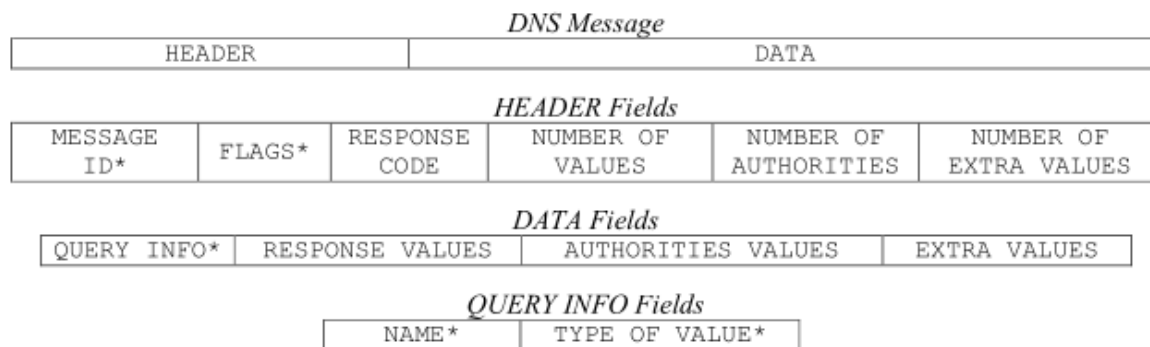


Fig. 3. Representação lógica da Mensagem de DNS

Neste projeto, consideramos que uma mensagem DNS tem um cabeçalho de tamanho fixo de 45 bits, e uma parte de dados que deve ocupar até 1 KByte.

Campos de Cabeçalho

A seguir serão especificados os campos de cabeçalho:

- **MESSAGE ID** (2 bytes): Identificador de mensagem (número inteiro entre 1 e 65535, gerado aleatoriamente pelo CL ou pelo servidor que faz a query original) que irá ser usado para relacionar as respostas recebidas com a query original;
- **FLAGS** (3 bits): Suporta as flags Q, R e A;
 - A flag Q ativa (xx1) indica que a mensagem é uma query, senão é uma resposta a uma query;
 - Se a flag R estiver ativa (x1x) na query indica que se deseja que o processo opere de forma recursiva e não iterativa (que é o modo por defeito);
 - Se a flag R estiver ativa (x1x) na resposta indica que o servidor que respondeu suporta o modo recursivo;
 - Se a flag A estiver ativa (1xx) na resposta indica que a resposta é autoritativa (o valor da flag A é ignorada nas queries originais);

- **RESPONSE CODE** (2 bits): Indica o código de erro na resposta a uma query.
 - Se o valor for zero então não existe qualquer tipo de erro e a resposta contém informação que responde diretamente à query.
 - A resposta é guardada em cache
 - Se houver erros, o sistema suporta os seguintes códigos de erro:
 - 1: O domínio incluído em NAME existe mas não foi encontrada qualquer informação direta com um tipo de valor igual a TYPE OF VALUE
 - O campo de resultados vem vazio
 - O campo com a lista de valores de autoridades válidas para o domínio e o campo com a lista de valores com informação extra podem ser incluídos
 - Este caso é identificado como resposta negativa e pode ser guardada em cache para que a um pedido semelhante, num horizonte temporal curto, o servidor possa responder mais rapidamente;
 - 2: O domínio incluído em NAME não existe
 - O campo de resultados vem vazio
 - O campo com a lista de valores de autoridades válidas onde a resposta foi obtida e o campo com a lista de valores com informação extra podem ser incluídos
 - Este caso é identificado como resposta negativa e pode ser guardada em cache;
 - 3: A mensagem DNS não foi decodificada corretamente;
- **NUMBER OF VALUES** (1 byte): Número de entradas relevantes (máximo de 255) que respondem diretamente à query (ou seja, as entradas na cache ou na base de dados do servidor autoritativo e que têm um parâmetro igual a NAME e um tipo de valor igual a TYPE OF VALUE) e que fazem parte da lista de entradas incluídas no campo RESPONSE VALUES;
- **NUMBER OF AUTHORITIES** (1 byte): Número de entradas (máximo de 255) que identificam os servidores autoritativos para o domínio incluído no RESULT VALUES;
- **NUMBER OF EXTRA VALUES** (1 byte): Número de entradas (máximo de 255) com informação adicional relacionada com os resultados da query ou com os servidores da lista de autoridades;

Campos de Dados

A parte de dados das mensagens DNS contém sempre quatro partes distintas:

- I. Os dados de uma query original;
- II. Os resultados diretos a essa query original;
- III. Informação sobre os servidores que têm informação autoritativa sobre os dados da resposta
- IV. Informação adicional indiretamente ligada aos resultados ou aos dados da query original.

A seguir serão especificados os campos de dados:

- **QUERY INFO:** Informação do parâmetro da query (NAME) e o tipo de valor associado ao parâmetro (TYPE OF VALUE);
 - Os tipos suportados são os mesmos suportados na sintaxe dos ficheiros de base de dados dos SP;
 - Na resposta a queries, os servidores devem copiar a informação do QUERY INFO e incluí-la na mensagem de resposta;
- **RESPONSE VALUES:** Lista das entradas que fazem match no NAME e TYPE OF VALUE incluídos na cache ou na base de dados do servidor autoritativo;
 - Cada entrada deve ter a informação completa tal como é definida na base de dados DNS do SP do domínio referente ao NAME;
- **AUTHORITIES VALUES:** Lista das entradas que fazem match com o NAME e com o tipo de valor igual a NS incluídos na cache ou na base de dados do servidor autoritativo;
 - Cada entrada deve ter a informação completa tal como é definida na base de dados DNS do SP do domínio referente ao NAME;
- **EXTRA VALUES:** Lista das entradas do tipo A (incluídos na cache ou na base de dados do servidor autoritativo)
 - Entradas fazem match no parâmetro com todos os valores no campo RESPONSE VALUES e no campo AUTHORITIES VALUES de forma a que o elemento que o CL ou servidor que recebe a resposta não tenha que fazer novas queries para saber os endereços IP dos parâmetros que vêm como valores nos outros dois campos;
 - Cada entrada deve ter a informação completa tal como é definida na base de dados DNS do SP do domínio referente ao NAME.

Envio/Receção de Queries

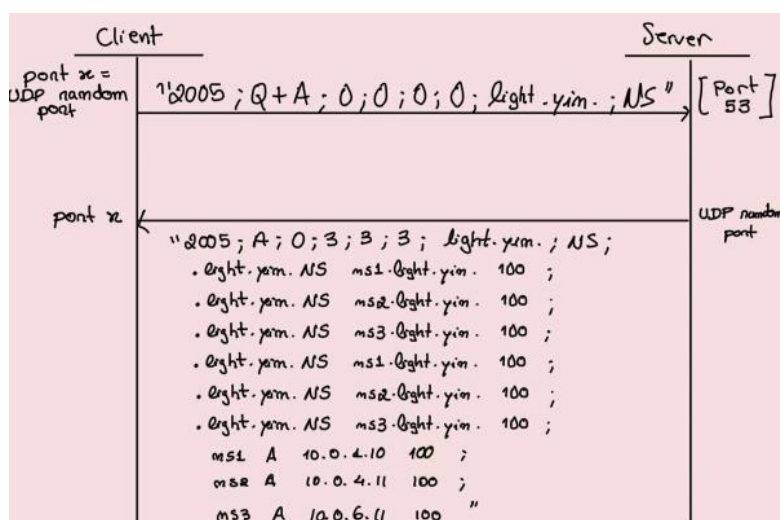


Fig.4 Exemplo de envio e resposta de Query

Todas as interações começam com o envio de uma query DNS para um servidor. Essa query vem dum CL ou dum outro qualquer servidor DNS. A query é transportada numa mensagem DNS.

Na mensagem da query só são usados os campos assinalados com “*”, presentes na figura 2. Os restantes campos do cabeçalho são ignorados (campos colocados a zero) e os restantes campos dos dados são nulos (não são incluídos) na mensagem.

Um servidor deve processar a query recebida.

Se a decodificação da informação da query for correta, o servidor deve tentar encontrar informação direta que responda à query:

1. Na cache.
2. Na base de dados (se for um servidor autoritativo para o domínio do NAME).

Se o servidor não encontrar resposta direta à query (ou seja, não encontrou na cache uma entrada com NAME e valores do tipo TYPE OF VALUE), então deve reenviar a query para um SDT que seja o servidor do domínio de topo incluído no NAME (ou tem a informação desse SDT em cache ou tem de a obter enviando uma query a um ST).

Tipos de Funcionamento

A seguir será apresentada a sequência de acontecimentos ao realizar o pedido DNS pelo host *shiva.di.uminho.pt* a pretender o endereço IP *degaia.cs.umass.edu*, nos dois modos possíveis: recursivo ou iterativo.

No processo recursivo, o servidor coloca o fardo da resolução no servidor de nomes contactado, tal como se pode observar na figura 4:

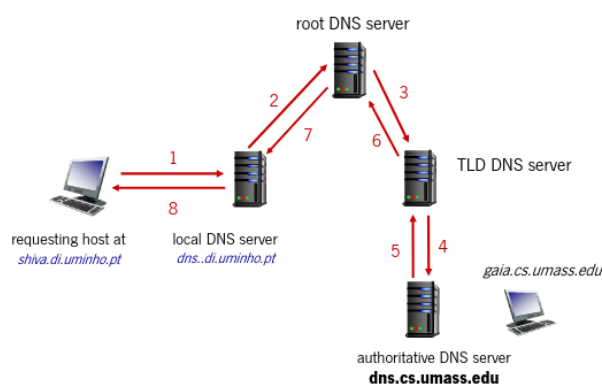


Fig. 5. Funcionamento modo recursivo

No processo iterativo, o servidor contactado responde com o nome do servidor a contactar, tal como é possível observar na figura 5:

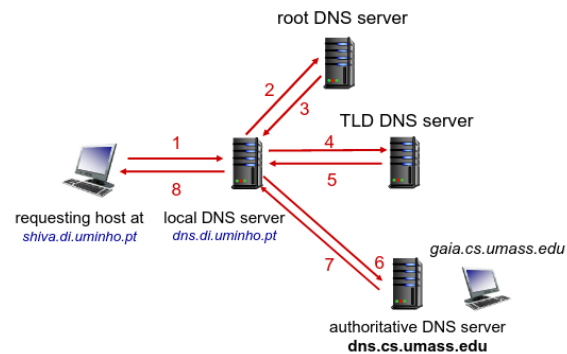


Fig. 6. Funcionamento modo iterativo

O processo continua, de uma forma iterativa ou recursiva, até o servidor obter uma resposta final, isto é, uma resposta em que já obteve informação direta à query ou uma resposta em que, a partir dali, não é possível obter mais informações diretas sobre a query.

Modo de Funcionamento

Para além disso, existem dois modos de funcionamento dos componentes:

- **Modo Normal:** As mensagens DNS têm que ser codificadas em binário, de uma forma eficiente, para uma mensagem ocupar o menor espaço possível. As entradas nos ficheiros de log devem continuar a utilizar o mesmo formato do modo debug.
- **Modo Debug:** As mensagens DNS a ser transmitidas são uma sequência de caracteres (string). Estas sequências devem ser também usadas nas entradas dos ficheiros de log sempre que for necessário registar o envio ou a receção de queries e de respostas.

