Introduction to the Use of Computers

Networking Protocols

Friday 30th November 2012

This lab session is about some aspects of networking protocols, including the Domain Name Service for looking up hostnames and the HTTP and SMTP protocols for web browsing and electronic mail.

You will probably find these tasks easiest to achieve if you are using the Unix shell on `igor.gold.ac.uk`; utilities to do the same things do exist on Windows, but are not often installed; on the other hand, the Mac OS X Terminal (under `Applications→Utilities`) should have access to the relevant utilities.

1. This part involves using the `dig` command-line utility to perform DNS protocol requests and to display the responses. (The syntax for this command is a little bit complex; for today's purposes, we will be using just one forms of the command:

   - `dig [@`*`server`*`] -t` *type name* for lookups of records of type *type*, using the DNS server at `server` to resolve queries.

   (a) Start by looking up the A record for `igor.gold.ac.uk` itself, by using '`a`' for *type* and `igor.gold.ac.uk` for *name* in the command line above. The important information will be in the portion of the response headed `ANSWER SECTION`; note the IPv4 address returned.

   (b) Look up the NS record for *uk*, and use the follow the chain (as in the lecture) to look up the A records for `www.google.co.uk`. Do these A records agree with the ones provided by the default local nameserver?

2. This part involves using SMTP to send an e-mail; specifically, we will use the department's mail server infrastructure to send an e-mail to your college e-mail address, `maXXXyy@gold.ac.uk`.

   (a) The first step is to look up the servers responsible for handling mail for `gold.ac.uk`; use `dig` for records of type `mx` to do this.

   (b) The record that you obtained in the previous step will contain a server name for the highest priority entry; look up the IPv4 addresses for that server name, using `dig` to look up the `a` record.

   (c) We can now make a connection to one of the IPv4 addresses given. We will use the `netcat` utility, which can open a TCP connection to a specified host and port. It is run by typing

   ```
   netcat [host] [port]
   ```

   and hitting Return, replacing `[host]` with the IPv4 address of the host, and `[port]` with the port number. Use it to connect to port 25 (which is the port for SMTP mail servers) on the host whose IP address you looked up in the last step.

   (d) Because of the College's implementation of "Office365", this step will fail. Instead, we will have to use `igor.gold.ac.uk` for the subsequent steps; look up its IPv4 address now, and connect to it on TCP port 25 using `netcat`.

(e) The mail server should have printed a banner, and is now waiting for your input. The first step is to greet it, giving the client (your) principal domain name as a parameter. To do that, type `HELO gold.ac.uk` and hit Return.

(f) The other required components of an SMTP connection are the sender and receiver 'envelope addresses', which are communicated using `MAIL FROM:` and `RCPT TO:`. Using your College identifier for `maXXXyy`, set up the envelope addresses by typing

```
MAIL FROM:<maXXXyy@gold.ac.uk>
```

and hitting Return, then

```
RCPT TO:<maXXXyy@gold.ac.uk>
```

and Return once more.

(g) To type the body of your message, first type `DATA` and hit Return. Then (following the reply from the mail server) type a message to yourself, finishing as the server instructs.

(h) The message should now have been sent. Exit the SMTP session by typing `QUIT` and hitting Return. Check your College e-mail address for the mail.

(i) Your mail will probably look a little bit odd, as there may be no `From:` or `Subject:` header. These fields are transmitted in the `DATA` part of the exchange; repeat the SMTP transaction described above, but this time, when typing the message after sending `DATA` to the server, and before typing the 'body' of the message, type

```
From: Your Name <maXXXyy@gold.ac.uk>
Subject: IS50004A lab08 test
X-Extra-Information: Using netcat!
```

and leave a blank line before the body text.

3. THis part is about understanding what happens under the hood when requesting the URL `http://www.doc.gold.ac.uk/` using a Web Browser.

(a) First, look up the IPv4 address for the hostname `www.doc.gold.ac.uk`. Note that, unlike in the SMTP case, we look up the `a` record directly, rather than finding the `mx` record first and then the `a` record of the MX host.

(b) Next, connect to port 80 on that address using `netcat`. We will be using an extended version of the HTTP 1.0 protocol (rather than the 1.1 version currently used in browsers) for simplicity.

(c) To request the content for the page, type

```
GET / HTTP/1.0
Host: www.doc.gold.ac.uk
```

and hit Return to leave a blank line.

(d) The first line of the server's response to your request will contain a numerical response code. A code of '`200`' indicates that the server has completed the request. Codes in the three hundreds (such as '`301`' or '`302`') indicate that the content requested is available but at at different location, specified by a `Location:` line in the response. Find that line, and note the given URL.

(e) Repeat the request for the content, but using the new URL provided in the first response. The general form for an HTTP request for the URL `http://host.example/sample/path` is

```
GET /sample/path HTTP/1.0
Host: host.example
```

Replace the host and path with the host and path given in the response from the previous step.

(f) Continue following redirections until you retrieve content (the web server gives a status code of 200).

(g) Experiment with retrieving `http://www.doc.gold.ac.uk/does-not-exist` and `http://www.doc.gold.ac.uk/~maXXXyy/` (replacing `maXXXyy` with your college user identifier), noting in particular the different response codes (in the first line of the response).

Other resources:

- SMTP, RFC 2821. `http://tools.ietf.org/html/rfc2821`
- HTTP, RFC 2616. `http://tools.ietf.org/html/rfc2616`