

Universidade Federal de Uberlândia
Faculdade de Computação
Sistemas Distribuídos - GSI028 - 2022-01

André de Oliveira Alcântara, Gustavo Ramos; Pedro Henrique
Silva Santana; Victor Hugo Martins Alves

Relatório TCD
Implementação de um
Servidor de Nomes
utilizando python

Uberlândia - Minas Gerais
2023

1. Introdução	2
2. Inspiração	2
3. Implementação	3
4. Funcionamento	4
4.1. Passo a passo da resolução www.google.com.br :	4
4.2. Prints do passo a passo	5
5. Resultados	7
6. Limitações	7
6.1. Requisições por segundo	7
6.2. Tempo de resposta	8
6.3. Resposta em queda de servidor	8
6.4. Comportamento ao exceder limite de caracteres no hostname(1024 bytes)	9
7. Conclusão	10
8. Bibliografia	10

1. Introdução

O projeto tem como objetivo implementar um servidor de nomes estruturados de comunicação cliente-servidor de maneira síncrona, single thread, stateless e recursivo. Um servidor de nomes tem como função armazenar informações referentes a entidades; no caso deste trabalho, foi definido que o projeto seria similar ao modelo DNS (Domain Name Server), cujo objetivo do servidor seria armazenar URLs e traduzir os seus respectivos endereços IPs mediante solicitação do cliente;

As principais vantagens dele são:

Flexibilidade: Os servidores de nomes permitem que os recursos sejam renomeados ou movidos para outros endereços sem afetar os usuários finais ou os aplicativos. Isso é especialmente importante em ambientes de nuvem, onde os recursos são frequentemente escalados ou movidos para balancear a carga.

Escalabilidade: Os servidores de nomes podem ser configurados para distribuir as solicitações entre vários servidores, o que aumenta a capacidade de processamento e a disponibilidade dos recursos.

Segurança: Os servidores de nomes podem ser configurados para restringir o acesso a recursos específicos, o que aumenta a segurança da rede.

Melhoria de desempenho: Os servidores de nomes mantêm cópias locais das respostas de DNS para solicitações que eles já responderam, o que aumenta o desempenho das consultas.

Em resumo, os servidores de nomes são essenciais para garantir que os recursos em uma rede sejam facilmente acessíveis, escaláveis, seguros e gerenciáveis. Eles permitem que os usuários e aplicativos acessem recursos usando nomes amigáveis em vez de endereços IP complexos.

2. Inspiração

A inspiração para a execução desse projeto foi o BIND (Berkeley Internet Name Domain) que é um software de servidor DNS open source amplamente utilizado, atualmente em sua versão BIND9. Ele funciona como um servidor de nome de domínio, traduzindo Host names em endereços IP.

Pode ser configurado como um servidor DNS recursivo, no qual, caso ele não tenha a resposta para uma solicitação em sua cache ou em sua zona configurada, ele fará uma busca recursiva através de outros servidores DNS até encontrar a resposta, e também pode ser configurado como um servidor autoritativo, respondendo com sua melhor resposta possível mas deixando a tarefa de realizar a chamada do próximo servidor autoritativo com o resolver do cliente. Além disso, ele pode ser configurado para atuar como um servidor de cache, mantendo uma cópia local das respostas DNS para solicitações que já respondeu, aumentando assim o desempenho das consultas.

Ademais, o BIND9 também suporta vários protocolos de segurança como o DNSSEC e TSIG para proteger as comunicações entre os servidores DNS e garantir a autenticidade das informações.

3. Implementação

A estrutura do servidor de nomes distribuídos é demonstrado na imagem abaixo:

Estrutura

Estrutura geral da implementação do Servidor de nomes Distribuído recursivo

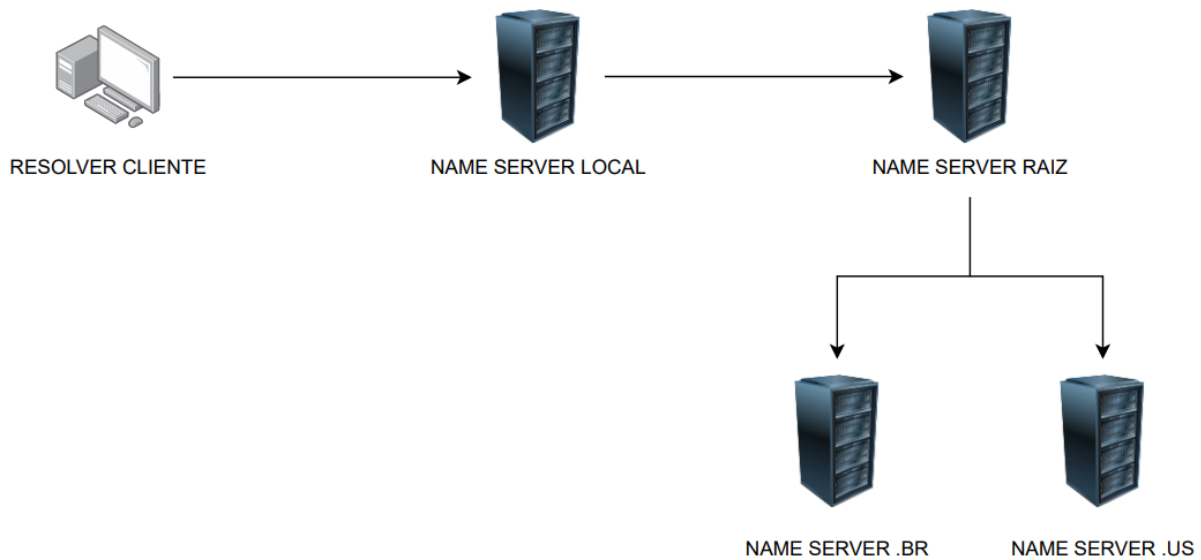



figura 01 - Estrutura do servidor de nomes

O *resolver cliente* é capaz de se conectar apenas com o servidor local, fazendo uma requisição e aguardando a resposta. Este servidor de nomes local contém um arquivo de cache com poucos nomes, caso ele consiga resolver, o hostname retorna o IP do servidor desejado, caso contrário, o local requisita para o servidor raiz que contém outro arquivo com uma coleção maior para resolver nomes em específico; se este também não conseguir resolver, é feita uma avaliação do hostname para identificar para qual outro servidor mandar (.br ou .us), que por sua vez possuem a mesma estrutura de resolução de nomes do raiz porém com uma coleção de nomes focada apenas no domínio que o servidor se propõe; entretanto caso não sejam capaz de resolver, retornam o código 404 para os servidores anteriores sequencialmente até o *resolver*. Essa estrutura do servidor de nomes é conhecida como recursiva.

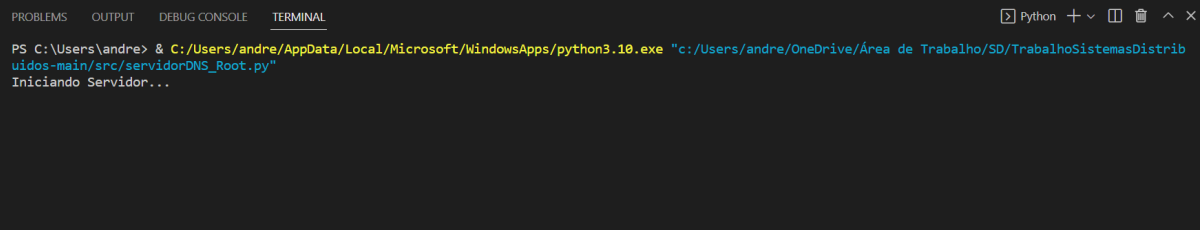
A conexão entre os servidores é feita através do módulo socket que foi configurado para utilizar UDP e IPV4.

