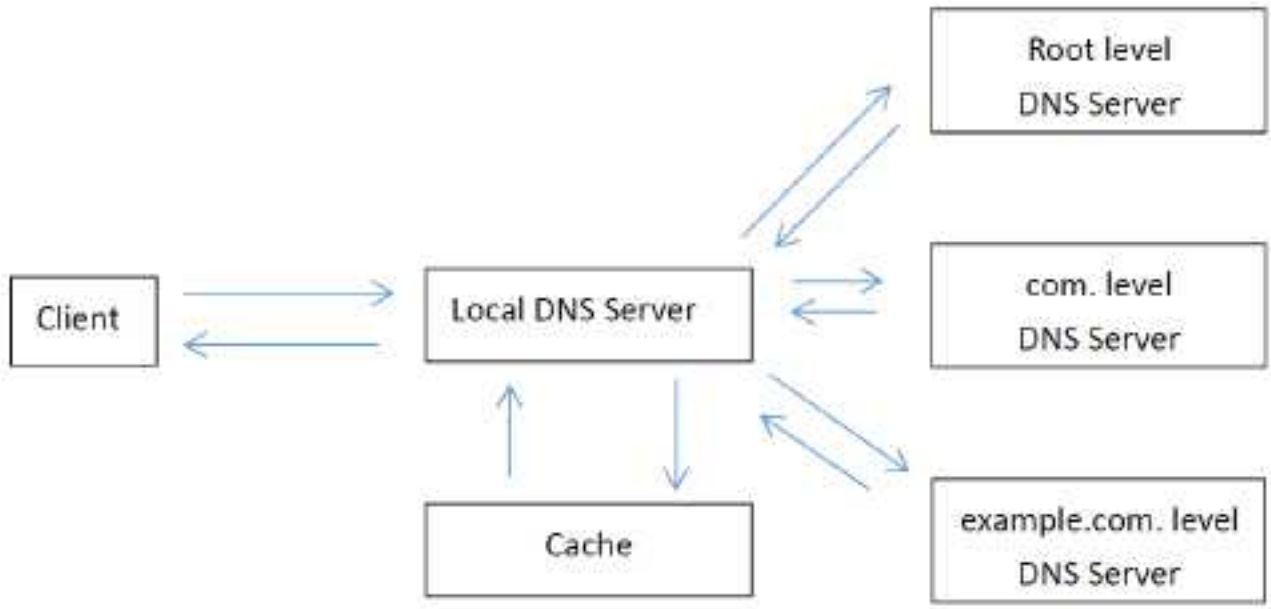
CS305 2022 Spring Assignment 1

A **local DNS server** is a server which is setup inside of a company, homes network, or residential ISP for address/name translations. The goal of this assignment is to implement a **local DNS server** in Python.

IMPORTANT NOTES:

- Your python code should be named as "LocalDNSServer.py" using UTF-8 as file encoding.
- The local DNS server should listen on port 5533 (do NOT change it!!!) and IP address 127.0.0.1 (do NOT change it!!!) .
- When querying, the RD in DNS message should be set to 0* (*or set the flag to be 0x0000*), which means that your DNS query is iterative.
- Comments in code is MUST.
- A basic template will be provided and you need to implement the functionalities in the **TODO** fields. You may choose to not use the template, but you need to confirm your solutions match the corresponding input and output as shown in the examples below.
- We will use dig as the client in this assignment.



Environments

- Python 3.9.7
- dnspython and dnslib libraries (optional, but recommended and used in the template)
- Your may use other libraries, but you must ensure that your query procedure is iterative

Implementation detail

1. The local DNS server should listen on port 5533 and IP 127.0.0.1. The main functionalities of the local DNS server include:

(1) Listen and accept DNS queries from client, support default query type: A type.

(2) Maintain a Cache: ① Cache should be human-readable, following the RR format: (name, value, type, ttl).

② A new query should be recorded if not in the cache already.

(3) Maintain a list of records about the domain names and IPv4 addresses of Root DNS servers.

(4) Search related DNS responses in cache according to the received query.

(5) Invoke DNS queries to other DNS servers if needed and cache the new response.

(6) Reply to client with the response.

2. Take a DNS query to "www.example.com" as an example what your need to implement is about:

   (1) The client sends a DNS query about "www.example.com" to the local DNS server.

