University of Glasgow | School of Computing Science

Assessed Coursework

Course Name	Networked Systems (H)			
Coursework Number	Summative exercise 1			
Deadline	Time:	4:30pm	Date:	1 March 2018
% Contribution to final course mark	20%			
Solo or Group ✓	Solo	✓	Group	
Anticipated Hours	20			
Submission Instructions	Submit via Moodle, following instructions in the lab 3 hand-outs.			
Please Note: This Coursework cannot be Re-Assessed				

Networked Systems (H) Laboratory Exercise 3: Understanding the Topology of the Internet

Dr Colin Perkins
School of Computing Science
University of Glasgow
`https://csperkins.org/teaching/2017-2018/networked-systems/`

8 February 2018

Introduction

The Internet is a network of networks. In this laboratory exercise you will explore the topology of the Internet, studying the router-level structure of the network to gain some understanding of the scale and connectedness of the network, how the different networks that form the Internet interconnect.

This is a summative exercise. It is assessed, and worth 20% of the marks for this course. This exercise comprises three parts that build on each other. You should read all three parts, followed by the submission instructions and marking scheme, before starting to implement your solution.

Part 1: IP Addresses of Popular Websites

The Internet separates naming from addressing. The network operates using IP addresses, that denote locations in the network, while users prefer to use readable domain names. The domain name system (DNS) is an application that runs of the network, and translates names into addresses.

Popular sites are implemented using multiple servers, hosted at different locations in the network. This increases robustness of the service to failures, and also allows for load balancing across different data centres and network connections. To support this, domain names often correspond to more than one IP address. These addresses can represent servers in different locations in the network, or perhaps connected using different protocols, such as IPv4 and IPv6.

Write a program, `dnslookup`, that takes a list of one or more domain names on the command line, and performs a DNS lookup for each name (hint: use `getaddrinfo()` as described in the laboratory exercise 1 client, but replacing the `socket()` and `connect()` calls with a call to `inet_ntop()` to print the address). After each DNS lookup, your program must print out the list of IP addresses returned, prefixed with the name and address type. For example, your program might print the following, if asked for the addresses of `www.google.com`:

```
$ ./dnslookup www.google.com
www.google.com IPv4 172.217.23.36
www.google.com IPv6 2a00:1450:4009:801::2004
$
```

The number and type of IP addresses returned for a given name might vary, depending on the date and location from which you perform the lookup. The format of the output from your program *must* match this example, with the host name, address type, and IP address on each line, separated by a single space character, with each line ending with a single carriage return ("`\n`").

Your program must be written as a single file, `dnslookup.c`. This must compile without errors or warnings, using the command “`clang -W -Wall dnslookup.c -o dnslookup`” run on the Linux machines in the lab (note that the letter `W` is capitalised in the two places it appears this command).

Use your `dnslookup` tool to find the IP addresses for a range of websites. These should be a mix of academic, commercial, social network, news, and personal websites, from a wide range of different countries. You should find the addresses of around 15-20 different websites.

Part 2: The Router-level Network Topology

The `traceroute` tool can discover the network path used to reach a particular destination. For example, a traceroute to the IPv6 address of `www.google.com` previously found might show:

```
$ traceroute -6 -q 1 -n 2a00:1450:4009:801::2004
traceroute to 2a00:1450:4009:801::2004 (2a00:1450:4009:801::2004), 30 hops max, 80 byte packets
 1  2001:630:40:f00:e22f:6dff:fe2c:ed80  0.276 ms
 2  2001:630:40:20::11  0.298 ms
 3  2001:630:40:20::72  0.723 ms
 4  2001:630:0:900c::1  0.587 ms
 5  2001:630:0:10::185  1.016 ms
 6  2001:630:0:10::1fd  5.337 ms
 7  2001:630:0:10::1f1  7.400 ms
 8  2001:630:0:10::1dd  11.276 ms
 9  2001:630:0:10::1c9  11.742 ms
10  *
11  2001:4860:0:1::ca1  12.514 ms
12  2001:4860:0:1::2b3  12.918 ms
13  2a00:1450:4009:801::2004  12.234 ms
```

