Homework 1: All about DNS
CSE 534, Fall 2018
Instructor: Aruna Balasubramanian
Due date: 09/26/2017, 9.00pm

Almost everything on the Internet involves DNS resolution. If DNS is slow, the performance of your application is going to suffer. The goal of this homework is write your own DNS resolver, and compare its performance with other existing DNS resolvers. You get to write your own "dig" tool and a "DNSSEC" resolver.

Part A (50 points)

You will be implementing a DNS resolver. In response to an input query, the resolver will first contact the root server, then the top-level domains, all the way down to the corresponding name server to resolve the DNS query.

You can assume that you have access to a library that can resolve a single iterative DNS query. The set of libraries that you may use are given in the Appendix. The libraries also perform complete DNS resolution, but you are not allowed to use that.

1. You can access the IP address of the root servers from https://www.iana.org/domains/root/servers. Your server program must use this list to find a working root server. When the request to the root is not answered, your code should move on to the next server in the list.
2. Build a "dig"-like tool called "mydig". The mydig tool takes as input: (1) name of the domain you want to resolve and (2) type of DNS resolution. The type represents the DNS query type. Your tool should work for the following types: "A", "NS", "MX".

   When run as "./mydig <name> <type>", your tool should display the results "similar" to the results from the dig tool *(the additional section is not needed, just the resolved IP address).*

   The output should be of the following form

   ./mydig www.cnn.com A

```
QUESTION SECTION:
www.cnn.com.            IN   A

ANSWER SECTION:
www.cnn.com.       262  IN   A    151.101.209.67

Query time: 24 msec
WHEN: Fri Feb 2 10:26:27 2018
```

```
MSG SIZE rcvd: 84
```

A similar output is expected for MX and NS records. In some cases, when using MX and NS records, the dig tool returns the answers in the "Authority" section instead of Answer section. It is ok to return the authority section results in that case.

3. In some cases, you may need additional resolution. For example, google.co.jp will often not resolve to an IP address in one pass. Your tool should handle such cases.

Along with the code, you need to submit an output file called "mydig_output.txt", that contains the expected output for running your mydig program for the 3 types mentioned above. Please specify the input to your program.

PART B (40 points)

DNSSEC is a recent extension to the DNS protocol that guarantees the integrity of DNS. It was officially deployed in July of 2010 (though drafts started as early as 2004) and requires work from both DNS name servers as well as DNS resolvers to function properly. First read up on DNSSEC. This page provides a good first step: https://www.cloudflare.com/dns/dnssec/how-dnssec-works/

For this part of the assignment, you will write a DNS resolver that uses the DNSSEC protocol. To do this, you will have to send a query to the root with a special flag set that indicates that the DNSSEC protocol is being used. After each resolution, you will need to verify the integrity of the DNS response. DNSSEC uses fundamental principles from public-private key cryptography. While you won't need to know any encryption algorithms, you should have basic understanding of how public-private key cryptography is used in practice to understand DNSSEC.

The fundamental components of DNSSEC are as follows:

a. RRSET: the actual DNS query data, i.e. the fully qualified domain name and IP pair
b. DNSKEY: the public key of the DNS nameserver
c. RRSIG: the digital signature of the RRSET, as signed by the corresponding private key of the DNSKEY. Can be verified (decrypted) by the DNSKEY.
d. DS: Delegation Signer. DS helps establishes a *chain of trust* starting at the root.

You may use any library to send a single, iterative, DNS query. Note that not all domains will support DNSSEC. Your program should take a domain name as input and output according to three cases:

1- DNSSEC is configured and everything is verified (output the verified IP Address)
2- DNSSEC is not enabled (output "DNSSEC not supported")

3- DNSSEC is configured but the digital signature could NOT be verified. An example is http://www.dnssec-failed.org/, a site run by Comcast to specifically test whether or not your resolver has implemented DNSSEC correctly. They have a DNSKEY registered with .org that cannot resolve the digital signature at the www.dnssec-failed.org nameserver. In this case, output "DNSSec verification failed".

Please explain in detail how you implemented DNSSEC.

PART C (10 points)

Your last task is to measure the performance of your DNS resolver from Part A for resolving A records. Take the top 25 Websites from alexa.com (http://www.alexa.com/topsites.)

Experiment 1: Run your DNS resolver on each website 10 times, and find the average time to resolve the DNS for each of the 25 websites..

Experiment 2: Now use your local DNS resolver and repeat the experiment (state the name of the DNS resolver you used). Find the average time to resolve the address for the 25 websites, computed as the average over 10 runs.

Experiment 3: Change the DNS resolver to Google's public DNS (The IP address of this public DNS is often 8.8.8.8, or 8.8.4.4, but you need to verify). Repeat the experiment one more time.

Draw a graph that shows the Cumulative Distribution Function (CDF) of the DNS resolution time for Experiment 1, Experiment 2, and Experiment 3. You will have one graph to show all the CDFs. Your graph should have the correct x axis, y axis, units, labels, and legends. If you do not know how to draw a CDF, look it up. Explain your result as best as you can.

Submission instruction

You need to submit your homework in a single zip file as follows:

- The zip file and (the root folder inside) should be named using your last name, first name, and the homework name, all separated by a dash ('-')
  e.g. lastname-firstname-HW1.zip
- The zip file should contain your code corresponding to Part A and Part B. Please be sure to put both the code in the root folder rather than in separate folders. Provide sufficient comments to your code.
- Include the expected output file "mydig_output.txt" as specified in Part A.
- Include the text or PDF containing the details of your DNSSec implementation.
- Include the text or PDF file containing the answers for Part C (any format is ok).

- You should provide a README.txt file describing:
  - External libraries used.
  - Instructions on how to run your programs.

APPENDIX

A. You may write your programs in the following languages: python, Java, and C/C++. For each language, you can use the corresponding DNS library to perform the single DNS resolution. The recommended libraries are:
python —> dnspython
Java —> dnsjava (org.xbill.DNS)
C/C++ —> DSNQuery

If you choose to write your program using any other language or using a different DNS library, you should get permission from the instructor first. Remember that you cannot use these libraries to perform the entire resolution.

B. You can verify whether or not they are performing DNSSEC correctly with this tool by verisign: http://dnssec-debugger.verisignlabs.com/

A note on running your code using the Universities network

If you are not able to connect to the root servers on campus, this is because the campus network blocks access to the root servers (but the CS department network does not do so). You can work around this by using a VPN.