SERVIDOR DNS LOCAL

GÓMEZ EDGAR, Universidad Central de Venezuela, Venezuela

Este documento presenta la implementación y evaluación de un servidor DNS local desarrollado en Python utilizando la biblioteca `dnserver`. El servidor permite la resolución de nombres de dominio para diferentes tipos de registros DNS (A, CNAME, MX, NS, TXT, SOA, SRV) en un entorno controlado. Se describe la configuración de zonas DNS mediante archivos TOML, la implementación del servidor y cliente de prueba, y se presentan los resultados de las consultas realizadas. La implementación demuestra los principios fundamentales del sistema de nombres de dominio y su funcionamiento en sistemas distribuidos.

CCS Concepts: • **Organización de sistemas informáticos → Sistemas Distribuidos**; *Servidor DNS*.

Additional Key Words and Phrases: Sistemas Distribuidos, servidor DNS

ACM Reference Format:
GÓMEZ EDGAR. 2025. SERVIDOR DNS LOCAL. *Proc. ACM Hum.-Comput. Interact.* 1, 1, Article 1 (December 2025), 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1. Introducción

El Sistema de Nombres de Dominio (DNS) es un componente crítico de la infraestructura de Internet que traduce nombres de dominio legibles por humanos en direcciones IP numéricas. Esta práctica se enfoca en implementar un servidor DNS local en Python para comprender los principios fundamentales de la resolución de nombres en sistemas distribuídos.

Los objetivos específicos de esta implementación son:

- Configurar un servidor DNS local que responda consultas para múltiples tipos de registros
- Implementar un cliente de prueba que realice consultas a los diferentes tipos de registros configurados
- Analizar las respuestas del servidor y validar su correcto funcionamiento
- Comprender la estructura y formato de los diferentes tipos de registros DNS

En esta implementación se configuraron los siguientes tipos de registros:

- **Registro A (Address):** Asocia un nombre de dominio con una dirección IPv4
- **Registro CNAME (Canonical Name):** Crea un alias para un nombre de dominio
- **Registro MX (Mail Exchange):** Especifica servidores de correo para el dominio
- **Registro NS (Name Server):** Identifica servidores DNS autoritativos para el dominio (apunta a servidores DNS que contienen los registros DNS oficiales del dominio)
- **Registro TXT (Text):** Contiene información textual, comúnmente usada para verificaciones
- **Registro SOA (Start of Authority):** Contiene información administrativa sobre la zona
- **Registro SRV (Service):** Especifica información sobre servicios disponibles

2. Implementación del Servidor DNS

2.1. Configuración del Entorno

Para la implementación se utilizó Python 3.x y las siguientes bibliotecas:

- `dnserver`: Para crear y gestionar el servidor DNS
- `dnslib`: Para manejar paquetes DNS

Author's Contact Information: GÓMEZ EDGAR, Universidad Central de Venezuela, Caracas, Venezuela.

2025. ACM 2573-0142/2025/12-ART1
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- `toml`: Para leer archivos de configuración en formato TOML

2.2. Archivo de Configuración de Zonas

Se creó un archivo `zones.toml` que contiene la definición de todos los registros DNS que el servidor responderá. Cada registro se define con su host, tipo y respuesta correspondiente.

```
[[zones]]
host = 'misitio.com'
type = 'A'
answer = '192.168.1.120'

[[zones]]
host = 'alias.com'
type = 'CNAME'
answer = 'result.com'

[[zones]]
host = 'email.com'
type = 'MX'
answer = ['whatever.com.', 5]

[[zones]]
host = 'example.com'
type = 'MX'
answer = ['mx2.whatever.com.', 10]

[[zones]]
host = 'example.com'
type = 'MX'
answer = ['mx3.whatever.com.', 20]

[[zones]]
host = 'group.com'
type = 'NS'
answer = 'ns1.group.com.'

[[zones]]
host = 'group.com'
type = 'NS'
answer = 'ns2.group.com.'

[[zones]]
host = 'text.com'
type = 'TXT'
answer = 'hello this is some text'
```

```
[[zones]]
host = 'soaex.com'
type = 'SOA'
answer = ['ns1.example.com', 'dns.example.com']

[[zones]]
host = 'testing.com'
type = 'TXT'
# because the next record exceeds 255 in length dnserver will automatically
# split it into a multipart record, the new lines here have no effect on that
answer = """
one long value: IICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAg
FWZUed1qcBziAsqZ/LzT2ASxJYuJ5sko1CzWFhFuxiluNnwKjSknSjanyYnm0vro4dhAtyiQ7O
PVROOaNy9Iyklvu91KuhbYi6l80Rrdnuq1yjM//xjaB6DGx8+m1ENML8PEdSFbKQbh9akm2bkN
w5DC5a8Slp7j+eEVHkgV3k3oRhkPcrKyoPVvniDNH+Ln7DnSGC+Aw5Sp+fhu5aZmoODhhX5/1m
ANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA26JaFWZUed1qcBziAsqZ/LzTF2ASxJYuJ5sk
"""

[[zones]]
host = '_caldavs._tcp.example.com'
type = 'SRV'
answer = [0, 1, 80, 'caldav']
```