   (2) The local DNS server searches its mapping in the cache.

   - if the mapping is found in Cache, the local DNS server replies to the client as a DNS response object.
   - If a mapping is not found, then a translation is resolved **iteratively** as follows:
     - (a) the local DNS server sends the query to **root level DNS Server**, and gets a DNS response message about the list of .com level DNS servers.
     - (b) the local DNS server then sends a query to **.com level DNS server**, and gets the DNS response message about the list of example.com level name servers.
     - (c) the local DNS server then sends the query to **example.com. level DNS Server**, and gets the DNS response message about the mapping of the original DNS query.
     - (d) the local DNS server caches the mapping in cache.
     - (e) the local DNS server replies to the client with the mapping.

3. Grading criteria
   - Basic iterative query to get a name/address translation (80)
   - Caching (20)
   - Multi-threading local DNS server and concurrency control (bonus, up to 20)
     - Implement a multi-thread version local DNS server that can serve multiple queries concurrently.
     - Provide at least one implementations to tackle the problem when several clients access the server in the same time, which means that you should ensure the thread safety in cache, since it could be overwritten by multi threads at the same time.

5. **NOTE again: in the DNS message's flag, RD should be set to 0 which means it does not query recursively!**

Test detail

Input format

There are two places that you need to input.

- The first one is when you run the LocalDNSServer.py , you need to input the ip address of your network interface card.

- The second one is when you open dig, you should input a dig DNS request, e.g.,

  dig @"your ipv4 address" "queried domain name" a -p "your port"

  Here please specify the port as 5533

Output format

- The output format is standard DNS message format, you can use dnslib library to construct and resolve it (we have this example in the template).
- When the DNS mapping is read from cache your should print "read from cache" on the console ("" is not need)

Test process

① Run LocalDNSServer.py.

② Enter IP address (you can use ipconfig to search your IP address) and port (this port is arbitrary but you should avoid conflict with other process) in console. If success, it will print the information about the root DNS server.

This means that you can use this IP address and port to complete and resolve a basic DNS query.

③ Open another command line and input the command:

dig @127.0.0.1 www.baidu.com a -p 5533

Then the console should print your local DNS server's response.

and if it is read from cache, you should print "read from cache" in the python console where your server runs

If you resolve a domain name and get a result in cname type, you should keep that result and include it in the response

The above output example shows

- when you search www.baidu.com and find a cname type record of www.a.shifen.com,
- you need to print www.a.shifen.com and then use www.a.shifen.com to query the final response, which is shown as the above screenshot.

④ We will test about more than 4 testcases continuously.

What to submit

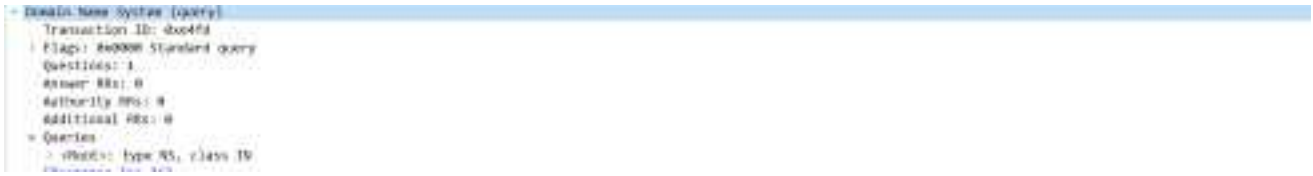
- You should provide your implementation in the file LocalDNSServer.py and another file such as the cache if needed.
- Please provide detailed comments in your code to explain different parts of your code. You need to at least illustrate in your implementation about basic part, cache part, and concurrency part (if any).
- A PDF file that includes some necessary screenshots to show your code works and some brief explanations.

TIPS

The below WireShark screenshots show the process about resolving a DNS query **iteratively**.