Transferência de Zona

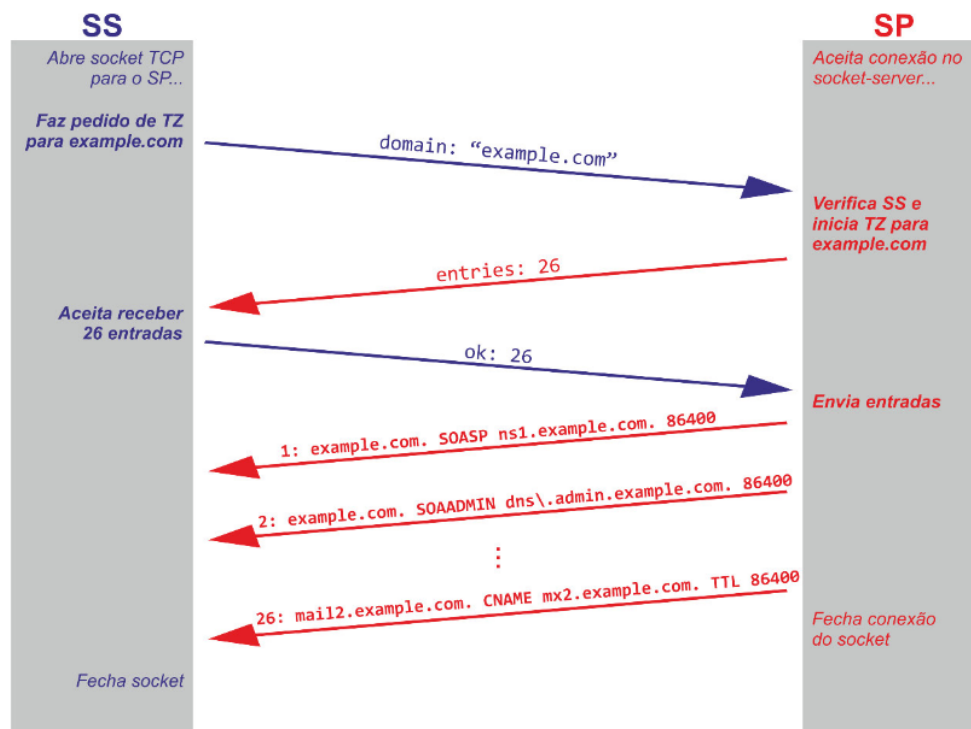


Fig. 7. Exemplo de Transferência de Zona

Existe apenas uma situação em que as interações são realizadas utilizando uma conexão TCP, nas Transferências de Zona, ou seja, quando os SS fazem uma cópia da base de dados dos SP.

Normalmente, um SS utiliza queries normais para saber se a versão da base de dados do respetivo SP é mais atual que a sua. Se for, o SS inicia uma tentativa de transferência de zona enviando o nome completo do domínio para o qual quer receber uma cópia da base de dados do SP. O SP verifica a validade do domínio e que aquele SS tem autorização para receber a cópia da sua base de dados. Se for esse o caso, o SP envia o número de entradas do ficheiro de base de dados (valor máximo de 65535; as linhas de comentário e linhas em branco não são contadas nem transmitidas).

Depois do SS aceitar receber essa quantidade de entradas (respondendo ao SP com o mesmo valor) o SP pode enviar todas entradas em formato de texto, tal como estão no ficheiro de base de dados, mas numerando-as por ordem crescente (acrescenta um número de ordem da entrada antes da entrada propriamente dita).

O SS vai verificando se recebeu todas as entradas esperadas até um tempo predefinido se esgotar. Quando esse tempo terminar o SS termina a conexão TCP e desiste da transferência de zona, sendo que deve tentar outra vez após um intervalo de tempo igual a SOARETRY.

Neste projeto, os SS apenas realizarão transferência de zona quando são inicializados.

Sequência de Acontecimentos Componentes

A seguir serão apresentadas as cronologias de acontecimentos que acontecem durante a execução dos componentes:

Funcionamento do Cliente

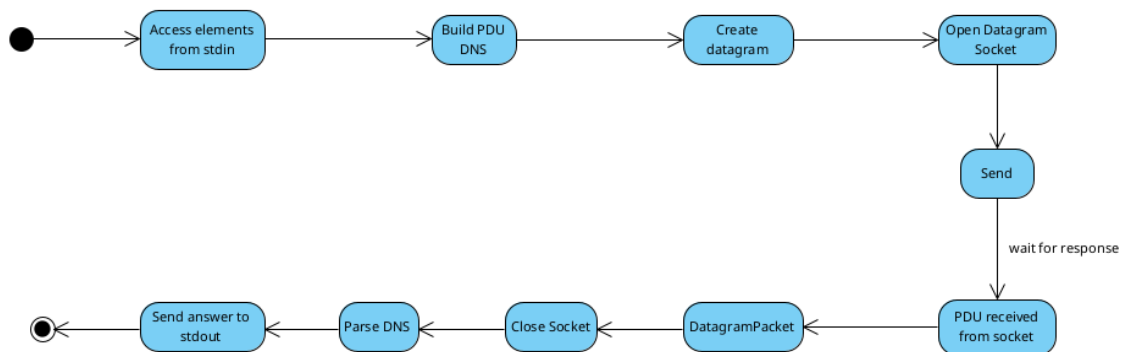


Fig. 8. Diagrama de Atividades do Cliente

Tal como se pode observar através da Fig. 8, o cliente, após ler a query pretendida proveniente do terminal (a porta do servidor, o IP do servidor, o timeout quando se espera pela resposta, indicação se o componente opera em modo debug e a query DNS com os respectivos campos pretendidos), retira a informação referente à query DNS.

Depois é criado o datagrama (junção do IP e porta do servidor com a query DNS) e aberto o socket. O Socket é um mecanismo de comunicação, usado normalmente para implementar um modelo cliente/servidor, que permite a troca de mensagens entre os processos de uma máquina/aplicação servidor e de uma máquina/aplicação cliente. Para a criação de um socket é necessário o IP e a porta do servidor (fornecidas através dos argumentos), ou seja, utiliza-se a camada 3 (Camada IP) e camada 4 (Camada de Transporte) para identificar o socket.

Após a abertura do socket é possível enviar o datagrama e o cliente fica, então, à espera da resposta. Assim que a recebe, retira os dados do datagrama (neste momento em bytes) e fecha o socket (considera-se neste exemplo que o cliente apenas fará uma query). Após a finalização deste processo, é tratada a informação recebida e enviada para o stdout do cliente.

Funcionamento do Servidor

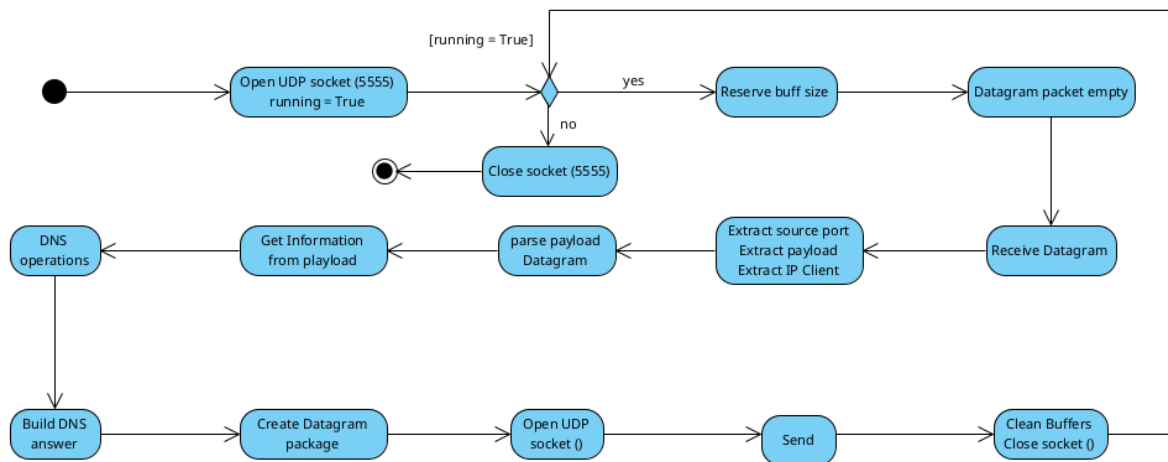


Fig. 9. Diagrama de Atividades do Servidor

Na Fig. 9 é possível observar o processo cronológico nos servidores.

Primeiro é aberto um UDP socket com o número de porta do servidor fixa. É colocado como verdade um booleano que indica se o servidor está ativo.

Enquanto essa variável for verificada, acontecem um conjunto de acontecimentos sequenciais. Primeiro é reservado um buffer com um tamanho fixo e criado um datagrama vazio. Assim que recebe um datagrama, extrai a porta, o IP e o payload do datagrama recebido. É realizado o parse do payload e obtida a informação necessária para a realização da query do cliente. Após a procura da resposta à query, na cache ou na base de dados, é construída a resposta DNS. É encapsulada, criando assim o datagrama que será enviado através de um novo socket. Não se utiliza o mesmo socket para enviar as respostas que o socket aberto anteriormente uma vez que aquele socket deve estar sempre aberto para receber novas queries. O datagrama é enviado, os buffers limpos e os sockets fechados.

Caso a condição já não se verifique, é fechado o socket inicial, terminando assim a execução no servidor.

Funcionamento do Componente udpComm

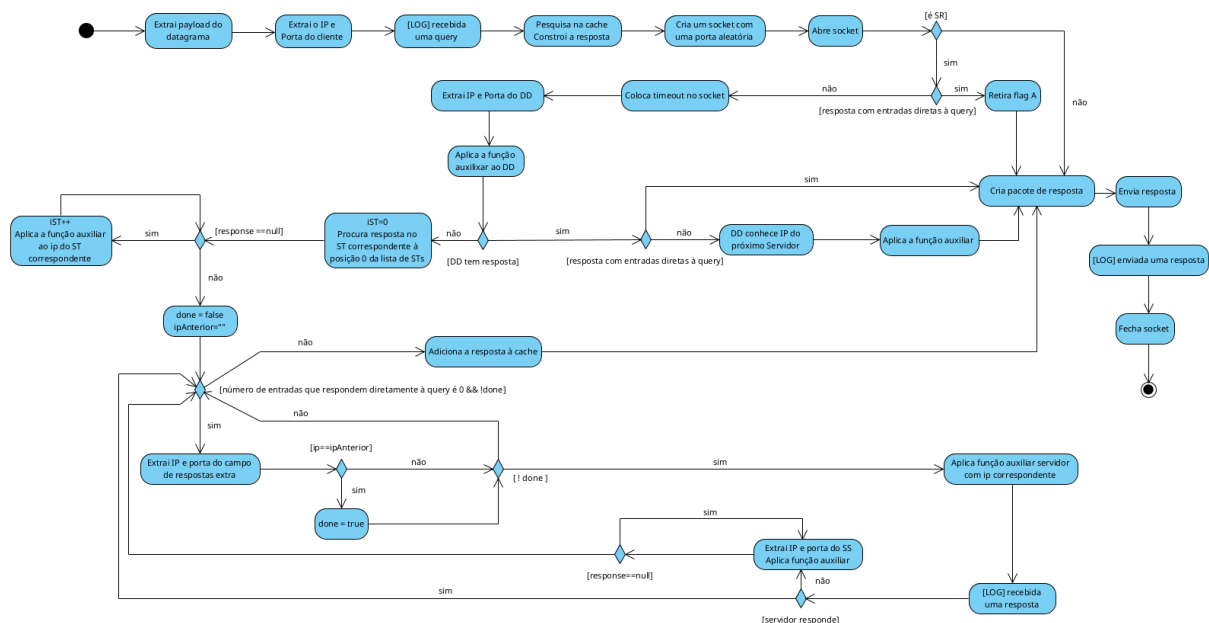


Fig. 10 Funcionamento da função de criação da Resposta

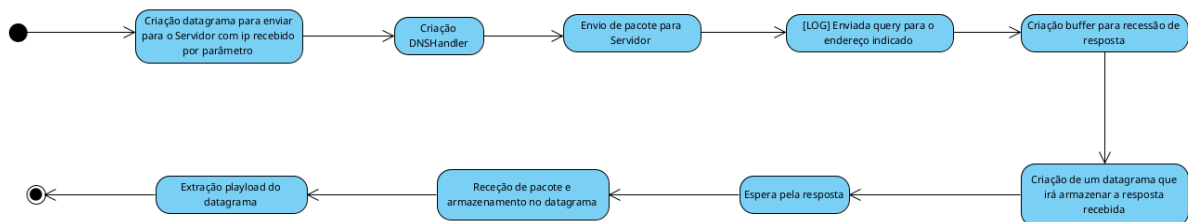


Fig. 11. Função Auxiliar

Quando um servidor recebe um pacote, começa por retirar o payload do diagrama, cria um log a indicar que uma query foi recebida, pesquisa na cache se tem informação acerca dessa query e cria e abre um socket para enviar a resposta.

A partir desse momento existem 3 possibilidades:

- Ou o servidor é um SR e tem entradas diretas à query na cache;
- Ou o servidor é um SR e não tem entradas diretas à query na cache;
- Ou não é um servidor SR

No final de qualquer uma destas possibilidades é criado o pacote de resposta, enviado pelo socket, criado o respetivo log a indicar que uma resposta foi enviada, e o socket é fechado.

