Lab: Wireshark DNS

See the Calendar for Lab Date and Due Date

August 7, 2018

14h 59min

1 Introduction and Goals

The goals of this assignment are:

- to gain familiarity with networking protocols and subsystems, in particular DNS, by viewing them in action via *Wireshark*.
- to answer some questions about networking operations.

1.1 CONCERNS

Because there can be security issues with a packet sniffer. **sniff only what you are asked to sniff**. If you would like to sniff other traffic, then **PRIOR** to sniffing, come talk to me, Tim Buchheim or Roger Wichmann.

1.2 TODO

Create a document **dns.pdf** to answer the following questions.

2 DNS Lab Activities

- If you can, clear your DNS cache.
- Open *Wireshark* and begin capturing packets.
- Open a browser window and contact some non-Claremont URL or use *nslookup* to inquire about some DNS name or use your DNS program from CS 105 to inquire about some DNS name. In all cases you want a host that is most likely not in your DNS cache
- Find the DNS Request Message

- 1
  Locate the DNS query and response messages. Are they sent over UDP or TCP?

- 2
  What is the destination port for the DNS query message: What is the source port of the DNS response message?

- 3
  To what IP address is the DNS query message sent? Use ipconfig to determine the IP address of your local DNS server. Are these two IP addresses the same?

- 4
  Examine the DNS query message. What ”Type” of DNS query is it? Does the query message contain any ”answers”?

- 5
  Examine the DNS response message. How many ”answers” are provided? What do each of these answers contain?

- 6
  So exactly what link are you watching with Wireshark, e.g., are you looking at a link that connects the CS switch

- 7
  How long did you send on this portion of the lab?

3 Comments on Caches

3.1 Linux, from tim

On Linux the cache is kept by the Name Service Caching Demon, **nscd**. Or on some systems **unscd** is used instead. **nscd**. and **unscd** are the only implementations Ive used, but there might be others. The GNU C library provides a standard interface for caching so anyone can write their own and drop it in.

Some people dont use the C librarys support for caching and instead run a full DNS server (either BIND or any of the other implementations) or a DNS proxy (e.g., dnsmasq). Many Linux systems dont have any cache set up at all. (I believe Ubuntu did not run any caching demon by default, and it was probably the most popular distribution.)

On Knuth we are running **unscd**. which provides a -i option to invalidate the cache. It takes the database to invalidate as an argument. So you would run **unscd -i hosts** to clear the DNS cache. I do not know of any way to view the cache with **unscd**. The man page doesnt mention anything and some quick poking around does not reveal anything.

3.2 OS X, from tim

I believe that programs which use POSIX mechanisms for DNS lookups (e.g., gethostbyname() and similar functions in the C library) do not have any DNS caching. Only programs that do lookups through Apples CFNetwork library and the libraries built on top of it (Foundation, AppKit, etc.) get any caching.

How to flush the cache depends on the version of OS X. **dscacheutil -flushcache** clears the cache on most recent versions. Older versions (10.4 or so) used **lookupd** rather than dscacheutil. Some people say to send a *SIGHUP to mDNSResponder* instead of or in addition to the flushcache command.

dscacheutil has a -cachedump command to print out the cache, but it does not work in recent versions of OS X. I tried it on both Mountain Lion (10.8) and Mavericks (10.9) but it just prints Unable to get details from the cache node. It works on Cortana, which is still running Snow Leopard (10.6). So on Cortana the command **dscacheutil -cachedump -entries host** produces output, but the list is really short and uninteresting as Cortana does not do a lot of DNS lookups (aside from POSIX-level stuff like sshd and slapd, which doesnt hit the cache) because nobody is logged in and running GUI stuff.