The `-6` option means trace using IPv6 (use `-4` to trace IPv4). The `-q 1` option means only probe each hop once, the `-n` option means don’t try to look-up the domain names for the routers. Each line gives the distance in hops, the IP address of the router, and the time taken for the response. The first router (2001:630:40:f00:e22f:6dff:fe2c:ed80 in this example) is the device directly connected to the sender. That connects, in turn, to a router with IP address 2001:630:40:20::11, which connects to 2001:630:40:20::72, and so on. A * response (as in hop 10 of the example) indicates that no response was received. A long sequence of * responses, sometimes seen at the end of a trace, indicates a firewall that is blocking traceroute requests.

Run `traceroute` to each of the IP addresses you discovered using your `dnslookup` tool in Part 1 of this exercise, redirecting the output into files, with separate files for IPv4 and IPv6 traceroute output. You might find it helpful to write a script to do this. This will give you a record of the IP addresses of the routers data traverses to reach each of the chosen sites (or, if there’s a firewall, until it reaches the firewall).

Write a program, in the language of your choice, that reads the saved traceroute files, strips out any hops that didn’t respond to traceroute (lines with * responses), and reformats them as a list of pairs of adjacent router addresses, in quotes, with each pair of addresses being separated by `--`. For example, if run on the traceroute output shown above, this program would give the following output:

```
"2001:630:40:f00:e22f:6dff:fe2c:ed80"  -- "2001:630:40:20::11"
"2001:630:40:20::11"  -- "2001:630:40:20::72"
"2001:630:40:20::72"  -- "2001:630:0:900c::1"
"2001:630:0:900c::1"  -- "2001:630:0:10::185"
"2001:630:0:10::185"  -- "2001:630:0:10::1fd"
"2001:630:0:10::1fd"  -- "2001:630:0:10::1f1"
"2001:630:0:10::1f1"  -- "2001:630:0:10::1dd"
"2001:630:0:10::1dd"  -- "2001:630:0:10::1c9"
"2001:630:0:10::1c9"  -- "2001:4860:0:1::ca1"
```

```
"2001:4860:0:1::ca1"  -- "2001:4860:0:1::2b3"
"2001:4860:0:1::2b3"  -- 2a00:1450:4009:801::2004"
```

Run your program on each of the saved traceroute files in turn, saving the results from each in a separate file (a "processed traceroute file"). Then, concatenate the contents of all your processed IPv4 traceroute files into a single file called `router-topology-v4.dot`, sorting them into order, removing duplicate lines (hint: `cat ... | sort | uniq > ...` in the command shell), and adding a line:

```
graph routertopology {
```

to the beginning, and a line containing just:

```
}
```

at the end. The resulting `router-topology-v4.dot` file is a description of the router-level network topology, and should be a valid file in the *dot* graph description language.[1] Do the same with your saved IPv6 traceroute files, to produce `router-topology-v6.dot`.

Use the `dot` program (or one of its variants such as `neato`, `twopi`, or `circo`), installed on the Linux machines in the lab and part of the Graphviz suite of tools[2] (or some other graph plotting program of your choice) to generate a router-level map of the router level network topology, in files called `router-topology-v4.pdf` and `router-topology-v6.pdf` for the IPv4 and IPv6 traceroutes. Figure 1 shows an example an IPv4 router level topology map, of the sort you are to produce.

Part 3: Understanding Network Topology and the `traceroute` Tool

Review the IP addresses found for each site in Part 1 of this exercise, and the router level network topology maps you prepared in Part 2, and write up a short report, not to exceed two sides of A4 paper when printed using a 10pt font, that discusses the following points:

- **IP addresses:** Looking at the addresses found for the different sites in Part 1, do some sites have more than one IP address, and what does it mean if they do? If you run your `dnslookup` program several times, do you always get the same IP addresses for a site, and if not, why might this be? Do you get the same IP addresses for a site if you run your `dnslookup` program from different locations, and if not, why? What proportion of sites have an IPv6 address?