Se a primeira opção é verificada, retira-se à resposta a flag A, uma vez que esta já não será uma resposta autoritativa.

Caso a segunda opção se verifique, é colocado um timeout no socket, extraída a porta e ip do DD e aplicada a função auxiliar ao DD. Verifica-se se a resposta recebida do DD tem respostas diretas à query. Caso se verifique, é criado o pacote e realizados os passos já descritos na última fase. Caso não se verifique, começa-se a procurar a resposta no ST. É realizado um ciclo que percorre os ST enquanto não há resposta dos STs anteriores, sendo que se admite que pelo menos um dos ST da lista de ST está ativo. Assim que existe uma resposta, verifica-se novamente se essa resposta contém respostas diretas à query. Caso tenha, são realizados os passos da última fase, caso contrário, é extraído o IP e a porta do campo de extra values da resposta do ST.

Verifica-se se o servidor que será contactado já realizou a query para que não sejam criados ciclos caso não haja resposta em nenhum servidor, quando a máquina em si não existe, por exemplo. Se ainda não foi contactado, é aplicada a função auxiliar ao servidor com o ip correspondente, criado um datagrama que armazena a resposta e é criado um log que indica que uma resposta foi recebida. Se o servidor não responder, é percorrida a lista de servidores secundários, encontrada nos valores extra da resposta, até encontrar uma resposta. Caso a resposta tenha respostas diretas à query é adicionada à cache e são realizados os passos da última fase. Senão, são percorridos os passos descritos neste parágrafo, sendo que, é sempre utilizado o ip extraído do campo de respostas extras da resposta recebida anteriormente.

Na função auxiliar, é criado um datagrama e o DNSHandler com o ip recebido e envia-se o pacote para o servidor. É criado um log com indicação que foi enviada uma query e um buffer e datagrama para a receção da resposta. Depois, espera pela resposta e, assim que a recebe, armazena no datagrama e extrai o payload que envia como resultado desta função auxiliar.

Caso seja verificada a terceira opção, são automaticamente realizados os últimos passos descritos anteriormente.

Funcionamento do Componente tcpComm

Quando o servidor primário inicializa este inicia uma thread que fica sempre à espera de pedidos de transferência de zona provenientes de servidores secundários com permissão (do mesmo domínio).

Assim que os servidores secundários iniciam é realizada a primeira transferência de zona. Estes guardam a versão da base de dados atual e depois estão sempre a fazer questionar o SP pela versão atual dele, até se tornar diferente e, por isso, ocorrer outra transferência de zona.

De notar que a transferência de Zona teria um acréscimo de funcionamento ao utilizar, no servidor primário, os atributos de SOAREFRESH, SOARETRY e SOAEXPIRE da base de dados, que devido a falta de tempo não foi possível implementar.

Por isso, neste momento, o funcionamento do componente tcpComm pode ser descritos pelos diagramas.

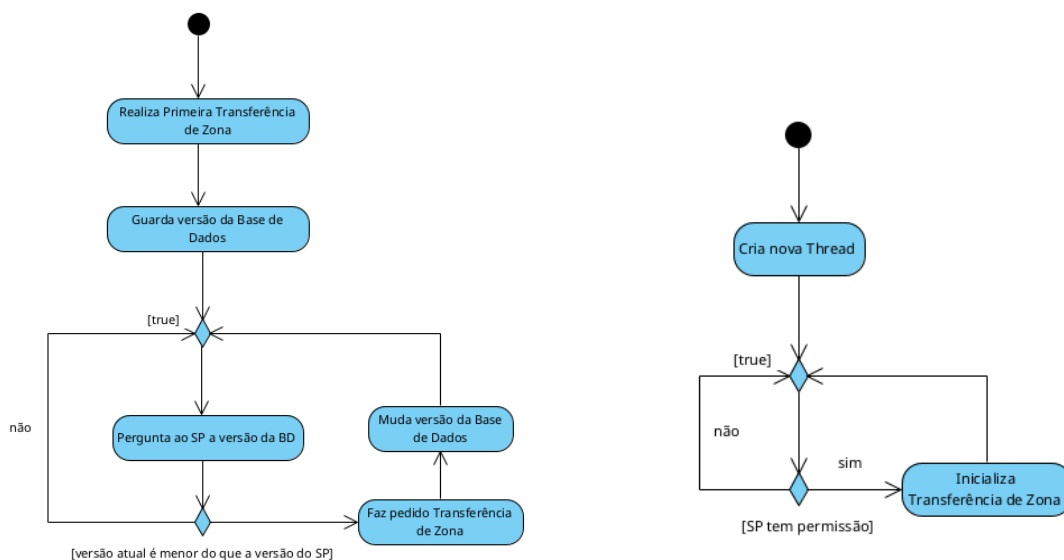


Fig. 12. Funcionamento da transferência de Zona

Funcionamento do Componente searchCache

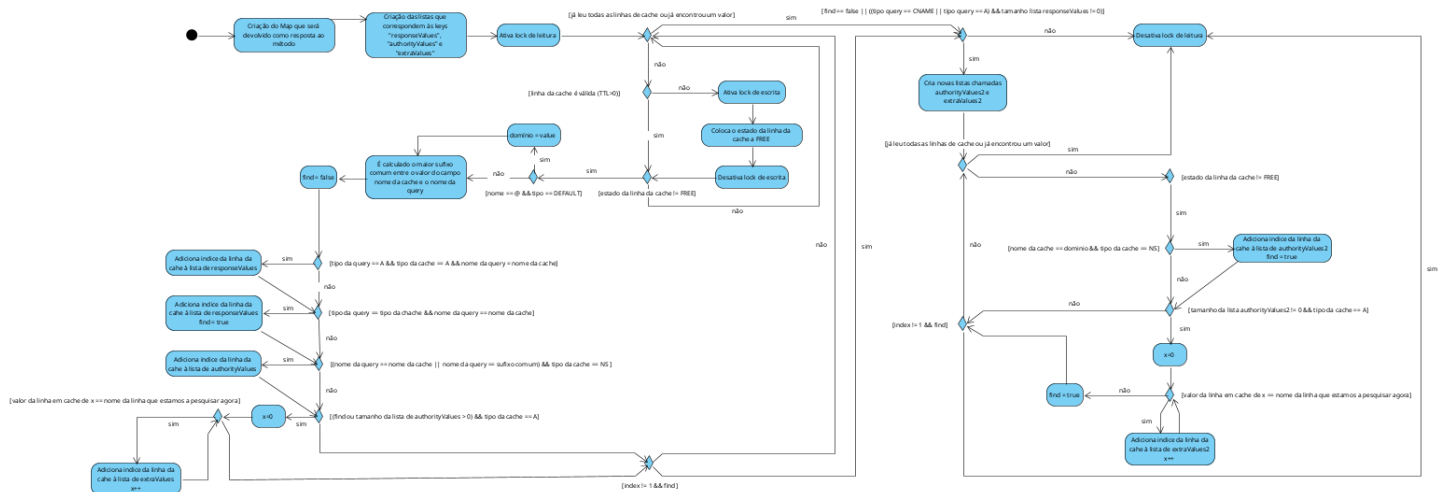


Fig. 13. Funcionamento da função de procura pela Resposta

Função que pesquisa a base de dados (neste caso contida na cache) para encontrar entradas que correspondam a um determinado nome e tipo de entrada. Este método faz isso começando a partir de uma posição específica na cache, procurando até encontrar todas as entradas correspondentes ou até chegar ao final da cache.

A função começa por criar um mapa que irá conter a lista de índices de entradas de `responseValues`, `authorityValues` e `extraValues`. Seguidamente, ela inicializa algumas variáveis, como o domínio e o maior sufixo comum entre o nome da query e o nome do domínio.

Em seguida, a função itera sobre a cache e verifica se o tempo de vida de cada entrada é menor do que o tempo desde que a entrada foi adicionada à cache. Se for, a entrada é marcada como "FREE".

A função também verifica se a entrada não está marcada como "FREE" e, se não estiver, compara o nome e o tipo da entrada com o nome e tipo de entrada que estão a ser pesquisados. Se houver correspondência, a entrada é adicionada à lista de responseValues. Para além disso, a função procura as correspondências do nome (ou maior sufixo comum) com o nome da entrada da cache e o tipo "NS" para adicionar à lista de authorityValues. Finalmente, procura as entradas com o tipo "A" correspondentes aos valores de responseValues e authorityValues encontrados, sem repetições.

Quando a iteração sobre a cache termina, a função devolve o mapa contendo as listas de entradas de resposta, autoridade e adicionais.

Funcionamento do Componente buildResponse

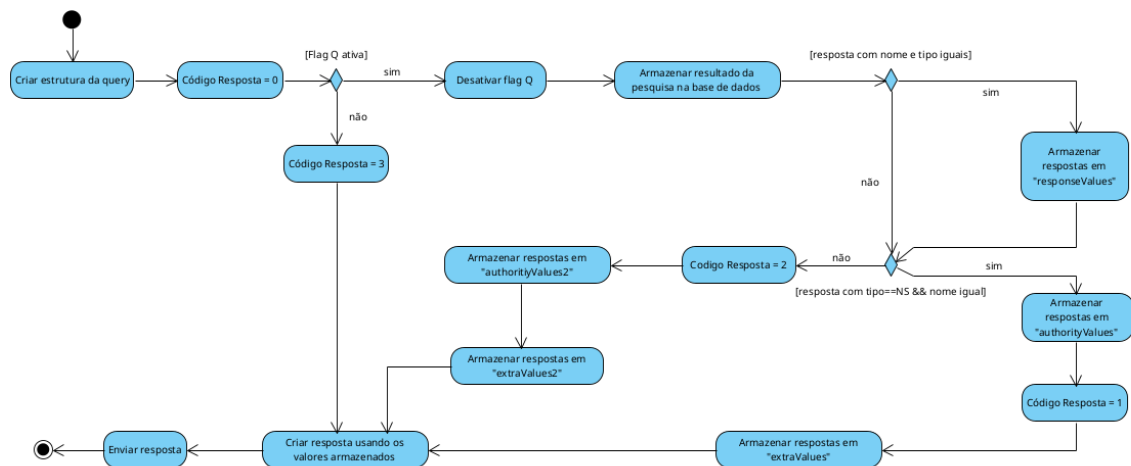


Fig. 14. Funcionamento da função de criação da Resposta

Esta função vai utilizar o mapa com as listas de resposta, autoridade e adicionais para construir uma resposta do tipo DNSHandler. Se a flag “Q” estiver ativa a resposta vai ser construída, caso contrário esta é descartada. Dependendo das correspondências feitas pela função searchCache, a função buildResponse irá calcular o código de resposta, o número de respostas, autoridades e adicionais encontradas, juntamente com as listas respectivas e irá construir o objeto DNSHandler.

Caso haja responseValues o código de resposta é 0. Caso não haja responseValues mas o domínio exista, o código de resposta é 1. Se a query for feita a um domínio que não existe, o código de resposta será 2 mas mesmo assim serão incluídos na resposta a autoridade que fez essa verificação.

Modelo de Informação

Ficheiros

- Os ficheiros de configuração são apenas lidos e processados no arranque do componente de software a que dizem respeito e moldam o seu comportamento.
- A informação presente nos ficheiros processados deve ser armazenada em memória.
- Se for necessário atualizar o comportamento dos servidores com informação modificada nos ficheiros de configuração ou de dados que lhes dizem respeito, a única alternativa é reiniciar esses servidores.
- Quando existe erro de leitura de ficheiros de entrada, o servidor escreve um log de erro e para de ler o ficheiro, terminando o processo por aí.

Sintaxe dos Ficheiros de configuração dos servidores SP, SS e SR

Os ficheiros de configuração dos servidores são os ficheiros em que é baseado todo o funcionamento dos servidores. Tal deve-se ao facto de que é nestes ficheiros que é declarada a localização do ficheiro de dados que é crucial visto que tem os dados sobre todos os componentes do domínio e os respectivos endereços IP.

É também nestes ficheiros que é estabelecida a hierarquia presente num domínio. Tal é realizado através da indicação dos SP e/ou SS e do servidor domínio por defeito que irá definir o comportamento dos servidores.

Outro aspeto essencial é a indicação sobre o ficheiro com a lista de ST, que será crucial na resolução de queries.

Finalmente, é indicado ficheiro de log onde será registada toda a atividade relevante do servidor.