2.3. Servidor DNS en Python

El servidor se implementó creando una instancia de DNSServer cargada desde el archivo de configuración TOML. Se configuró para escuchar en el puerto 5053.

Luego se realizan el llamado de los registro al servidor DNS con la libreria dnslib pasando el nombre y el tipo, luego se lee la respuesta del servidor DNS

```
from dnserver import DNSServer
from dnslib import DNSRecord
import time

server = DNSServer.from_toml('zones.toml', port=5053)
server.start()
assert server.is_running

queries = [
 ("misitio.com", "A"),
 ("alias.com", "CNAME"),
 ("email.com", "MX"),
 ("group.com", "NS"),
 ("text.com", "TXT"),
 ("soaex.com", "SOA"),
 ("testing.com", "TXT"),
```

```
   ("_caldavs._tcp.example.com", "SRV"),
  ]

  try:
   for name, qtype in queries:
    q = DNSRecord.question(name, qtype)
    resp = q.send("127.0.0.1", 5053, timeout=2)
    print(f"== {name} {qtype} ==")
    print(DNSRecord.parse(resp))
    time.sleep(0.1)
  finally:
   server.stop()
```

3. Resultados de la Ejecución

El servidor se ejecutó exitosamente, cargando 11 registros de zona desde el archivo de configuración. A continuación se presentan los resultados más relevantes de las consultas realizadas:

Cuadro 1. Resumen de respuestas DNS obtenidas

Dominio	**Tipo**	**Respuesta**
misitio.com	A	192.168.1.120 (TTL: 300)
alias.com	CNAME	result.com. (TTL: 300)
email.com	MX	whatever.com. con prioridad 5 (TTL: 300)
group.com	NS	ns1.group.com. y ns2.group.com. (TTL: 86400)
text.com	TXT	"hello this is some text"(TTL: 300)
soaex.com	SOA	ns1.example.com. dns.example.com. (TTL: 86400)
testing.com	TXT	Registro TXT largo dividido en múltiples partes
_caldavs._tcp.example.com	SRV	0 1 80 caldav (TTL: 300)

3.1. Análisis de las Respuestas

Registro A:. La consulta a `misitio.com` retornó correctamente la dirección IP 192.168.1.120 con un TTL de 300 segundos. Fig1

```
2025-12-28 17:42:20 [DNSHandler:ProxyResolver] Request: [127.0.0.1:49744] (udp) / 'misitio.com.' (A)
17:42:20: found zone for misitio.com.[A], 1 replies
2025-12-28 17:42:20 [DNSHandler:ProxyResolver] Reply: [127.0.0.1:49744] (udp) / 'misitio.com.' (A) / RRs: A
== misitio.com A ==
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28816
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;misitio.com.                   IN      A
;; ANSWER SECTION:
misitio.com.            300     IN      A       192.168.1.120
```

Fig. 1. Resolucion direccion IPV4

Registro CNAME:. El alias `alias.com` se resolvió exitosamente a `result.com`.

```
2025-12-28 17:42:21 [DNSHandler:ProxyResolver] Request: [127.0.0.1:49745] (udp) / 'alias.com.' (CNAME)
17:42:21: found zone for alias.com.[CNAME], 1 replies
2025-12-28 17:42:21 [DNSHandler:ProxyResolver] Reply: [127.0.0.1:49745] (udp) / 'alias.com.' (CNAME) / RRs: CNAME
== alias.com CNAME ==
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34137
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;alias.com.                    IN      CNAME
;; ANSWER SECTION:
alias.com.              300     IN      CNAME   result.com.
```

