Assignment 4: DNS, Clouds, & CDNs

CS640 Spring 2015

Due: Tuesday, April 21 at 11pm

Overview

For this assignment, you will explore how DNS, clouds, and CDNs play a role in accessing websites. You'll then write your own simple DNS server that performs recursive DNS resolutions, and appends a special annotation if an IP address belongs to an Amazon EC2 region.

Learning Outcomes

After completing this assignment, students should be able to:

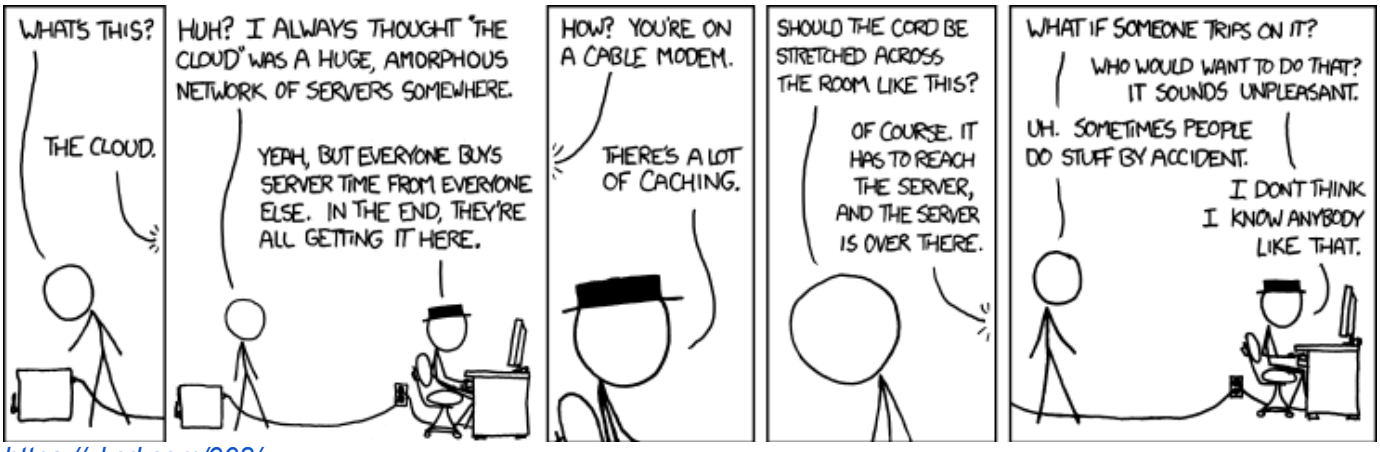
- Analyze how the use of domain names, cloud infrastructure, and content distribution networks (CDNs) affect web page load times
- Explain how the domain name system (DNS) works

Clarifications

- The names of the `setQuery()` and `isQuery()` methods in the DNS class are misleading due to an inversion of how they map to bits in the DNS header. You should patch your code as follows:

```
cd ~/assign4/
wget http://cs.wisc.edu/~agember/cs640/s15/files/assign4_query.patch
patch -p2 < assign4_query.patch
```

Part 1: Measurements



https://xkcd.com/908/

For this part of the assignment you will answer conduct some measurements to explore how DNS, clouds, and CDNs play a role in accessing websites. In particular, you'll explore these concepts in the context of the Science & Nature Pinterest page: https://www.pinterest.com/categories/science_nature/

We have already run an experiment using WebPagetest to measure various aspects of the page loading processes. The results of the experiment are available at: http://www.webpagetest.org/result/150409_PM_1NS/

You should use the WebPagetest results and the `dig` command to answer the following questions. You should submit your answers in a text file named `part1.txt`.

Questions

1. **DNS**

   a. According to the WebPagetest data, how many DNS lookups were performed by the client? How many of the lookups were redundant (i.e., the lookup was for the same domain name)?

   b. Assuming no DNS records were cached at the local DNS server, what DNS queries did the local DNS server need to issue to satisfy the client's first lookup?

   You can use the `dig` program (available on most Linux machines) to issue queries to DNS servers. When running `dig` for this question, you should use the following format:
   `dig +norecurse @`*`name.of.dns.server record-type domain-name`*
   `name.of.dns.server` is the domain name of the DNS server you wish to query
   `record-type` is the type of DNS record you wish to retrieve, e.g., A
   `domain-name` is the domain name you seek information on

   For each query issued by the local DNS server, you should specify what name server was queried (pick one if there are multiple options) and what records were received in response (ping one A record if there were multiple options). You should assume the local DNS server issues its first query to the root name server `a.root-servers.net`.

   Example: if the client's first lookup was for `cs.wisc.edu`, your answer would be

Name Server Queried	*Records Returned*
`a.root-servers.net`	`edu, NS, a.edu-servers.net` `a.edu-servers.net, A, 192.5.6.30`

`a.edu-servers.net`	`wisc.edu, NS, adns1.doit.wisc.edu`
	`adns1.doit.wisc.edu, A, 144.92.9.21`
	`adns1.doit.wisc.edu, AAAA,`
	`2607:f388::a53:1`

`adns1.doit.wisc.edu`	`cs.wisc.edu, NS, dns.cs.wisc.edu` `dns.cs.wisc.edu, A, 128.105.2.10`
`dns.cs.wisc.edu`	`cs.wisc.edu, A, 128.105.2.6`

   c. Assuming the local DNS server cached all records it retrieved while satisfying the client's first lookup, what DNS queries did the local DNS server need to issue to satisfy the client's second lookup? For each query specify what server was queried and what records were received.