- **Router-level Topology Maps:** Looking at the router level topology maps you produced in Part 2 of this exercise, what is the longest path you can find in the network? Are paths from different locations to the same destination disjoint? Are there multiple routes to some destinations? Can you infer anything about organisational boundaries (e.g., which parts of the network are operated by different ISPs) from changes in the IP address prefixes?

- **IPv4 and IPv6:** Do the IPv4 and IPv6 router level network topology maps match? The addresses will be different, of course, but do the topology maps have the similar structure? Discuss whether you expect that they should have similar structure.

- **The `traceroute` Tool:** How does the `traceroute` tool work? Do some background reading, and *in your own words* explain how `traceroute` can find the IP addresses of the routers on the path to a destination. You should discuss how the TTL is varied during a traceroute, what is ICMP, and how traceroute uses ICMP.

Prepare this report using the word processing package of your choice, formatted for A4 size paper, and using the Times font in 10pt. Save a copy of your report in PDF format, under the filename `report.pdf`.

[1] https://en.wikipedia.org/wiki/DOT_(graph_description_language)
[2] http://www.graphviz.org/

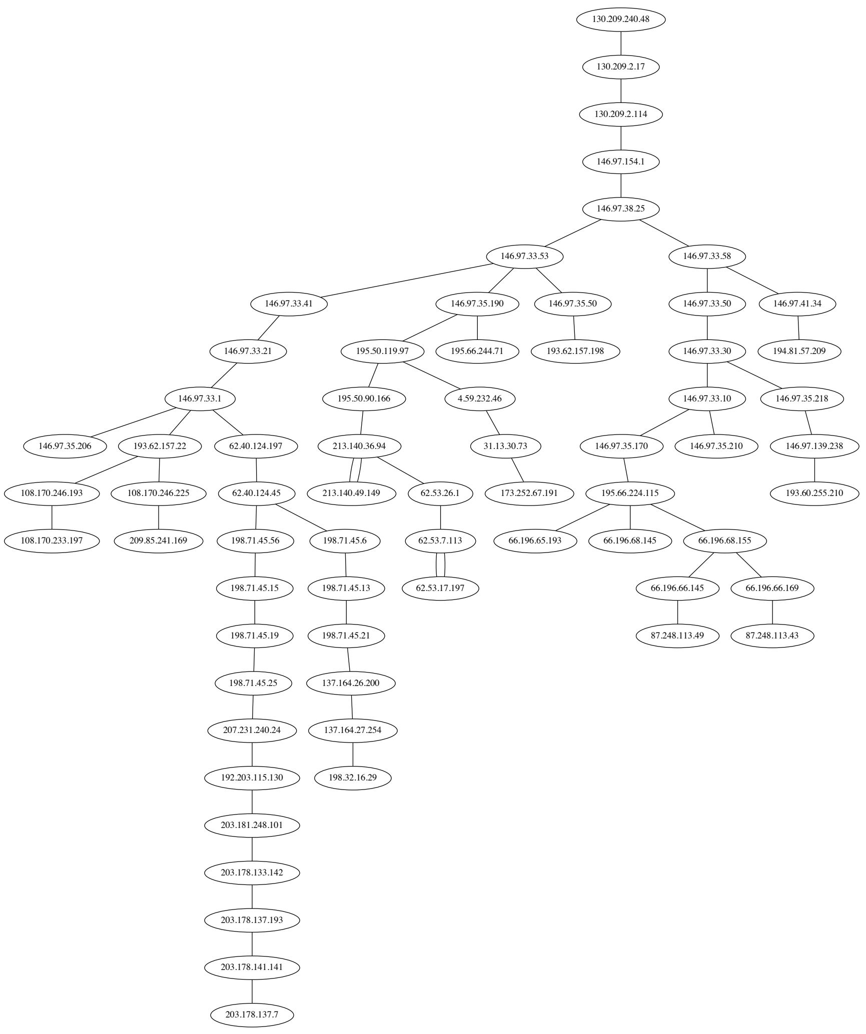

Figure 1: Sample router topology visualisation from a single site

Submission and Assessment

Create a directory named `lab03-`*`matric`*, where *`matric`* is replaced by your seven digit numeric matriculation number (note: 7-digits only, *do not* include the first letter of your surname). Put copies of the following four files into this directory: the source code for your `dnslookup.c` program; the two PDF files, `router-topology-v4.pdf` and `router-topology-v6.pdf` that contain your network topology maps; and the PDF file of your report, `report.pdf`. Create a zip archive of the directory, as a file called `lab03-`*`matric`*`.zip`. Submit the zip archive via Moodle.

For example, if your matriculation number was 1234567, you would perform the following steps to create your zip archive:

```
mkdir lab03-1234567
cp dnslookup.c lab03-1234567/
cp router-topology-v4.pdf lab03-1234567/
cp router-topology-v6.pdf lab03-1234567/
cp report.pdf lab03-1234567/
zip -r lab03-1234567.zip lab03-1234567/
```

If you have prepared your archive correctly, then running "`unzip -lqq lab03-`*`matric`*`.zip`" should produce output like the following:

```
$ unzip -lqq lab03-1234567.zip
        0  02-05-2017 18:13   lab03-1234567/
     1359  02-05-2017 18:13   lab03-1234567/dnslookup.c
    14315  02-05-2017 18:13   lab03-1234567/router-topology-v4.pdf