4.2. Prints do passo a passo



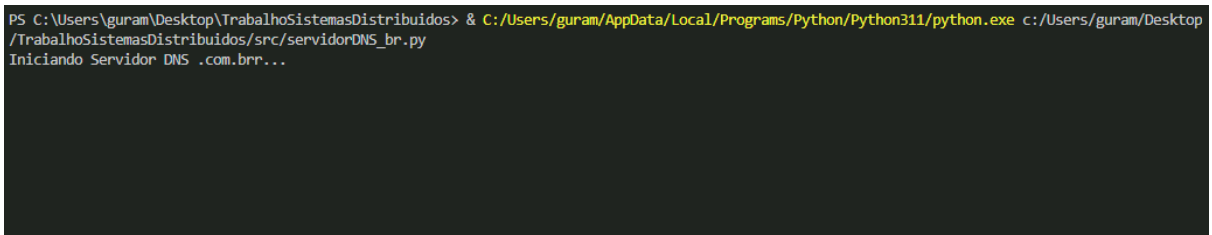
```
Iniciando Servidorr...
```

figura 03 - Iniciando servidor Local



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python + - [ ] [ ] ^ x
PS C:\Users\andre> & C:/Users/andre/AppData/Local/Microsoft/WindowsApps/python3.10.exe "c:/Users/andre/OneDrive/Área de Trabalho/SD/TrabalhoSistemasDistribuidos-main/src/servidorDNS_Root.py"
Iniciando Servidor...
```

figura 04 - Iniciando servidor Root



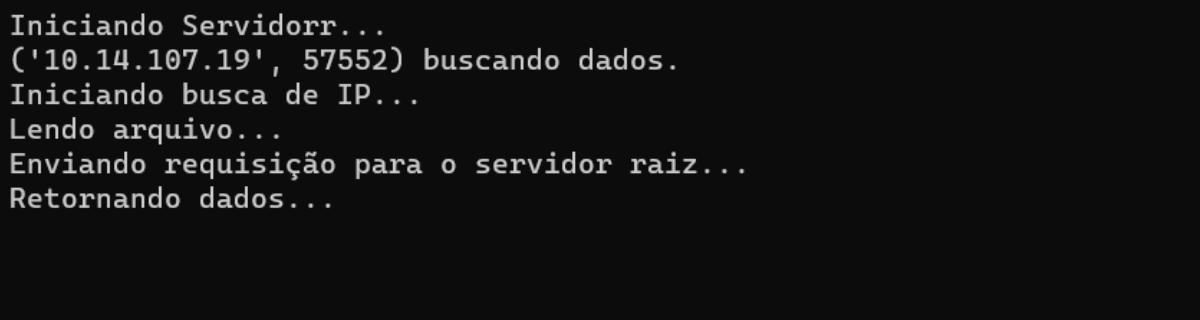
```
PS C:\Users\guram\Desktop\TrabalhoSistemasDistribuidos> & C:/Users/guram/AppData/Local/Programs/Python/Python311/python.exe c:/Users/guram/Desktop/TrabalhoSistemasDistribuidos/src/servidorDNS_br.py
Iniciando Servidor DNS .com.br...
```

figura 05 - Iniciando servidor .br



```
Enviando requisição...
-----
tempo de resposta: 1.305389165878296 seg
IP para o dominio www.google.com.br:172.217.160.238
-----
Continuar? (s/n)
```

figura 06 - Envio de requisição via Cliente para servidor de nomes