Fig. 2. Resolucion alias

Registro MX:. Para `email.com` se obtuvo un servidor de correo con prioridad 5, indicando el orden preferencial para entrega de correo.

```
2025-12-28 17:42:21 [DNSHandler:ProxyResolver] Request: [127.0.0.1:49746] (udp) / 'email.com.' (MX)
17:42:21: found zone for email.com.[MX], 1 replies
2025-12-28 17:42:21 [DNSHandler:ProxyResolver] Reply: [127.0.0.1:49746] (udp) / 'email.com.' (MX) / RRs: MX
== email.com MX ==
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60656
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;email.com.                     IN      MX
;; ANSWER SECTION:
email.com.              300     IN      MX      5 whatever.com.
```

Fig. 3. Resolucion de correos

Registro NS:. Se configuraron dos servidores de nombres para `group.com`, proporcionando redundancia.

```
2025-12-28 17:42:21 [DNSHandler:ProxyResolver] Request: [127.0.0.1:49747] (udp) / 'group.com.' (NS)
17:42:21: found zone for group.com.[NS], 2 replies
2025-12-28 17:42:21 [DNSHandler:ProxyResolver] Reply: [127.0.0.1:49747] (udp) / 'group.com.' (NS) / RRs: NS,NS
== group.com NS ==
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2107
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;group.com.                     IN      NS
;; ANSWER SECTION:
group.com.              86400   IN      NS      ns1.group.com.
group.com.              86400   IN      NS      ns2.group.com.
```

Fig. 4. Resolucion de servidores de nombre

Registro TXT:. El servidor manejó correctamente registros TXT largos, dividiéndolos automáticamente cuando excedieron el límite de 255 caracteres.

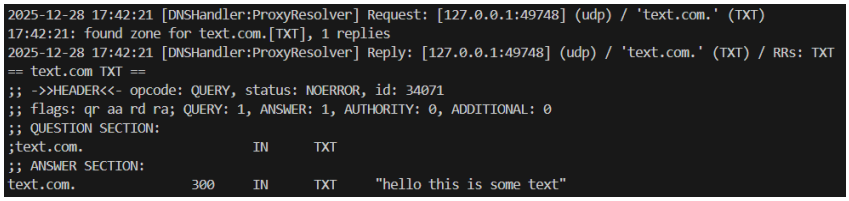

Fig. 5. Resolcion de texto corto

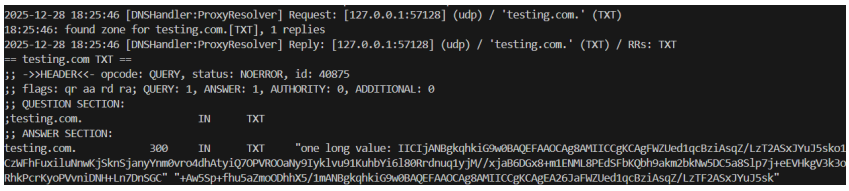

Fig. 6. Resolcion texto largo

Registro SOA:. Incluyó información completa sobre la zona, incluyendo serial number, tiempos de refresh y expire.

Fig. 7

Registro SRV:. Especificó correctamente los parámetros de servicio para el servicio CalDAV.

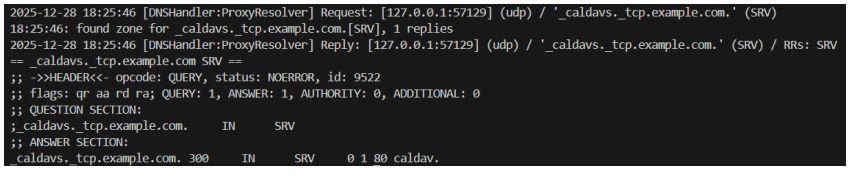

Fig. 8

4. Conclusión

Se implementó exitosamente un servidor DNS local en Python capaz de responder consultas para múltiples tipos de registros DNS. El servidor demostró ser funcional para propósitos educativos y de desarrollo, mostrando los principios fundamentales de la resolución de nombres en sistemas distribuídos.

La implementación confirmó la correcta interpretación y respuesta de los diferentes tipos de registros configurados, incluyendo el manejo automático de registros TXT largos. El uso de la biblioteca `dnserver` simplificó significativamente el desarrollo, permitiendo enfocarse en los aspectos conceptuales del DNS.