    15354  02-05-2017 18:13   lab03-1234567/router-topology-v6.pdf
   172851  02-05-2017 18:13   lab03-1234567/report.pdf
$
```

Check carefully that your zip archive extracts into the appropriate subdirectory, and contains only the requested files.

Assessment and Marking Scheme

This is an assessed exercise, worth 20% of the marks for this course. The deadline for submissions is 4:30pm on Thursday 1 March 2018. The Code of Assessment allows late submission up to 5 working days beyond this deadline, subject to a penalty of 2 bands for each working day, or part thereof, the submission is late. Submissions received more than 5 working days after the due date will receive an H (band value of 0). *This penalty will be strictly enforced.*

Submissions that are not made via Moodle, that have the wrong filename, that have a zip archive that extracts into the wrong directory or that contains the wrong files, or that otherwise do not follow the submission instructions will be subject to a 2 band penalty. This penalty is in addition to any late submission penalty. *This penalty will be strictly enforced.*

Marks will be awarded as follows:

- Up to **[7 marks]** for `dnslookup.c`, comprising [1 mark] for compiling without errors or warnings, then if the code compiles and runs, up to a further [6 marks] for correctly looking up domain names and printing the results in the requested format.

- Up to **[4 marks]** for `router-topology-v4.pdf`, comprising [2 marks] for readable formatting and [2 marks] for picking a set of IP addresses that illustrates the range of the network topology.

- Up to **[4 marks]** for `router-topology-v6.pdf`, comprising [2 marks] for readable formatting and [2 marks] for picking a set of IP addresses that illustrates the range of the network topology.

- Up to **[15 marks]** for `report.pdf`, comprising [4 marks] for discussion of IP addresses, [4 marks] for discussion of router-level topology maps, [2 marks] for discussion of IPv4 and IPv6, and [5 marks] for explanation of the `traceroute` tool. Marks are assigned for reasoned, clearly explained, and technically correct discussion of the points noted in Part 3 of this exercise.

The result will be a numeric mark out of 30. This numeric mark will be converted to a percentage, then the percentage will be converted to a band on the 22-point University of Glasgow scale using the standard translation table for the School of Computing Science. Any applicable penalty for late submission and/or for not following submission instructions will then be applied, and a band will be returned. A brief written justification for the band will also be supplied.

- + -