```
Iniciando Servidorr...
('10.14.107.19', 57552) buscando dados.
Iniciando busca de IP...
Lendo arquivo...
Enviando requisição para o servidor raiz...
Retornando dados...
```

figura 07 - Servidor local recebendo requisição, encaminhando para o root e recebendo os dados em recursão

5. Resultados

O procedimento detalhado no tópico 4 resulta na resolução dos domínios devolvendo ao cliente o endereço IP. No primeiro momento, foi tomado como consenso pela equipe que o servidor seria apenas para consulta, sendo possível a expansão de suas funções caso necessário.

Como este servidor é uma aplicação que possui sua esquematização envolvendo 4 máquinas domésticas conectadas em uma rede em comum, o mesmo teria dificuldades para ser implementado em esquemas com grandes números de clientes, gerando uma fila de requisições gerenciada pela própria biblioteca em uso no projeto e, por assim, elevados tempos de resposta.

6. Limitações

Enquanto eram executados os testes do servidor, a equipe identificou certas limitações que poderiam interferir no desempenho do sistema, assim, testes de exaustão foram efetuados para mensurar essas limitações. Todos os testes foram executados na rede da universidade, logo estão sujeitos às limitações da própria rede e ao tráfego da mesma, ressaltando que o projeto foi construído utilizando computadores domésticos de diferentes marcas e configurações. A velocidade dos processos e das solicitações também foi comprometida devido ao hardware limitado das máquinas e, na maioria das vezes, a execução do servidor ser concomitante com outras aplicações. O teste realizado em uma rede privada promoveu melhores resultados, se comparado com a rede pública da universidade. Apesar de todos percalços, tivemos êxito com a execução do trabalho, cumprindo com aquilo que havia sido proposto inicialmente.

6.1. Requisições por segundo

LOCAL		ROOT		LEAF	
nº	Requests	nº	Requests	nº	Requests
1	281	1	8,5	1	1,6
2	270	2	8,3	2	7,3
3	274	3	28	3	1,6
4	241	4	45,5	4	16,9
5	263	5	28,5	5	1,7
6	260	6	49,4	6	16,5
7	283	7	36,9	7	2,1
8	270	8	28,1	8	24,5
9	279	9	15,2	9	1,7
10	284	10	37,8	10	8,2
média	270,5	média	28,62	média	8,21

figura 10 - tabelas que indicam o número de requisições por segundo em 10 tentativas nos 3 tipos de servidores(*Local*, *Root* e *Leaf*)

O número de requisições por segundo respondidas pelos servidores foram testadas e uma média foi coletada, quando o servidor local possuía a resolução em cache a média atingida foi de 270,5 requisições por segundo. Quando a resolução estava no servidor Root a média de requisições por segundo foi de 28,62. Quando a resolução estava em um dos

servidores folha (.br ou .us) a média de requisições por segundo foi de 8,21. Essas métricas de desempenho são relativamente baixas, de forma a esperar que em múltiplas requisições, em especial nos servidores de mais baixa hierarquia, aconteçam atrasos no tempo de resposta do servidor.

6.2. Tempo de resposta

LOCAL		ROOT		LEAF	
nº	tempo	nº	tempo	nº	tempo
1	0,0036	1	0,1176	1	0,625
2	0,0037	2	0,1205	2	0,137
3	0,0036	3	0,0357	3	0,625
4	0,0041	4	0,022	4	0,0592
5	0,0038	5	0,0351	5	0,5882
6	0,0038	6	0,0202	6	0,0606
7	0,0035	7	0,0271	7	0,4762
8	0,0037	8	0,0356	8	0,0408
9	0,0036	9	0,0658	9	0,5882
10	0,0035	10	0,0265	10	0,122
média	0,0037	média	0,0506	média	0,3322