De notar que nestes ficheiros:

- As linhas começadas por '#' são consideradas comentários e são ignoradas.
- As linhas em branco são ignoradas.
- Deve existir uma definição de parâmetro de configuração por cada linha seguindo a sintaxe: {parâmetro} {tipo do valor} {valor associado ao parâmetro}

Existem alguns tipos de valor comuns entre todos os ficheiros, nomeadamente, LG e ST.

Todos os servidores têm, pelo menos, duas entradas do tipo **LG**. Cada uma delas é referente ao caminho do ficheiro de logs que se deve utilizar para registar a atividade do servidor associada ao domínio indicado no parâmetro. Um dos parâmetros indicado deve ser o domínio do respectivo servidor. Há uma entrada que refere-se ao caminho para um ficheiro de log com toda a atividade que não seja diretamente referente aos domínios especificados noutras entradas LG (neste caso o parâmetro deve ser igual a "all").

Os valores associados ao tipo **ST** indicam o caminho para o ficheiro com a lista dos ST (servidores de topo), sendo que o parâmetro deve ser igual a “root”.

No entanto, para cada tipo de servidor, existem outros tipos de valores. A seguir será explicado o valor correspondente a cada tipo de valor para cada tipo de servidor.

Ficheiros de Configuração dos SP

```
# Configuration file for primary server for .yin.  
.yin DB ./files/yin.db  
.yin SS 10.0.4.10  
.yin SS 10.0.6.10  
.yin DD 10.0.0.10  
.yin LG ./files/yin.log  
all LG ./files/all-yin.log  
root ST ./files/rootservers.db
```

Fig. 15. Ficheiro de Configuração do SP do domínio .yin.

Os Servidores Primários são os únicos que têm uma entrada de tipo de valor **DB**. Este valor indica o caminho para o ficheiro de base de dados com a informação do domínio indicado no parâmetro.

Os ficheiros associados a este tipo de servidores têm uma entrada **SS** por cada Servidor Secundário existente no domínio. Cada entrada indica o endereço IP[:porta] dum SS do domínio indicado no parâmetro. Apenas os SS indicados têm autorização para pedir transferência de zona, isto é, a transmissão da informação da base de dados.

Neste caso, o valor associado ao tipo de valor **DD** refere-se ao endereço IP[:porta] do próprio servidor primário. Assim, restringe o funcionamento do SP, uma vez que usam este parâmetro para indicar os únicos domínios para os quais o servidor responde (quer a resposta esteja em cache ou não).

Ficheiros de Configuração dos SS

```
# Configuration file for secondary server for .yin  
.yin SP 10.0.0.10  
.yin SS 10.0.6.10  
.yin DD 10.0.0.10  
.yin LG ./files/yin.log  
all LG ./files/all-yin.log  
root ST ./files/rootservers.db
```

Fig. 16. Ficheiro de Configuração do SS1 do domínio .yin.

Todos os Servidores Secundários têm, obrigatoriamente, uma entrada com o tipo de valor **SP**. O valor desta entrada indica o endereço IP[:porta] do SP do domínio indicado no parâmetro.

Caso exista mais algum SS no domínio, será indicada através de linhas de entrada com tipo de valores **SS**. Neste caso, como cada domínio apenas tinha dois SS, e, por isso, só é apresentada uma linha com o endereço do outro SS do domínio.

Tal como nos ficheiros de configuração dos servidores primários, o tipo de valor **DD** serve para restringir o funcionamento dos SS. No entanto, neste caso, o valor associado corresponde ao endereço do Servidor Primário do domínio.

Ficheiros de Configuração dos SR

```
# Configuration file for primary server for .yin.  
.yin. DB ./files/yin.db  
.yin. SS 10.0.4.10  
.yin. SS 10.0.6.10  
.yin. DD 10.0.0.10  
.yin. LG ./files/yin.log  
all LG ./files/all-yin.log  
root ST ./files/rootservers.db
```

Fig. 17. Ficheiro de Configuração do SR do domínio .yin.

Nos ficheiros de configuração dos Servidores de Resolução, os valores correspondentes às entradas **DD** indicam os endereços IP[:porta] dum SS ou dum SP do domínio. Assim, indicam quais os servidores que podem contactar diretamente se receberem queries sobre estes domínios (quando a resposta não está em cache), em vez de contactar um dos ST.

Ficheiro com a lista de ST

```
# DNS file for root domain servers  
  
# ST1 , default port: 53  
10.0.8.10:53  
  
# ST2 , default port: 53  
10.0.9.10:53
```

Fig. 18. Ficheiro com a lista de Servidores de Topo

Este ficheiro tem a lista de ST (servidores de topo) que devem ser contactados sempre que necessário, ou seja, quando se quer saber informação sobre domínios acima do domínio atual e a resposta não está em cache.

Contém um endereço IP[:porta] ST por cada linha do ficheiro, sendo que as linhas começadas por '#' ou em branco devem ser ignoradas.

Ficheiros de log

```
20:56:2022.19:56:17:034, ZT, 127.0.0.1, SP

20:56:2022.19:56:17:034, ZT, 10.0.0.10, SS: transfer time: 22 milissegundos

20:56:2022.19:56:20:850, QR, /127.0.0.1:12792,
1, Q+R, 0, 0, 0, 0;
(Null);
(Null);
(Null);

20:56:2022.19:56:20:859, RP, /127.0.0.1:12792,
1, A, 0, 2, 3, 5;
.yin. MX mx1.yin. 86400 10,
.yin. MX mx2.yin. 86400 20;
.yin. NS ns1.yin. 86400,
.yin. NS ns2.yin. 86400,
.yin. NS ns3.yin. 86400;
ns1.yin. A 10.0.0.10 86400,
ns2.yin. A 10.0.4.10 86400,
ns3.yin. A 10.0.6.10 86400,
mx1.yin. A 10.0.0.12 86400,
mx2.yin. A 10.0.0.13 86400;
```

Fig. 19. Exemplo conteúdo do ficheiro de logs do domínio .yin.

Os ficheiros de log são aqueles que registam toda a atividade relevante do componente.

Seguem os seguintes requisitos:

- Existe uma entrada de log por cada linha do ficheiro;
- Sempre que um componente arranca este deve verificar a existência dos ficheiros de log indicados no seu ficheiro de configuração. Se não existir, deve ser criado. Se já existir, as novas entradas devem ser registadas a partir da última entrada já existente no ficheiro.
- A sintaxe de cada entrada é a seguinte {etiqueta temporal} {tipo de entrada} {endereço IP[:porta]} {dados da entrada}
- A etiqueta temporal é a data e hora completa do sistema operativo na altura em que aconteceu a atividade registada

A seguir será explicado cada um dos tipos de entrada existentes:

- **QR/QE** – Foi 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** – Foi enviada/recebida uma resposta a uma query para o/do endereço indicado. Os dados da entrada devem ser os dados relevantes incluídos na resposta

à query. A sintaxe dos dados de entrada é a mesma que é usada no PDU de resposta às queries no modo debug de comunicação entre os elementos.

- **ZT** – Foi iniciado e concluído corretamente um processo de transferência de zona. O endereço deve indicar o servidor na outra ponta da transferência. Os dados da entrada devem indicar qual o papel do servidor local na transferência (SP ou SS) e, opcionalmente, a duração em milissegundos da transferência e o total de bytes transferidos.
- **EV** – Foi detectado um evento/atividade interna no componente. O endereço deve indicar 127.0.0.1 (ou localhost ou @). Os dados da entrada devem incluir informação adicional sobre a atividade reportada (por exemplo, ficheiro de configuração/dados/ST lido, criado ficheiro de log, etc.).
- **ER** – Foi recebido um PDU do endereço indicado que não foi possível decodificar corretamente. Opcionalmente, os dados da entrada podem ser usados para indicar informação adicional (como, por exemplo, o que foi possível decodificar corretamente e em que parte/byte aconteceu o erro).
- **EZ** – Foi detectado um erro num processo de transferência de zona que não foi concluída corretamente. O endereço deve indicar o servidor na outra ponta da transferência. Os dados da entrada devem indicar qual o papel do servidor local na transferência (SP ou SS).
- **FL** – Foi detectado um erro no funcionamento interno do componente. O endereço deve indicar 127.0.0.1. Os dados da entrada devem incluir informação adicional sobre a situação de erro (por exemplo, um erro na decodificação ou incoerência dos parâmetros de algum ficheiro de configuração ou de base de dados).
- **TO** – Foi detectado um timeout na interação com o servidor no endereço indicado. Os dados da entrada devem especificar que tipo de timeout ocorreu (resposta a uma query ou tentativa de contato com um SP para saber informações sobre a versão da base de dados ou para iniciar uma transferência de zona).
- **SP** – A execução do componente foi parada. O endereço deve indicar 127.0.0.1. Os dados da entrada devem incluir informação adicional sobre a razão da paragem se for possível obtê-la.
- **ST** – A execução do componente foi iniciada. O endereço deve indicar 127.0.0.1. Os dados da entrada devem incluir informação sobre a porta de atendimento, sobre o valor do timeout usado (em milissegundos) e sobre o modo de funcionamento (modo “shy” ou modo debug).

É importante referir que durante a segunda parte do trabalho acrescentou-se o funcionamento dos logs em modo debug, tal como é possível observar na figura 21.

```
root@SRyin:/tmp/pycore.45319/SRyin.conf# cd dns && java Servidor yin/sr
23:24:2022,09:24:14:604, EV, localhost, Ficheiro de configuração lido
23:24:2022,09:24:14:619, ST, localhost, porta: 53 timeout: 600000 debug: true
23:24:2022,09:24:14:627, EV, localhost, Ficheiro ST lido
23:24:2022,09:24:14:627, EV, localhost, Ficheiro de logs criado.
Query recebida de: /10.0.0.11:56135
23:27:2022,09:27:45:964, RR, /10.0.0.11:56135,
96, Q, 0, 0, 0, 0;
.yang., NS;
(Null);
(Null);
(Null);
23:27:2022,09:27:45:989, QE, /10.0.8.10:56135,
96, A, 2, 0, 0, 0;
.yang., NS;
(Null);
(Null);
(Null);
23:27:2022,09:27:46:072, RR, /10.0.0.11:56135,
96, A, 0, 3, 3, 3;
.yang., NS;
.yang. NS sp.yang. 86400,
.yang. NS ss1.yang. 86400,
.yang. NS ss2.yang. 86400;
.yang. NS sp.yang. 86400,
.yang. NS ss1.yang. 86400,
.yang. NS ss2.yang. 86400;
sp.yang. A 10.0.2.10 86400,
ss1.yang. A 10.0.5.10 86400,
ss2.yang. A 10.0.7.10 86400;
```

Fig 20. Terminal do SR do domínio yin após query NS ao domínio yang

Ficheiro de dados do SP

Estes ficheiros contêm toda a informação sobre um determinado domínio, isto é, todos os dados relevantes para a realização e manutenção de transferências de zona, os endereços de IP de todos os hosts que compõem o domínio (e servidores principais dos subdomínios) e os respectivos nomes (e nomes canónicos).

É sobre estes ficheiros que se irão realizar todas as operações de pesquisa relativas às queries recebidas dos clientes.