2. **Clouds & CDNs**

   a. Which domain names (if any) resolved to a node in Amazon EC2? In which EC2 region was the node located? Use the list of Amazon EC2 public IP ranges by region to help you answer this question.

   b. Which domain names (if any) resolved to a node in the Akamai or EdgeCast CDN?

   c. According to the WebPagetest data, how many HTTP requests were issued to each CDN node? Over how many connections were these requests issued?

d. How many of the requested web objects were already stored in the CDN's cache? (Hint: look at the `X-Cache` field in the HTTP response header)

Part 2: Simple DNS Server

For this part of the assignment you will implement your own simple DNS server. Your server will accept queries from clients, and issue queries to other DNS servers in order to respond to client queries. Your server will also appends a special TXT record if an IP address belongs to an Amazon EC2 region. For simplicity, your server will not cache any DNS records, nor will it be responsible for storing the records for any DNS zones.

Learning About DNS Packets

Before you write any code, you should familiarize yourself with the format of DNS messages. You should read the Network Sourcery RFC Sourcebook page on DNS. You should also issue some DNS queries using `dig`, and look at the DNS packets using Wireshark.

You will need to have root (or administrator) access to capture packets, so you should issue your queries either from your own machine, or from your Mininet VM. You can use tcpdump in your Mininet VM to capture DNS packets:

```
sudo tcpdump -n -i eth0 udp port 53 -w dnstrace.pcap
```

You should then scp the file to a machine with Wireshark. Wireshark is installed on all CS machines by default.

```
scp dnstrace.pcap USERNAME@MACHINE.cs.wisc.edu:~
```

If you use your own machine, you can use Wireshark to both capture and view the packets.

In Wireshark, you should select the DNS packet you want to view, then look at its details in the pane in the bottom half of the Wireshark window. You should pay particular attention to the *Flags*, *Questions*, and *Answers* parts of the DNS packet.

Starter Code

We are providing code to parse and construct DNS packets, as well as a CSV file with the list of public IP address ranges for each EC2 region. You can download these files from: http://cs.wisc.edu/~agember/cs640/s15/files/assign4.tgz

Command Line Arguments

Your DNS server should be invoked as follows:

```
java edu.wisc.cs.sdn.simpledns.SimpleDNS -r <root server ip> -e <ec2 csv>
```

- `-r <root server ip>` specifies the IP address of a root DNS server
- `-e <ec2 csv>` specifies the path to a comma-separated variable (CSV) file that contains entries specifying the IP address ranges for each Amazon EC2 region

Receiving Queries

You should start by writing code that receives and parses DNS queries. Your server should listen for UDP packets on **port 8053**. You should call the `deserialize` method in the `DNS` class in the `edu.wisc.cs.sdn.simpledns.packet` package to parse the payload of a UDP packet that contains a DNS query.

Your server only needs to handle opcode 0 (standard query), and query types A, AAAA, CNAME, and NS. You can silently drop all other client queries. Also, your server only needs to handle one client query at a time (i.e., it does not need to be multi-threaded).

Handling Queries

When your server receives a query of type A, AAAA, CNAME, or NS, with the recursion desired bit set to 1, it should recursively resolve the query, starting from the root name server. If the recursion desired bit is set to 0, it should only query the root name server.

If a client issues a query of type A or AAAA for a domain name, and the domain name resolves to a CNAME, then you should recursively resolve the CNAME to obtain an A or AAAA record for the CNAME. Your reply to the client should include both the CNAME record for the original domain and the A or AAAA record for the CNAME.

If a query is of type A, and your DNS server successfully resolves the query, then you should check if the address(es) are associated with an EC2 region. For each address associated with an EC2 region, you should add a TXT record to the answers you provide to the client. The TXT record should contain the name of the EC2 region (from the CSV file), followed by a hypen (-), followed by the address in dotted decimal form. For example:

```
pinterest.com TXT USEast-50.16.219.149
```

Don't forget to include the A record(s) as well!

Testing

You can test your code using `dig`. To force queries to use your DNS server, include the arguments "`-p 8053 @localhost`". The question, answer, authority, and additional sections output by `dig` when using your DNS server should match what is output produced by dig when you query your machine's default DNS server (although the addresses and name servers may be slightly different if an upstream DNS server is using round robin to select which records are returned). You can also use `tcpdump` to help you debug.

Submission Instructions

You must submit a single tar file containing `part1.txt` which has your answers to questions in Part 1, the Java source files for your simple DNS server, and a Makefile that compiles your simple DNS server. Upload the tar file to the *Assignment 4* dropbox on Learn @ UW. Please submit only one tar file per group.