① query 8.8.8.8 and get records of root DNS server in NS type

query:

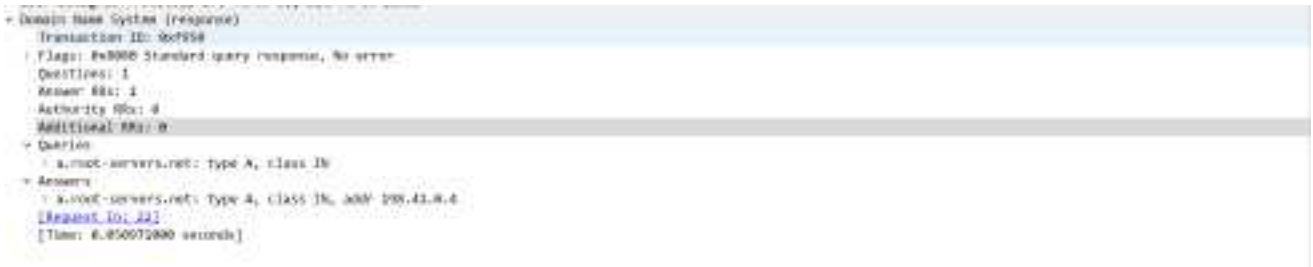


answer:

② query 8.8.8.8 and get IP address of root DNS server

query:

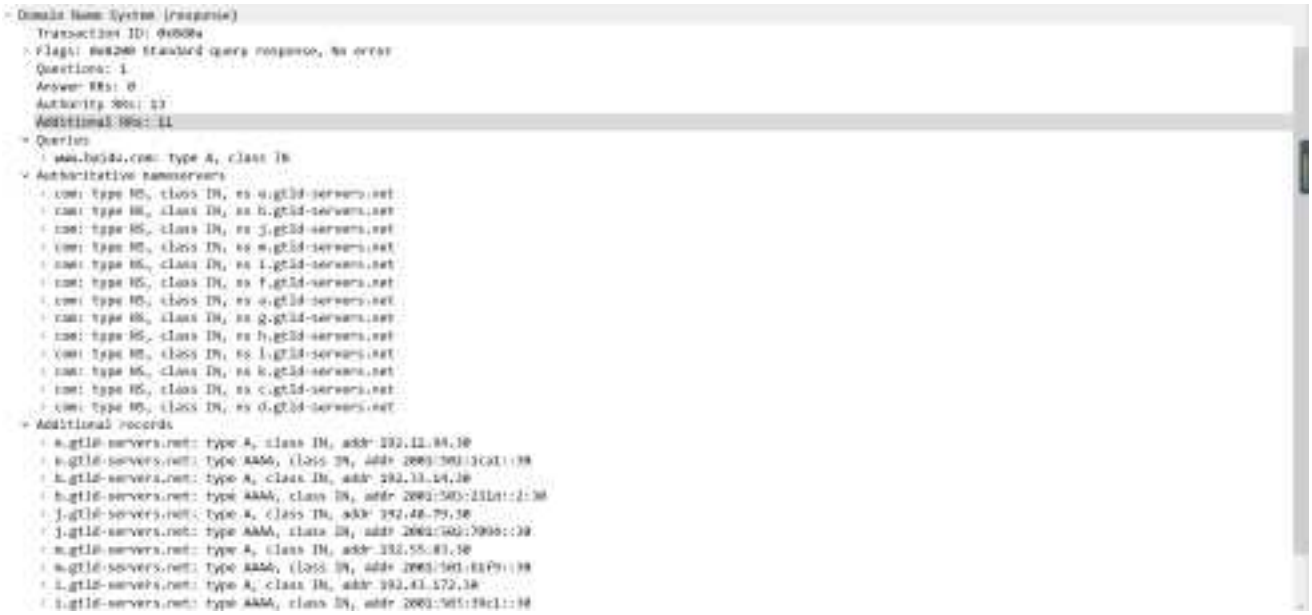
answer:



③ query root DNS server and get IP address of top-level DNS server

query:

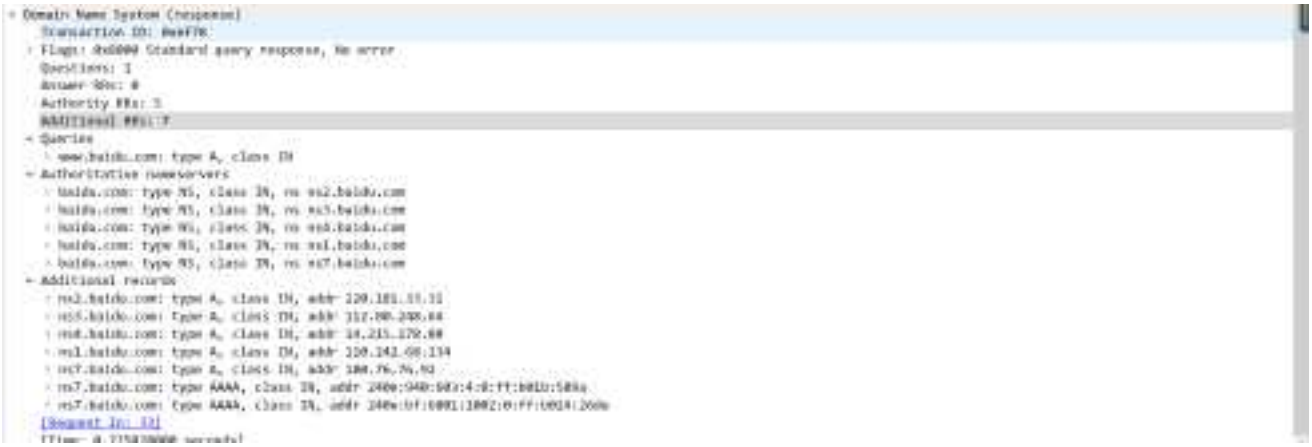
answer:



④ query top-level server and try to find authoritative domain name server

query:

answer:

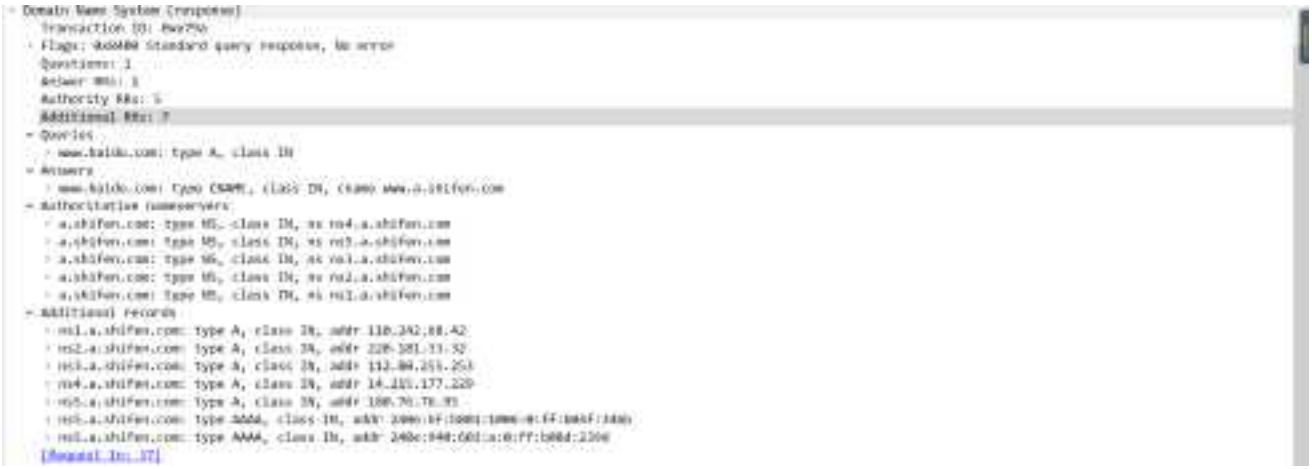


Hint: in this step you may find many potential routes or even not any mappings for further search. One recommendation is that you can save all search paths as a tree and make a depth-first search (DFS) in that tree.