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

@ DEFAULT yin.
TTL DEFAULT 86400

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

@ NS ns1.yin. TTL
@ NS ns2.yin. TTL
@ NS ns3.yin. TTL
light NS splight.light.yin.
@ MX mx1.yin. TTL 10
@ MX mx2.yin. TTL 20

ns1 A 10.0.0.10 TTL
ns2 A 10.0.4.10 TTL
ns3 A 10.0.6.10 TTL
splight.light A 10.0.1.10 TTL
mx1 A 10.0.0.12 TTL
mx2 A 10.0.0.13 TTL
www A 10.0.0.14 TTL 200

spyin CNAME ns1 TTL
ssyin1 CNAME ns2 TTL
ssyin2 CNAME ns3 TTL
mail1 CNAME mx1 TTL
mail2 CNAME mx2 TTL
```

Fig. 21. Ficheiro de dados do domínio .yin.

A seguir são apresentados os requisitos deste tipo de ficheiros:

- As linhas começadas por '#' ou brancas são consideradas comentários e são ignoradas.
- Existe uma definição de parâmetro de dados por cada linha seguindo a sintaxe: {parâmetro} {tipo do valor} {valor} {tempo de validade} {prioridade}
- O tempo de validade (TTL) é o tempo máximo em segundos que os dados podem existir numa cache de um servidor (quer seja cache normal ou cache negativa, se for suportada).
- Quando o TTL não é suportado num determinado tipo, o seu valor deve ser igual a zero.
- O campo da prioridade é um valor inteiro menor que 256 e que define uma ordem de prioridade de vários valores associados ao mesmo parâmetro. Quanto menor o valor maior a prioridade.
- Para parâmetros com um único valor ou para parâmetros em que todos os valores têm a mesma prioridade, o campo não deve existir.
- O campo prioridade ajuda a implementar sistema de backup de serviços/hosts quando algum está em baixo (números de prioridade entre 0 e 127) ou balanceamento de carga de serviços/hosts que tenham vários servidores/hosts aplicativos disponíveis (números de prioridade entre 128 e 255).
- Os nomes completos de e-mail, domínios, servidores e hosts devem terminar com um '.'.
- Quando os nomes não terminam com '.' subentende-se que são concatenados com um prefixo por defeito definido através do parâmetro @ do tipo DEFAULT.

Tipos de valores possíveis:

- **DEFAULT*** – Define um nome (ou um conjunto de um ou mais símbolos) como uma macro que deve ser substituída pelo valor literal associado (não pode conter espaços nem o valor dum qualquer parâmetro DEFAULT). O parâmetro @ é reservado para identificar um prefixo por defeito que é acrescentado sempre que um nome não apareça na forma completa (i.e., terminado com '.').
- **SOASP** – O valor indica o nome completo do SP do domínio (ou zona) indicado no parâmetro.
- **SOADMIN** – O valor indica o endereço de e-mail completo do administrador do domínio (ou zona) indicado no parâmetro. O símbolo '@' deve ser substituído por '.' e '.' no lado esquerdo do '@' devem ser antecidos de '\';
- **SOASERIAL** – O valor indica o número de série da base de dados do SP do domínio (ou zona) indicado no parâmetro. Sempre que a base de dados é alterada este número deve ser Incrementado.
- **SOAREFRESH** – O valor indica o intervalo temporal em segundos para um SS perguntar ao SP do domínio indicado no parâmetro qual o número de série da base de dados dessa zona.
- **SOARETRY** – O valor indica o intervalo temporal para um SS voltar a perguntar ao SP do domínio indicado no parâmetro qual o número de série da base de dados dessa zona, após um timeout.
- **SOAEXPIRE** – O valor indica o intervalo temporal para um SS deixar de considerar a sua réplica da base de dados da zona indicada no parâmetro como válida,

deixando de responder a perguntas sobre a zona em causa, mesmo que continue a tentar contactar o SP respetivo.

- **NS** – O valor indica o nome de um servidor que é autoritativo para o domínio indicado no parâmetro, ou seja, o nome do SP ou dum dos SS do domínio; este tipo de parâmetro suporta prioridades.
- **A** – O valor indica o endereço IPv4 de um host/servidor indicado no parâmetro como nome; este tipo de parâmetro suporta prioridades.
- **CNAME** – O valor indica um nome canónico (ou alias) associado ao nome indicado no parâmetro. Um nome canónico não deve apontar para um outro nome canónico nem podem existir outros parâmetros com o mesmo valor do nome canónico.
- **MX** – O valor indica o nome de um servidor de e-mail para o domínio indicado no parâmetro; este tipo de parâmetro suporta prioridades.
- **PTR** – O valor indica o nome de um servidor/host que usa o endereço IPv4 indicado no parâmetro. A indicação do IPv4 é feita como nos domínios de DNS reverso (rDNS) quando se implementa reverse-mapping.

É de referir que foram realizadas as seguintes correções à primeira fase do trabalho, referente ao tipo de pesquisa CNAME. O resultado obtido pode ser observado na fig. 22.

```
<1/SRyin.conf/dns# java cldns localhost ns1.yin. CNAME

# Header
MESSAGE-ID = 155, FLAGS = A, RESPONSE-CODE = 0,
N-VALUES = 1, N-AUTHORITIES = 3, N-EXTRA-VALUES = 3;

# Data: Query Info
QUERY-INFO.NAME = ns1.yin., QUERY-INFO.TYPE = CNAME;

# Data: List of Responses, Authorities and Extra Values
RESPONSE-VALUES = sp.yin. CNAME ns1.yin. 86400;
AUTHORITIES-VALUES = .yin. NS ns1.yin. 86400,
AUTHORITIES-VALUES = .yin. NS ns2.yin. 86400,
AUTHORITIES-VALUES = .yin. NS ns3.yin. 86400;
EXTRA-VALUES = ns1.yin. A 10.0.0.10 86400,
EXTRA-VALUES = ns2.yin. A 10.0.4.10 86400,
EXTRA-VALUES = ns3.yin. A 10.0.6.10 86400;
```

Fig 22. Terminal do SR do domínio yin após query NS ao domínio yang

Codificação PDU

Para que a mensagem DNS possa ser enviada pelos sockets UDP, a informação tem que ser codificada. No sistema desenvolvido a mensagem DNS é um objeto (DNSHandler) que contém os vários campos do cabeçalho e dados da mensagem como atributos. Inicialmente, o objeto é transformado numa String, convertendo todos os atributos para String e separando-os por vírgulas. Feito isto, é usado o método `getBytes()` da biblioteca base de Java.

Na recepção das mensagens no outro socket, é extraída a parte dos dados do DatagramPacket. Após isso, é realizada a conversão de um array de bytes de volta para String. Depois, é feito um parsing dessa String, separando-a em pedaços sempre que é encontrada uma vírgula e é construída novamente numa mensagem DNS (DNSHandler).

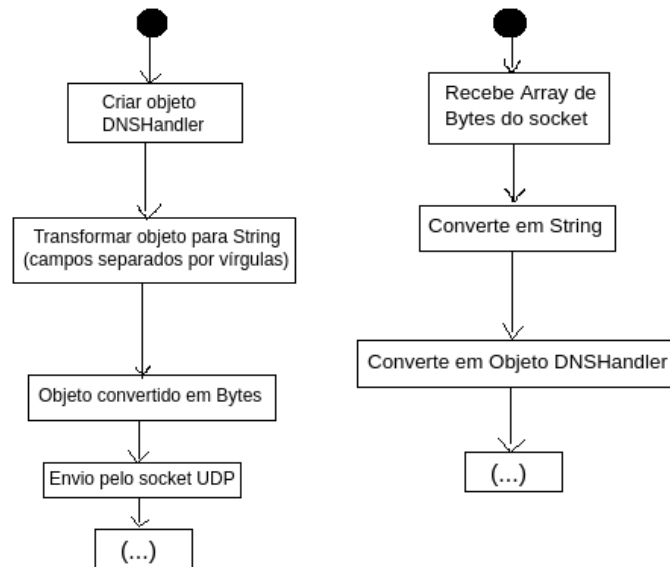


Fig. 23. Codificação e decodificação da mensagem DNS

Ambiente de Teste

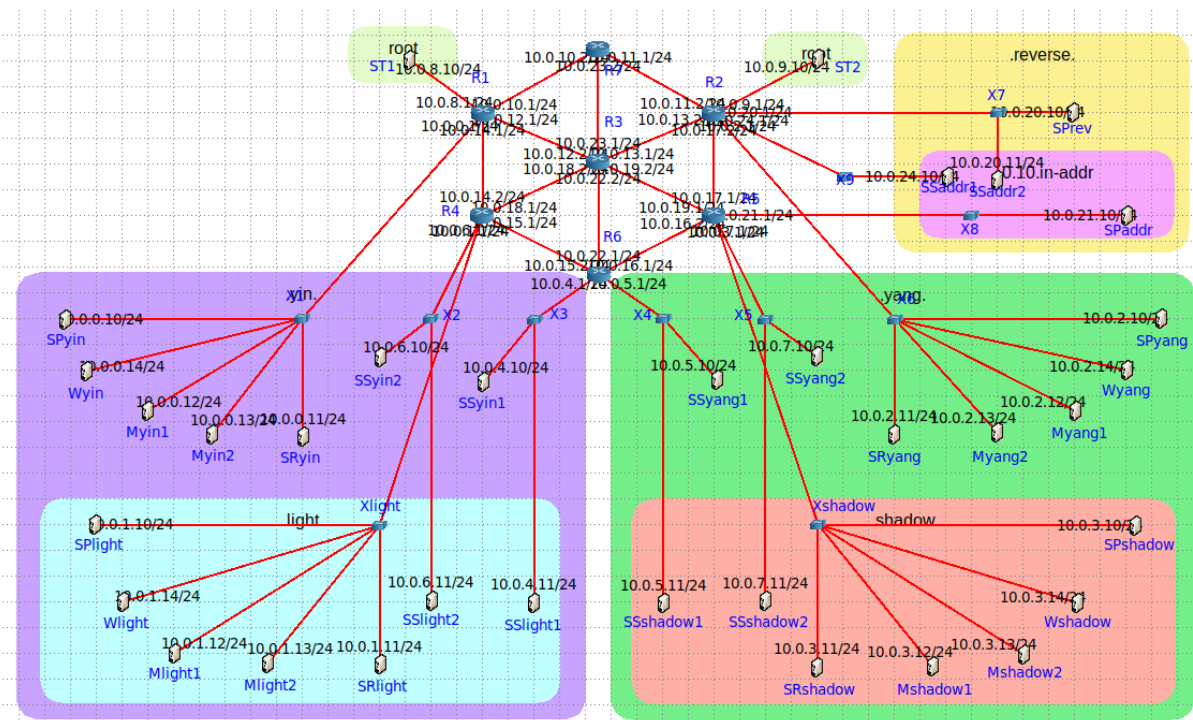


Fig 24. Arquitetura do Ambiente de Teste

O ambiente de teste configurado no CORE podem-se observar os seguintes componentes:

- Dois Servidores de Topo (ST);
- Três domínios (.yin., .yang. e .reverse.)
- Um subdomínio em cada domínio de topo (light.yin., shadow.yang. e 0.10.in-addr.reverse., respetivamente);
- Um domínio de topo nomeado de .reverse onde estarão pendurados os domínios de DNS reverso;
- Um Servidor Principal (SP - que é o SDT nos domínios de topo), dois Servidores Secundários (SS) e um Servidor de Resolução (SR) em cada domínio/subdomínio;
- Adicionalmente, um Servidor Web (W) e dois Servidores de Email (M) em cada domínio/subdomínio;
- Um cliente que é implementado no SR do subdomínio shadow.yang;

É nestes componentes que será realizada a implementação das aplicações que determinam o funcionamento dos componentes.

Como medidas de segurança, foi acrescentada redundância ao nível dos routers, e a existência de switches entre os routers e os servidores/clientes. Para além disso, os servidores de topo encontram-se conectados com routers diferentes já que, caso uma rede falhe, ainda existe um servidor de topo ativo. O mesmo raciocínio foi efetuado para a separação entre os servidores secundários e primários.

Note-se que apesar dos servidores estarem visualmente agrupados na mesma zona da topologia isto não implica que geograficamente a distribuição destes seja representada pela imagem.

Testes dos Protótipos SP, SS e CL

```

robert@robert:~/aulas/3ano/CC/TP2/code/dns$ java Servidor 55555 1000
True SPyin.conf
SERVIDOR PRINCIPAL INICIALIZADO.

À espera de pedido de Transferência de Zona.
SS: 127.0.0.1 conectado.
#####
Transferência de Zona terminada.

-CONEXÃO UDP INICIADA-
À espera de queries:

Query recebida de: /127.0.0.1:52748
# Header
MESSAGE-ID = 36167, FLAGS = Q+R, RESPONSE-CODE = 0,
N-VALUES = 0, N-AUTHORITIES = 0, N-EXTRA-VALUES = 0;
# Data: Query Info
QUERY-INFO.NAME = .yin., QUERY-INFO.TYPE = NS;
# Data: List of Responses, Authorities and Extra Values
RESPONSE-VALUES = (Null);
AUTHORITIES-VALUES = (Null);
EXTRA-VALUES = (Null);

```

Fig 25. Terminal do Servidor Principal do domínio .yin.

```

robert@robert:~/aulas/3ano/CC/TP2/code/dns$ java Servidor 55556 1000 True
SSyin1.conf
SERVIDOR SECUNDÁRIO INICIALIZADO.

Transferência de Zona iniciada.
#####
Transferência de Zona terminada.

