Lab: Building a DNS Sinkhole Server

The DNS service is one of the fundamental elements of the internet. However, using your ISP DNS servers, as your local DNS server, exposes you to censorship, while using a public DNS server (like 8.8.8.8) exposes your DNS queries to 3rd parties and their privacy policies. A popular way to regain control over your privacy is to run your own DNS resolver server.
While at it, you can also implement an additional useful feature: blocking unwanted domains. E.g., advertisements, tracking, gambling, adult content, and more. A popular example of such a server can be found in the Pi-hole project.

In this lab, you will implement a DNS Sinkhole server that blocks resolution for a given list of domains (hence the name “sinkhole”) and iteratively resolves any request for domains not included in that block-list.

Usage Example (`+noadflag +noedns` are used to suppress new dig functionality)
Assume that your DNS server is listening on port 5300:

```
$ dig @127.0.0.1 -p 5300 +noadflag +noedns www.facebook.com

; <<>> DiG 9.10.6 <<>> @127.0.0.1 -p 5300 +noadflag +noedns
www.facebook.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36549
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

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

;; ANSWER SECTION:
www.facebook.com.   3600 IN    CNAME     star-mini.c10r.facebook.com.

;; Query time: 201 msec
;; SERVER: 127.0.0.1#5300(127.0.0.1)
;; WHEN: Thu Dec 17 20:54:23 IST 2020
;; MSG SIZE  rcvd: 91
```

Dig installation reminder

- dig is usually available by default on Linux and MacOS
- dig installation manual: https://www.digitalocean.com/docs/networking/dns/resources/use-dig/

DNS Server (70%)

Your DNS server needs to be able to answer recursive A type DNS queries without relying on an OS/ISP/Public resolver. The DNS server that you implement should get a DNS query packet according to the DNS RFC and iteratively finds an answer for it, starting by sending a query to a random root server.
The high-level process should be:

Listen on **UDP port 5300**, and for every incoming packet do the following:

1. assume it is a valid DNS query (class INET, type A)
2. send it to a randomly chosen root server
3. while (the response code is NOERROR) AND (the number of ANSWER records is 0) AND (the number of AUTHORITY > 0)
    - Send the query to the first name server in the AUTHORITY section
4. send the final response to the client
    - with slight changes

Assumptions & Hints

1. You may assume the input is correct (a valid DNS query of type A only)
2. You may assume all DNS requests will be answered (no need to implement timeouts and retries)
3. You may use the nameserver domain name when sending the query
    a. i.e., you don't need to find it's IP in the glue (ADDITIONAL) records or implement recursive resolution
4. You can limit the number of iterations to 16 (to avoid rare cases of loops)
5. Use **dig** and **wireshark** to test your code
6. You don't need to parse the entire DNS packet or generate DNS packets from scratch
    a. Refresh your Java bitwise operators skills
    b. Remember that network packets are in **Big Endian** notation
7. When reading records from the authority section, note the name in the RDATA section may be compressed (RFC1035 section 4.1.4)
8. Don't forget to fix the flags when sending the response to the client
    a. you can compare your response's flags to the ones received from a public DNS (like 8.8.8.8)

Block list (30%)

Your program should accept, as an optional argument, a path to a text file containing a list of domains that have to be blocked. You can use the attached file (blocklist-example.txt) for testing. When your server gets a query where the domain name that needs to be resolved is in the specified blocklist file, it should return a response with an NXDOMAIN error (AKA “Name Error”, RCODE=3).

Assumptions & Hints

1. You may assume the blocklist file, if specified as an argument, is valid (exists, containing one valid domain name per line)
2. You may assume that the blocklist fits in memory
3. You might want to load the list into an efficient data structure (like the ones implementing Set) before starting the server
4. Note that you only need to make small changes to the query to create a “blocked” response
    a. You can send an invalid query to 8.8.8.8 and watch the difference between the query and the response on Wireshark

Submission guidelines

Compilation & Execution

1. Your code must compile with Java 11
    a. Submissions that fail to compile would get a 0 grade
    b. Submissions with compilation warnings will lose points
    c. We highly recommend to manually compile your code in the command line (to verify compiler version and no warnings)
2. You are **not allowed** to use java ready-made classes to Parse/Compose the DNS packets, or use any external library (e.g., apache.commons, guava, etc.)
3. You **may** have hard-coded values in your code, as long as they are well documented (**Best**: using a “static final” with a meaningful name, **Acceptable**: inline comments)
4. You **should** print meaningful errors to the STDERR (e.g., using `System.err.printf`)
5. You **should not** print stack traces or have any unhandled exceptions thrown at the user
6. You must put your classes in the following package: “**il.ac.idc.cs.sinkhole**”
7. You must put your main function in **SinkholeServer.java**

The following commands will be used to build and run your code:

```
$ javac -d out/ -Xlint src/il/ac/idc/cs/sinkhole/*.java

$ java -cp out il.ac.idc.cs.sinkhole.SinkholeServer
$ java -cp out il.ac.idc.cs.sinkhole.SinkholeServer blocklist.txt
```

Administration

1. Submission **in pairs** by moodle by Friday, 15.1.2021
2. The file name must be **lab_[username1]_[username2].zip**
3. **Do not send any .class files in your zip**.
4. Misformatted files will not be accepted.
5. Attach a README file (in the .zip) containing
    a. The names & usernames of both students
    b. For each file submitted: a single line describing its purpose
6. **Use** Piazza to ask any questions regarding this lab
    a. **Do not** upload any solution code to Piazza

Good luck!