⑤ by iteratively querying, finally get a mapping of A type

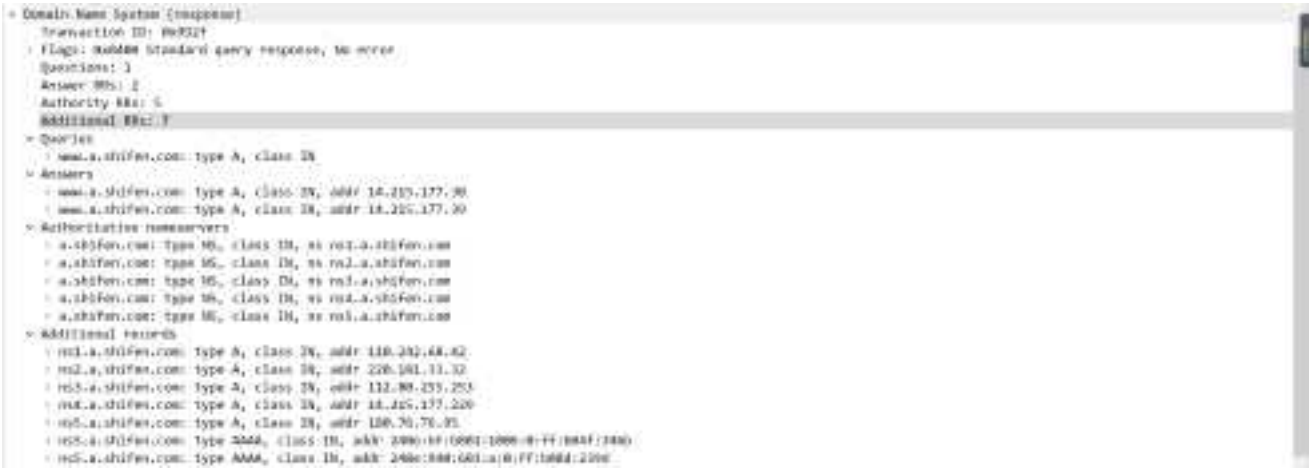
query:

answer:



query:

answer:



Hint: In this step, you may sometimes get a record in CNAME: you should use the CNAME to continue query and consequently find a record in A type

Another test examples

Results may different every time!

Please NOTE that if the result is read from cache, you should print "read from cache" in the console where your server runs

```
dig @127.0.0.1 www.sustech.edu.cn a -p 5533

; <<>> DiG 9.16.26 <<>> @127.0.0.1 www.sustech.edu.cn a -p 5533
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52651
;; flags: qr rd ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;www.sustech.edu.cn.            IN      A

;; ANSWER SECTION:
www.sustech.edu.cn.     60      IN      CNAME   www.sustech.edu.cn.w.cdngslb.com.
www.sustech.edu.cn.w.cdngslb.com. 60 IN A       125.77.142.122

;; Query time: 1748 msec
;; SERVER: 127.0.0.1#5533(127.0.0.1)
;; WHEN: Sat Mar 05 12:51:21 ;; MSG SIZE  rcvd: 98
```

```
dig @127.0.0.1 www.bilibili.com a -p 5533

; <<>> DiG 9.16.26 <<>> @127.0.0.1 www.bilibili.com a -p 5533
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62872
;; flags: qr rd ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;www.bilibili.com.              IN      A

;; ANSWER SECTION:
www.bilibili.com.       60      IN      CNAME   s.w.bilicdn1.com.
s.w.bilicdn1.com.       60      IN      A       47.103.24.173

;; Query time: 1565 msec
;; SERVER: 127.0.0.1#5533(127.0.0.1)
;; WHEN: Sat Mar 05 13:07:42 ;; MSG SIZE  rcvd: 77
```

```
>dig @127.0.0.1 www.github.com a -p 5533

; <<>> DiG 9.16.26 <<>> @127.0.0.1 www.github.com a -p 5533
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43045
;; flags: qr rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;www.github.com.                        IN      A

;; ANSWER SECTION:
www.github.com.         60      IN      A       20.205.243.166

;; Query time: 608 msec
;; SERVER: 127.0.0.1#5533(127.0.0.1)
;; WHEN: Sat Mar 05 12:50:19 ;; MSG SIZE  rcvd: 48
```

when read from cache

```
D:\anaconda\envs\py397\python.exe D:/CS305-Project-simple_p2p_simulation/example_complete.py
Enter your ip:
Enter your port:
198.41.0.4
a.root-servers.net.
read from cache
```

```
>dig @127.0.0.1 www.baidu.com a -p 5533

; <<>> DiG 9.16.26 <<>> @127.0.0.1 www.baidu.com a -p 5533
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29091
;; flags: qr rd ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;www.baidu.com.                 IN      A

;; ANSWER SECTION:
www.baidu.com.          60      IN      CNAME   www.a.shifen.com.
www.a.shifen.com.       60      IN      A       14.215.177.38

;; Query time: 0 msec
;; SERVER: 127.0.0.1#5533(127.0.0.1)
;; WHEN: Sat Mar 05 14:11:53 ;; MSG SIZE  rcvd: 74
```