-CONEXÃO UDP INICIADA-
À espera de queries:

Query recebida de: /127.0.0.1:8432
# Header
MESSAGE-ID = 36186, FLAGS = Q+R, RESPONSE-CODE = 0,
N-VALUES = 0, N-AUTHORITIES = 0, N-EXTRA-VALUES = 0;
# Data: Query Info
QUERY-INFO.NAME = .yin., QUERY-INFO.TYPE = MX;
# Data: List of Responses, Authorities and Extra Values
RESPONSE-VALUES = (Null);
AUTHORITIES-VALUES = (Null);
EXTRA-VALUES = (Null);

```

Fig 26. Terminal do Servidor Secundário 1 do domínio .yin.

```

robert@robert:~/aulas/3ano/CC/TP2/code/dns$ java Cliente 55555 .yin. NS
# Header
MESSAGE-ID = 36167, FLAGS = A, RESPONSE-CODE = 0,
N-VALUES = 3, N-AUTHORITIES = 3, N-EXTRA-VALUES = 3;
# Data: Query Info
QUERY-INFO.NAME = .yin., QUERY-INFO.TYPE = NS;
# Data: List of Responses, Authorities and Extra Values
RESPONSE-VALUES = .yin. NS ns1.yin. 86400,
RESPONSE-VALUES = .yin. NS ns2.yin. 86400,
RESPONSE-VALUES = .yin. NS ns3.yin. 86400;
AUTHORITIES-VALUES = .yin. NS ns1.yin. 86400,
AUTHORITIES-VALUES = .yin. NS ns2.yin. 86400,
AUTHORITIES-VALUES = .yin. NS ns3.yin. 86400;
EXTRA-VALUES = ns1.yin. A 10.0.0.10 86400,
EXTRA-VALUES = ns2.yin. A 10.0.4.10 86400,
EXTRA-VALUES = ns3.yin. A 10.0.6.10 86400;
robert@robert:~/aulas/3ano/CC/TP2/code/dns$ java Cliente 55556 .yin. MX
# Header
MESSAGE-ID = 36186, FLAGS = A, RESPONSE-CODE = 0,
N-VALUES = 2, N-AUTHORITIES = 3, N-EXTRA-VALUES = 5;
# Data: Query Info
QUERY-INFO.NAME = .yin., QUERY-INFO.TYPE = MX;
# Data: List of Responses, Authorities and Extra Values
RESPONSE-VALUES = .yin. MX mx1.yin. 86400 10,
RESPONSE-VALUES = .yin. MX mx2.yin. 86400 20;
AUTHORITIES-VALUES = .yin. NS ns1.yin. 86400,
AUTHORITIES-VALUES = .yin. NS ns2.yin. 86400,
AUTHORITIES-VALUES = .yin. NS ns3.yin. 86400;
EXTRA-VALUES = ns1.yin. A 10.0.0.10 86400,
EXTRA-VALUES = ns2.yin. A 10.0.4.10 86400,
EXTRA-VALUES = ns3.yin. A 10.0.6.10 86400,
EXTRA-VALUES = mx1.yin. A 10.0.0.12 86400,
EXTRA-VALUES = mx2.yin. A 10.0.0.13 86400;

```

Fig 27. Terminal do Cliente

Nas imagens acima (fig 17, 18, 19) podemos observar uma sequência de comandos que foram concretizados para servir como testes para verificar se a implementação dos componentes vai de acordo com o definido no planejamento.

Inicialmente, inicia-se o Servidor Primário do domínio .yin.. Este fica à espera de receber um pedido de Transferência de Zona de um Servidor Secundário num socket TCP. Assim que o SS é inicializado, envia um pedido de Transferência de Zona ao SP, enviando por argumento, o nome do domínio. O SP aceita a transferência de zona, uma vez que pertencem ao mesmo domínio e a transferência da base de dados é concluída com sucesso. Ambos os servidores ficam, em espera, à escuta de pedidos enviados por UDP. Neste exemplo, o SP está à escuta na porta 55555, o SS está à escuta na porta 55556.

Assim, o Cliente envia uma pergunta ao SP sobre o nome “.yin.” e o tipo “NS”. O SP realiza uma pesquisa na base de dados e/ou cache e, ao encontrar uma resposta, envia-a ao Cliente, sendo esta impressa no ecrã do mesmo. Para testar se a transferência de zona realmente funcionou, o Cliente envia uma pergunta ao SS sobre o nome “.yin.” e o tipo “MX”. Este envia a resposta que encontra, neste caso, na cache e o Cliente após recebê-la, imprime-a.

Zona Reversa

A zona reversa do DNS é uma parte do sistema de nomes de domínio que mapeia endereços IP para nomes de domínio. Enquanto o DNS normal é usado para traduzir nomes de domínio para endereços IP, a zona reversa faz o processo inverso, ou seja, mapeia endereços IP para nomes de domínio. Isso é útil em várias situações, como na verificação da autenticidade de um endereço de email ou na solução de problemas de rede. A zona reversa é configurada como uma sub-rede de uma zona de DNS normal e é geralmente mantida pelo provedor de serviços de internet ou pelo administrador de rede.

Como todos os ips são do estilo 10.0.x.x. não é preciso adicionar muitas informações na base de dados e, por isso, foi adicionado um sub-domínio ao .reverse. Chamado .0.10.in-addr. que contém uma base de dados com entradas do tipo PTR que mapeiam os servidores primários e dos servidores de mail de todos os domínios e subdomínios em questão.

Situações de Erro

Possíveis Situações de Erro - Modelo de Comunicação

A seguir serão indicadas as situações possíveis de erro e como o sistema lida com elas, incluindo ameaças de segurança:

- Queries a um domínio ou subdomínio que não existe - Caso esta situação aconteça, nenhum servidor conseguirá responder à respetiva query e, por isso, o último verificado, se não tiver na sua base de dados uma entrada que responda à query, terá que enviar uma mensagem de erro returning a NXDOMAIN response, ou "Non-existent Domain".
- Rede falhar - Como existe redundância na arquitetura apresentada, se a rede falhar, é possível enviar a mensagem por outro caminho.
- Tentativa de transferência de zona para ips desconhecidos - O Servidor principal tem conhecimento de todos os servidores secundários e, por isso, consegue restringir para que servidores pode-se realizar transferência de zona. Assim, é impedido que os dados que se encontram na base de dados do servidor primário, sejam enviados para servidores externos.

Possíveis Situações de Erro - Modelo de Informação

O componente deve saber reportar as situações de incoerência de configuração que possam resultar dum ficheiro de configuração ou sintaxes que não entenda; nesses casos o componente deve registar a informação nos logs respetivos e encerrar imediatamente a execução.

O SP deve saber reportar as situações de incoerência nos valores dos parâmetros ou de sintaxe não compreendida no ficheiro de dados.

Tabela de Atividades Primeira Fase

Atividade Desenvolvida	Participante(s)
1ª Versão Ambiente de Teste	Rita e Robert
1ª Implementação no Core do Ambiente de Teste	Robert
Definição de Requisitos	Rita
1ª Versão Ficheiros de Configuração	Rita
Lista Servidores de Topo	Robert
1ª Versão Ficheiros de Dados	Robert
Descrição da Arquitetura do Sistema	Rita
Descrição do Modelo de Comunicação	Rita
Descrição do Modelo de Informação	Robert
Implementação nova versão do Ambiente de Teste no core	Robert
Descrição do Ambiente de Teste	Robert
Criação do Código referente ao Componentes	Rita
Refazer ficheiros de Configuração e Dados	Robert
Criação de código referente ao parse dos Ficheiros recebidos pelos Servidores	Rita
Criação de Código referente à transferência de queries através de sockets	Robert
Criação de Código referente à procura da resposta no Servidor	Rita
Criação de Código referente à Transferência de Zona	Robert
Adicionar criação de logs nos servidores	Rita

Tabela de Atividades Segunda Fase

Tarefa	Participação Rita	Participação Robert
B.1	80	20
B.2	20	80
B.3	80	20
B.4	20	80
B.5	50	50
B.7	80	20
B.8	20	80
B.9	50	50

Conclusão Primeira Fase

Na primeira parte deste trabalho conseguimos expandir o nosso conhecimento sobre o funcionamento do DNS, ao desenvolvermos e implementarmos uma versão mais simplificada.

Neste momento temos uma versão básica do sistema que implementa perguntas e respostas a mensagens DNS, e realiza uma cópia da base de dados entre servidores. Ainda não há implementação de paralelismo, o sistema corre apenas sequencialmente.

Na segunda fase foram implementadas funcionalidades que permitem paralelismo para aumentar a quantidade de servidores e clientes concorrentes, aproximando-nos assim mais um pouco do contexto real.

Para além disso, os componentes foram aprimorados, tornando possível uma pesquisa iterativa e o funcionamento da cache.

Lista de referências e/ou bibliografia usada

1. Powerpoint Domain Name System fornecido pelo Professor
2. Computer Networking - A Top-Down Approach, 7th Edition, Kurose, Ross

Anexos

1. Manual de Utilização

Funcionamento do Servidor - Programa sdns

```
root@SRyin:/tmp/pycore.45319/SRyin.conf/dns# java Servidor --help
Argumentos válidos para inicialização do Servidor:
-Argumentos opcionais-
    -p [porta]
    -t [timeout]
    -d [debug]
-Argumento obrigatório- (inserir sempre no fim)
    [fileConfig]
```

Fig 28. Opções de inicialização do Servidor

Tal como é possível observar através da imagem 23, para inicializar o servidor existem argumentos obrigatórios e opcionais. Esta opção de ajuda foi criada para possibilitar uma visualização geral dos argumentos.

Assim, existem os seguintes argumentos opcionais:

- -p [porta] - default 53
- -t [timeout] - default 600000
- -d [debug] - default true

Estas opções podem ser chamadas em qualquer ordem, sendo que, no final, tem de ser sempre incluído o caminho para o ficheiro.

Seguindo a arquitetura utilizada, os ficheiros estão organizados da seguinte forma:

Existe uma pasta geral designada files. Dentro dessa pasta existem quatro pastas: config, database, logs e rootservers.

Cada uma destas pastas contém a informação associada ao seu respetivo nome. Por exemplo, todos os ficheiros de configuração encontram-se na pasta config e assim respetivamente.

Dentro da pasta config, encontram-se as pastas com os nomes associados a domínios e subdomínios, sendo que cada uma terá os respetivos ficheiros. Por exemplo, o caminho para o ficheiro de configuração do servidor primário do subdomínio light será o seguinte: files/config/light/sp.conf

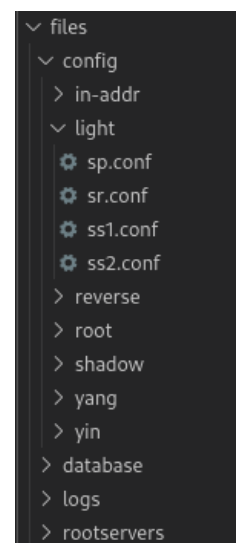


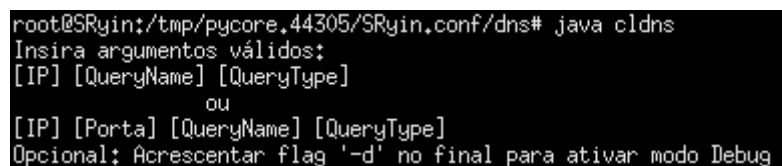
Fig 29. Organização de Ficheiros

O processo é mais simples para os ficheiros de dados já que apenas existe um ficheiro por domínio/subdomínio e para os ficheiros de logs.

A indicação dos Servidores de topo encontram-se no ficheiro rootservers.db na pasta rootservers.

Para simplificar o processo de escrita aquando da inicialização dos servidores, ou seja, no acesso aos ficheiros de configuração, apenas é necessário escrever a pasta do subdomínio/domínio com o respetivo nome do ficheiro que se pretende. Por exemplo, para aceder ao ficheiro de configuração do servidor SP shadow, basta apenas colocar na inicialização shadow/sp.

Funcionamento do Cliente - Programa cldns