figura 11 - tabelas que indicam o tempo(em segundos) que uma requisição leva para ser solucionada em 10 tentativas nos 3 tipos de servidores(*Local*, *Root* e *Leaf*)

O tempo de resposta para requisições atingiu valores abaixo de 3,7 milissegundos ao acessar o local, 50,6 milissegundos no root e 332,2 milissegundos para requisições nas camadas mais externas. Mesmo sendo valores baixos, em casos de múltiplos usuários requisitando ao mesmo tempo, principalmente conteúdos da coleção de servidores mais externos, é possível que aconteça atraso nas respostas devido à fila de requisição.

6.3. Resposta em queda de servidor

Em situação que ocorra a queda do servidor ou parte deste, as requisições do cliente não serão respondidas, neste caso foi identificado a necessidade de acrescentar um *timeout* de 20 segundos pela parte do cliente para constatar a falha na conexão e assim encerrar o processo do cliente.

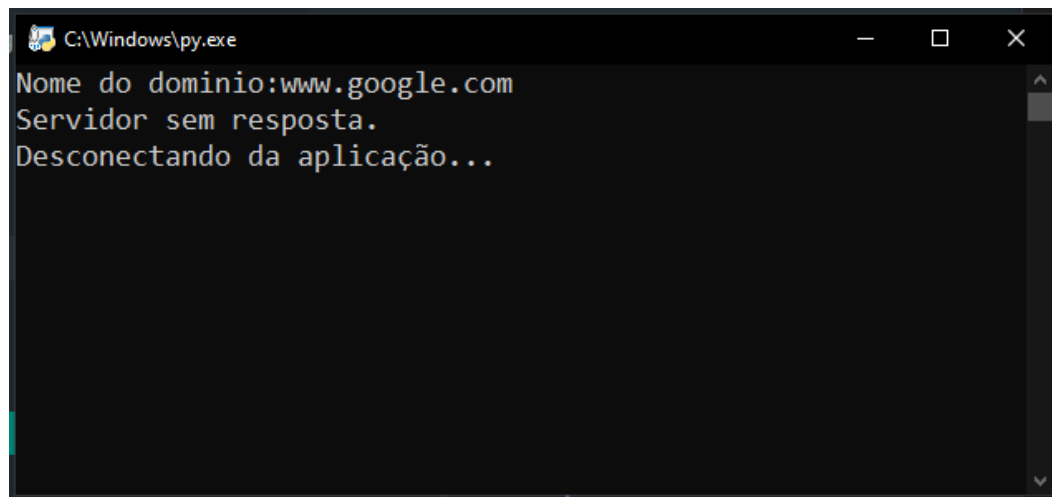


figura 12 - falha de conexão com servidor local

6.4. Comportamento ao exceder limite de caracteres no hostname(1024 bytes)

Devido a limitações postas na biblioteca utilizada para construir o socket de comunicação, as requisições são limitadas a um tamanho de 1024 bytes. Em uma situação em que um hostname de tamanho superior aos 1024 bytes é enviado do cliente para o servidor local, o servidor local é derrubado por tamanho limite excedido na requisição e o cliente recebe um timeout por limite de tempo na espera do servidor.

6.5. Solicitação de algo diferente de um domínio

Neste teste foram executadas consultas diferentes do previsto pelo sistema, sendo uma tentativa com um tipo inteiro e outra com uma tupla contendo um inteiro e um texto. em ambas as tentativas o sistema retornou o 404 para ilustrar que não foram encontrados dados para essas consultas. Devido o fato de cada consulta passar por um tratamento interno para identificar seus resultados, independente do tipo da entrada, as mesmas estarão sujeitas ao tratamento, assim percorrendo por todo o processo até os servidores mais distantes no conjunto.

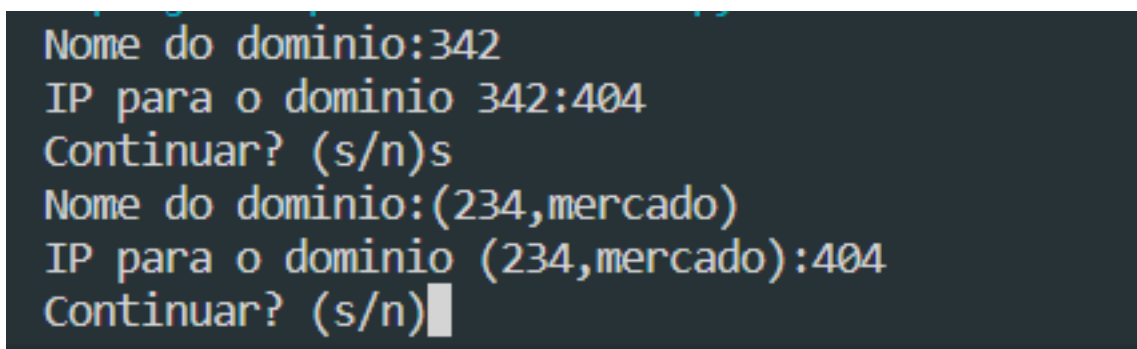


figura 13 - tentativas de requisições diferentes do previsto

7. Conclusão

Durante a etapa de construção do projeto, foram definidas necessidades primárias que o servidor deveria cumprir sendo elas: construir um servidor de nomes estruturados de comunicação cliente-servidor de maneira síncrona, single thread, stateless e recursivo.

Com esses tópicos definidos, foram definidas reuniões para estruturação do servidor. Com o andamento do projeto, a equipe concluiu as necessidades antes estipuladas e atingiu o que foi proposto anteriormente. Os maiores desafios se apresentaram na etapa de construção do servidor propriamente dito e em como aplicar as teorias estudadas nos capítulos para a prática de programação. Ao final, com o funcionamento do servidor, testes foram efetuados na rede pública da universidade e os resultados foram satisfatórios para a equipe.

8. Bibliografia

<https://github.com/VictorHugoMA/TrabalhoSistemasDistribuidos>

<https://www.isc.org/bind/>

<https://pythontic.com/modules/socket/socket>

https://bind9.readthedocs.io/en/v9_16_6/index.html

TANENBAUM, ANDREW S., STEEN, MAARTEN VAN; **Sistemas distribuídos: princípios e paradigmas**. Pearson, 2a ed., 2008.