```
root@SRyin:/tmp/pycore.44305/SRyin.conf/dns# java cldns
Insira argumentos válidos:
[IP] [QueryName] [QueryType]
ou
[IP] [Porta] [QueryName] [QueryType]
Opcional: Acrescentar flag '-d' no final para ativar modo Debug
```

Fig 30. Opções de inicialização do Cliente

No caso do cliente, é preciso indicar os seguintes argumentos:

- IP - IP do Servidor de Resolução que irá contactar diretamente
- Porta (opcional) - Porta do Servidor de Resolução - por default 53
- QueryName - Nome da query
- QueryType - Tipo da query

Para facilitar, é possível escrever “localhost” em vez do IP do Servidor de Resolução, sendo que o IP corresponderá ao Servidor de Resolução do domínio onde é inicializado o cliente.

O nome da query corresponde aos nomes dos servidores ou domínios/subdomínios da arquitetura. Por exemplo, é possível realizar a uma query ao subdomínio .light.yin. ou à máquina ns1.yin. Por outro lado, também é possível que o nome seja um ip de um servidor quando se pretende utilizar o modo reverso, sendo o tipo da query, neste caso, PTR.

Os tipos de valores possíveis são indicados na secção *Modelo de Informação*.

Quando se pretende guardar os resultados das queries realizadas pelo Cliente, ativa-se o modo debug no cliente, adicionando a flag “-d” no final da query. Por exemplo, utiliza-se o seguinte comando para realizar-se uma query do tipo NS ao .yin.

- `java cldns localhost .yin. NS -d.`

Todos os logs realizados pelo cliente são guardados na pasta de logs, no ficheiro cl.log.

2. Alguns testes realizados

→ Verificar se o DD é contactado antes do ST

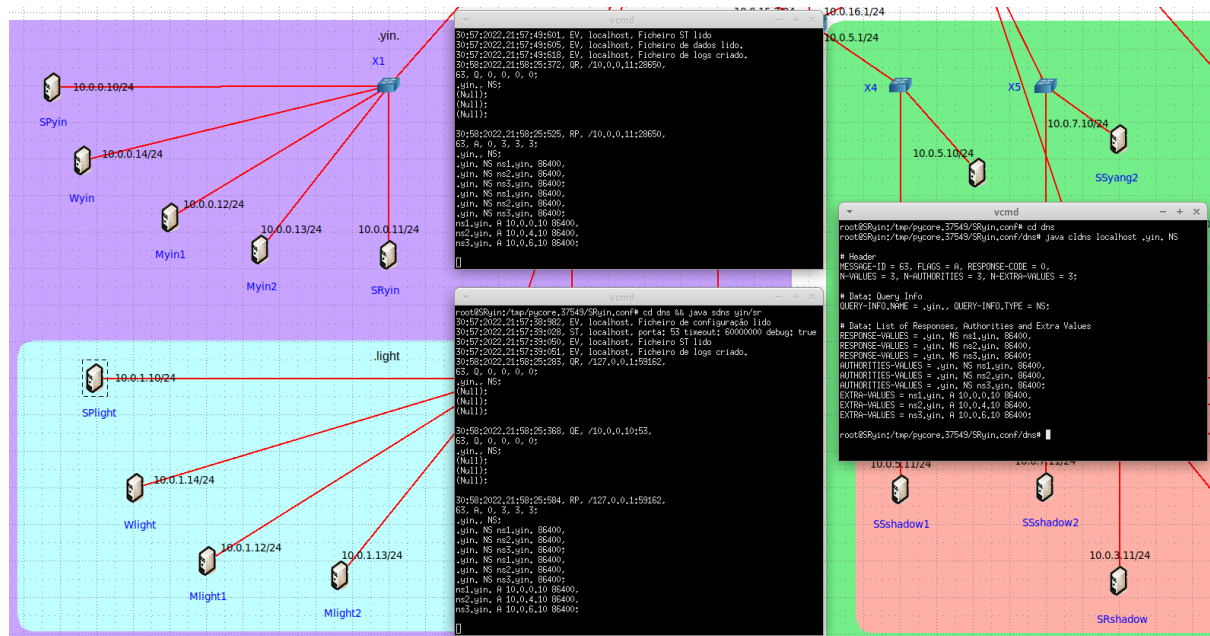


Fig 31. Query

→ Verificar se acede diretamente ao .light.yin. após ida ao DD

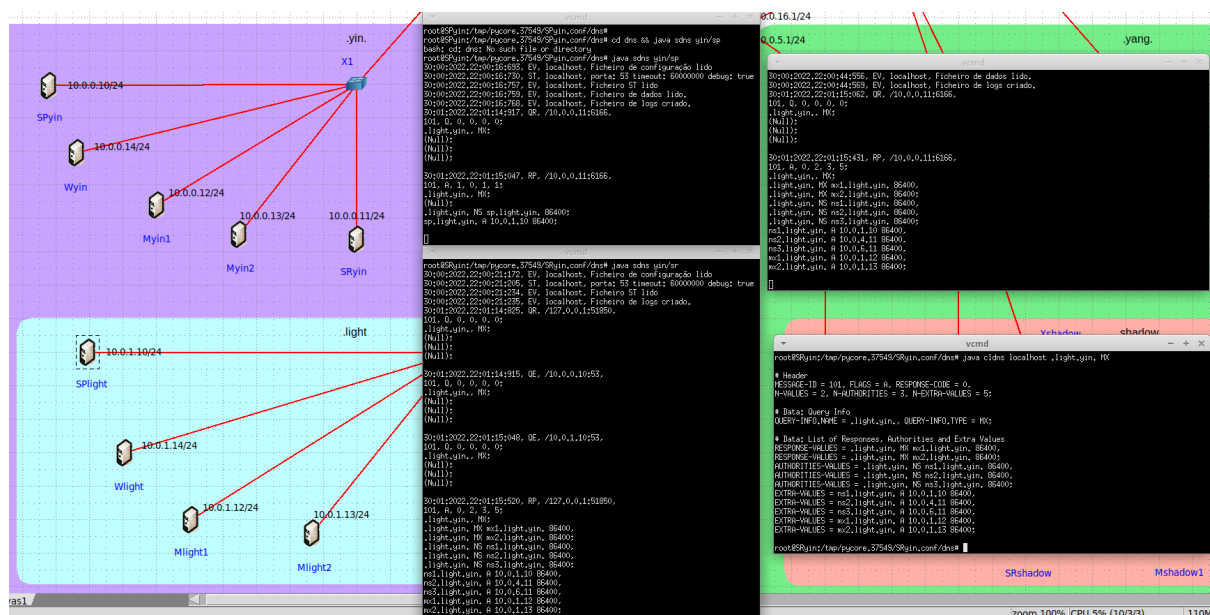


Fig 32. Query

→ Verificar se a query está a ir para o SS quando o SP falha

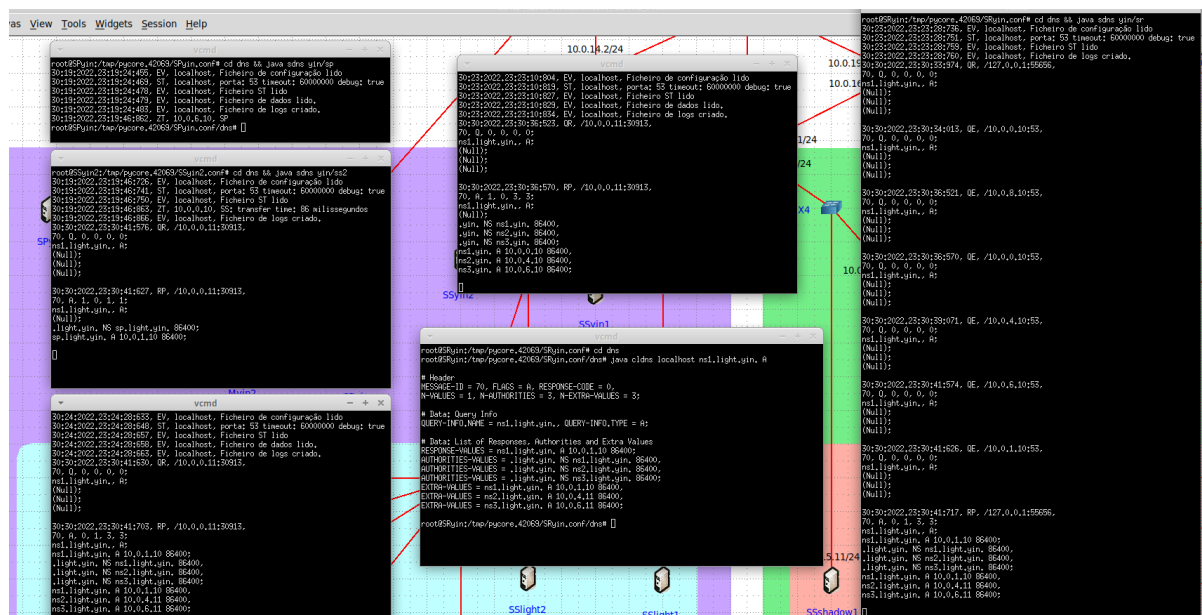


Fig 33. Query

→ Verificar que a query vai para o ST2 quando o ST1 não está ativo

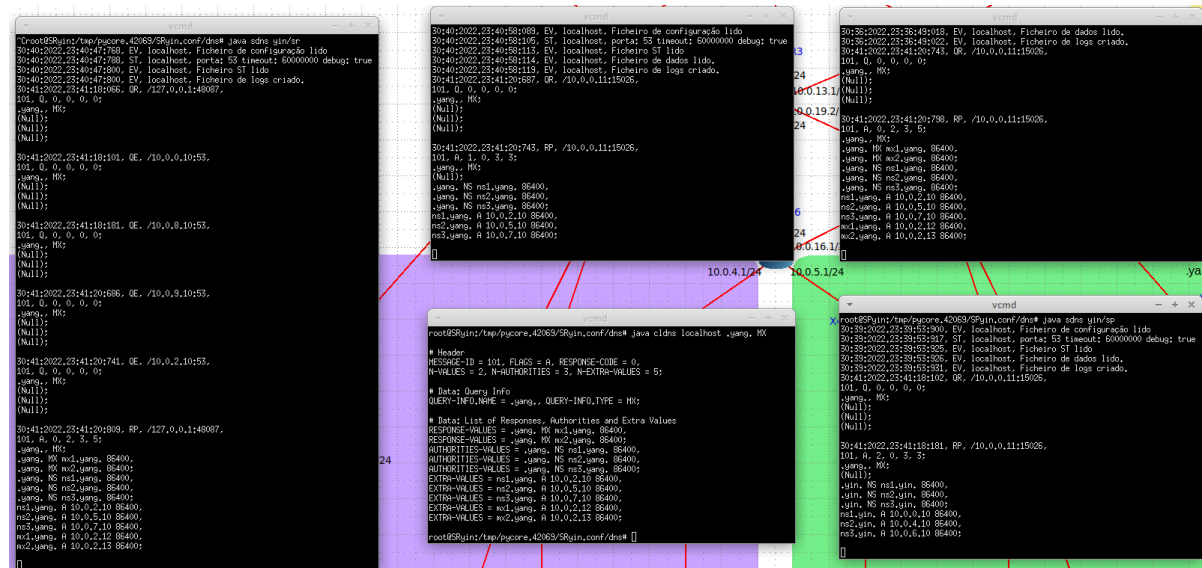


Fig 34. Query

→ Verificação query CNAME à máquina ns1 do subdomínio shadow

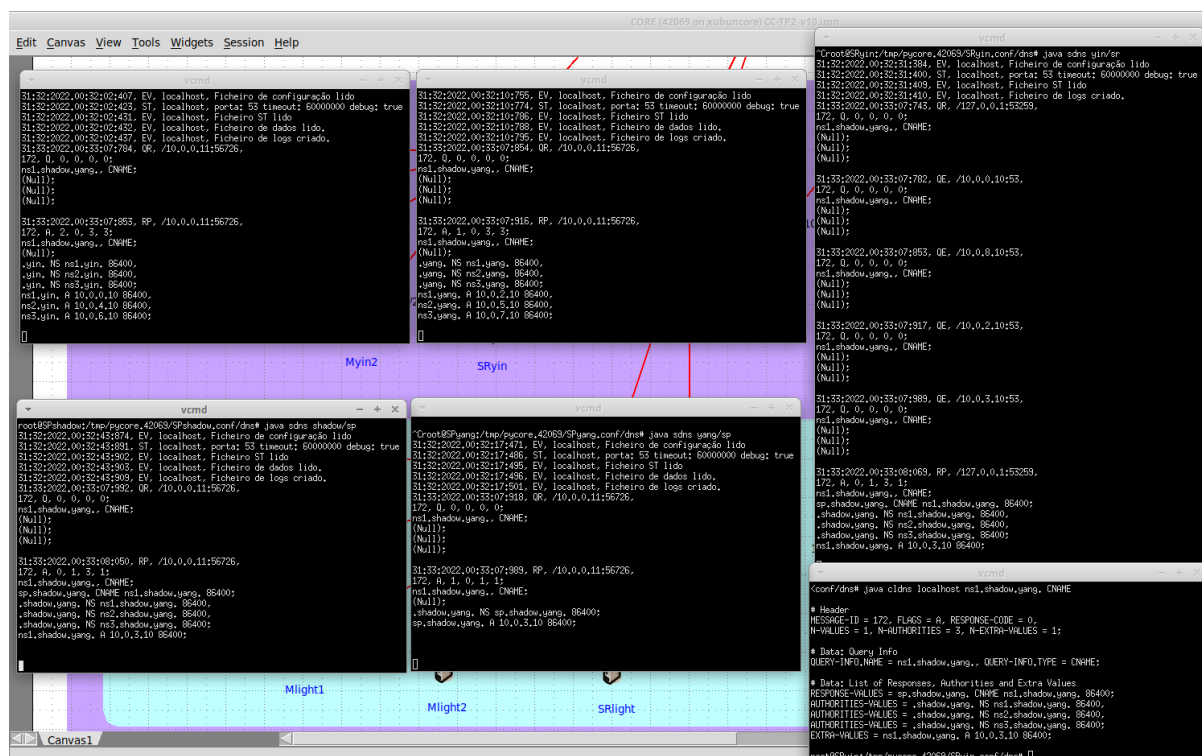


Fig 35. Query

→ Verifica que o código de resposta é 2 quando a máquina não existe no subdomínio

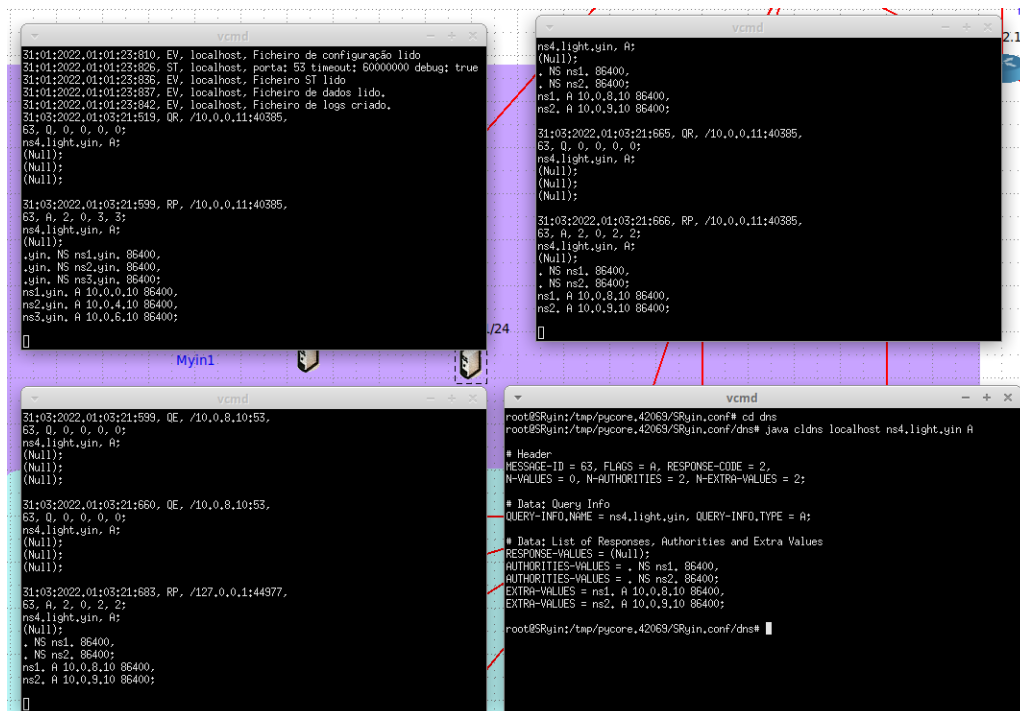
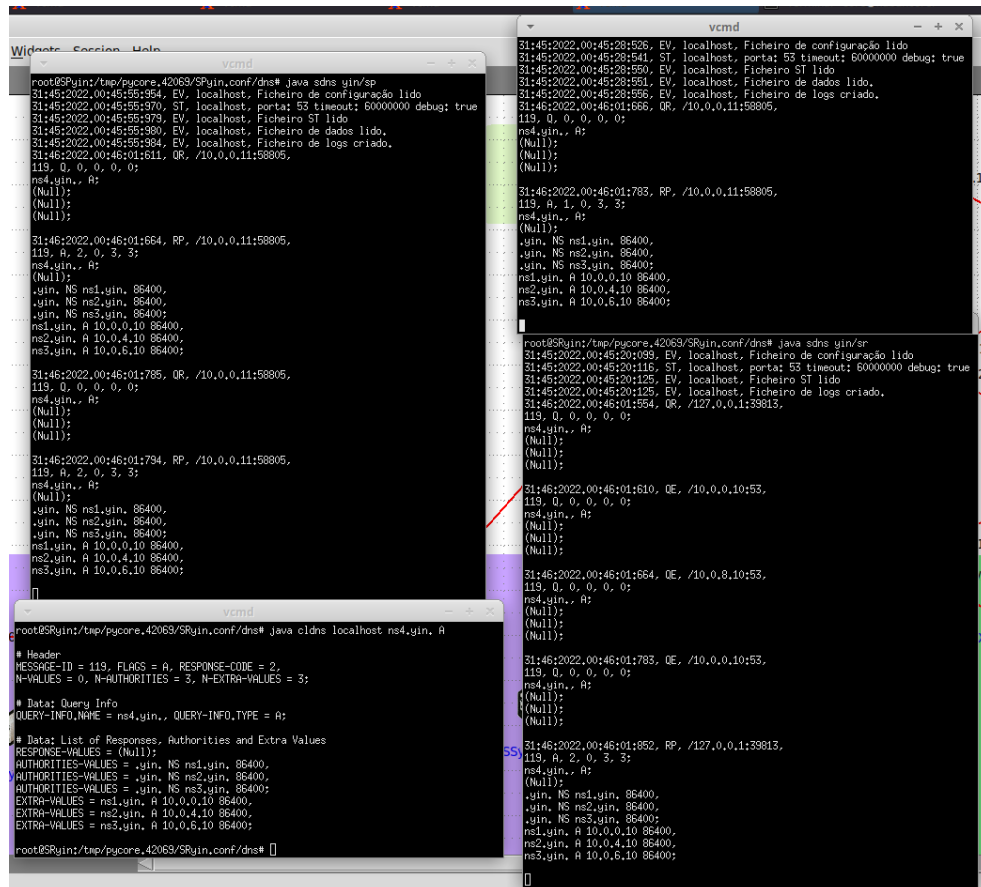


Fig 31. Query

→ Verifica que o código de resposta é 2 quando a máquina não existe no domínio



```
root@SRyint:/tmp/pycore.42069/SRyint/conf/dns# java sdns yin/sp
31:46:2022.00:46:55:354, EV, localhost, Ficheiro de configuração lido
31:46:2022.00:46:55:370, ST, localhost, porta: 53 timeout: 60000000 debug: true
31:46:2022.00:46:55:379, EV, localhost, Ficheiro ST lido
31:46:2022.00:46:55:380, EV, localhost, Ficheiro de dados lido.
31:46:2022.00:46:55:384, EV, localhost, Ficheiro de logs criado.
31:46:2022.00:46:01:611, QR, /10.0.0.11:58805,
119, 0, 0, 0, 0;
ns4.yin., A;
(Null);
(Null);
(Null);
31:46:2022.00:46:01:664, RP, /10.0.0.11:58805,
119, A, 2, 0, 3, 3;
ns4.yin., A;
(Null);
.yin. NS ns1.yin. 86400;
.yin. NS ns2.yin. 86400;
.yin. NS ns3.yin. 86400;
ns1.yin. A 10.0.0.10 86400;
ns2.yin. A 10.0.4.10 86400;
ns3.yin. A 10.0.6.10 86400;
31:46:2022.00:46:01:785, QR, /10.0.0.11:58805,
119, 0, 0, 0, 0;
ns4.yin., A;
(Null);
(Null);
(Null);
31:46:2022.00:46:01:784, RP, /10.0.0.11:58805,
119, A, 2, 0, 3, 3;
ns4.yin., A;
(Null);
.yin. NS ns1.yin. 86400;
.yin. NS ns2.yin. 86400;
.yin. NS ns3.yin. 86400;
ns1.yin. A 10.0.0.10 86400;
ns2.yin. A 10.0.4.10 86400;
ns3.yin. A 10.0.6.10 86400;
[
root@SRyint:/tmp/pycore.42069/SRyint/conf/dns# java cldns localhost ns4.yin. A
# Header
MESSAGE-ID = 119, FLAGS = A, RESPONSE-CODE = 2,
N-VALUES = 0, N-AUTHORITIES = 3, N-EXTRA-VALUES = 3;
# Data: Query Info
QUERY-INFO.NAME = ns4.yin., QUERY-INFO.TYPE = A;
# Data: List of Responses, Authorities and Extra Values
RESPONSE-VALUES = (Null);
AUTHORITIES-VALUES = .yin. NS ns1.yin. 86400;
AUTHORITIES-VALUES = .yin. NS ns2.yin. 86400;
AUTHORITIES-VALUES = .yin. NS ns3.yin. 86400;
EXTRA-VALUES = ns1.yin. A 10.0.0.10 86400;
EXTRA-VALUES = ns2.yin. A 10.0.4.10 86400;
EXTRA-VALUES = ns3.yin. A 10.0.6.10 86400;
root@SRyint:/tmp/pycore.42069/SRyint/conf/dns#
```

```
vcmd
31:46:2022.00:46:28:526, EV, localhost, Ficheiro de configuração lido
31:46:2022.00:46:28:541, ST, localhost, porta: 53 timeout: 60000000 debug: true
31:46:2022.00:46:28:550, EV, localhost, Ficheiro ST lido
31:46:2022.00:46:28:561, EV, localhost, Ficheiro de dados lido.
31:46:2022.00:46:28:566, EV, localhost, Ficheiro de logs criado.
31:46:2022.00:46:01:666, QR, /10.0.0.11:58805,
119, 0, 0, 0, 0;
ns4.yin., A;
(Null);
(Null);
(Null);
31:46:2022.00:46:01:783, RP, /10.0.0.11:58805,
119, A, 1, 0, 3, 3;
ns4.yin., A;
(Null);
.yin. NS ns1.yin. 86400;
.yin. NS ns2.yin. 86400;
.yin. NS ns3.yin. 86400;
ns1.yin. A 10.0.0.10 86400;
ns2.yin. A 10.0.4.10 86400;
ns3.yin. A 10.0.6.10 86400;
[
root@SRyint:/tmp/pycore.42069/SRyint/conf/dns# java sdns yin/sr
31:46:2022.00:46:20:099, EV, localhost, Ficheiro de configuração lido
31:46:2022.00:46:20:116, ST, localhost, porta: 53 timeout: 60000000 debug: true
31:46:2022.00:46:20:125, EV, localhost, Ficheiro ST lido
31:46:2022.00:46:20:125, EV, localhost, Ficheiro de dados lido.
31:46:2022.00:46:01:854, QR, /127.0.0.1:39813,
119, 0, 0, 0, 0;
ns4.yin., A;
(Null);
(Null);
(Null);
31:46:2022.00:46:01:610, QR, /10.0.0.10:53,
119, 0, 0, 0, 0;
ns4.yin., A;
(Null);
(Null);
(Null);
31:46:2022.00:46:01:654, QR, /10.0.8.10:53,
119, 0, 0, 0, 0;
ns4.yin., A;
(Null);
(Null);
(Null);
31:46:2022.00:46:01:783, QR, /10.0.0.10:53,
119, 0, 0, 0, 0;
ns4.yin., A;
(Null);
(Null);
(Null);
31:46:2022.00:46:01:852, RP, /127.0.0.1:39813,
119, A, 2, 0, 3, 3;
ns4.yin., A;
(Null);
.yin. NS ns1.yin. 86400;
.yin. NS ns2.yin. 86400;
.yin. NS ns3.yin. 86400;
ns1.yin. A 10.0.0.10 86400;
ns2.yin. A 10.0.4.10 86400;
ns3.yin. A 10.0.6.10 86400;
```

Fig 31. Query