8.4. Domain Name Services

8.4.1. Overview

Hosts on the Internet have names in addition to their IP addresses. While one IP address is assigned to each network interface that is configured at a host, the name of a host is not directly associated with a network interface. Typically, a host on the Internet has a single name and multiple aliases based on the services hosted on it. For example, a machine with the name `gold.cs.mcgill.ca` could also be known as `www.cs.mcgill.ca`. The later name indicates that the machine `gold.cs.mcgill.ca` hosts a web server.

Besides providing human-friendly identifiers, host names present another layer of indirection for locating hosts. For instance, `ftp.cs.mcgill.ca` is typically known as the host that provides the file transfer services for the `cs.mcgill.ca` domain. Using the name `mail.cs.mcgill.ca` to access the service provides an opportunity to change the actual machine that is hosting the service without end-user intervention. This property is widely used to deploy clusters of machines to host popular resource-consuming services.

Domain name service (DNS) is the widely used naming service on the Internet. In DNS, the names are organized into well structured hierarchies. The particular hierarchy that a name is associated with is often determined by the ownerships. For instance, `mimi.cs.mcgill.ca` is the *fully qualified domain name* (FQDN) of the host with name `mimi` in domain `cs.mcgill.ca`.

The simplest way to implement DNS is to place a file at each host that contains the hostname-to-IP bindings. When `mimi.cs.mcgill.ca` wants to access `apollo.cs.concordia.ca`, it looks up the database file for the IP address that corresponds to `apollo.cs.concordia.ca`. This approach is feasible for small networks that have static configurations and is used to setup name lookup service in Beowulf clusters.

In practical Internet settings, we need a distributed approach for name resolution. This is exactly what is provided by DNS.

8.4.2. Objective

This experiment is about setting up DNS over an example multi-LAN IP network using the GINI Toolkit and studying various aspects of it. In particular, the experiment focuses on studying the following concepts: (a) naming (differences between naming and addressing), (b) name resolution process, (c) reverse name resolution, and (d) DNS spoofing.

8.4.3. Background Material

As part of the preparation for this experiment, read the literature on the following topics: (a) DNS names and (b) operating systems APIs for name resolution. The depth of coverage provided by the textbook is not sufficient to carry out this experiment. You are advised to read the relevant RFCs that are available from the IETF website. Alternatively, you can read the somewhat old but useful paper on DNS by Mockapetris (P. Mockapetris and K. Dunlap, "Development of the Domain Name System," *ACM SIGCOMM Computer Communications Review*, Vol. 18, No. 4, Aug. 1988, pp. 123-133). For more information on configuring the DNS server (i.e., `named`) in Linux, consult the following HOWTO document: `http://www.tldp.org/HOWTO/DNS-HOWTO.html`. Another paper that is very interesting and useful to read is V. Ramasubramanian and E. Gun Sirer, "The design and implementation of a next generation name service for the Internet," *ACM*

SIGCOMM, 2004, pp. 331-342. Only sections 1 and 2 from the above paper are relevant to this experiment.

8.4.4. Description

Consider the IP network shown in Figure 8.3, where three routers are used to interconnect three LAN segments. You will use the GINI Toolkit to create this virtual network. Once configured, you will implement a DNS server that manages all the hostnames on the network. To further distribute the DNS, you will finally setup three DNS servers to host the three TLDs (`qc`, `mb`, and `pe`). One of these servers will act as the root DNS server as well as a TLD DNS server. This will form an accurate portrayal of how DNS servers work on the real Internet.

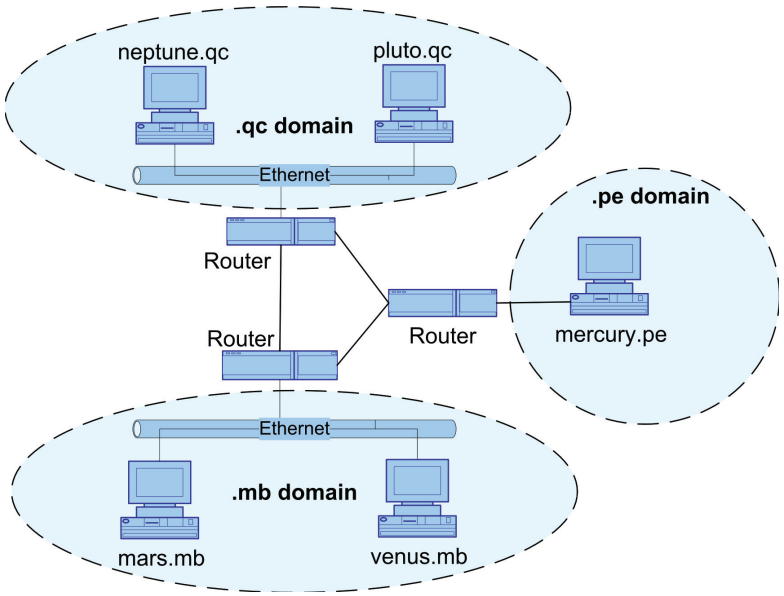

Figure 8.3: DNS Configuration on a multi-LAN IP network.

8.4.5. Useful Tools

- `/etc/hosts` — File determining static hostname-to-IP settings
- `/etc/resolv.conf` — File for configuring DNS connection settings
- `named` — DNS server
- `/var/log/messages` — Log file used by `named`, useful for debugging

8.4.6. Suggested Experimental Procedure

1. Use `gBuilder` to build a virtual network corresponding to the network given in Figure 8.3. The IP/MAC auto generation can be used to configure all the proper addresses and initialize all the routes. Test this by pinging individual machines by IP address and fixing any errors using the tools described in Chapter 6.

We will have three *top-level domains* (TLDs) to match the three networks we have emulated in GINI: `qc`, `mb`, and `pe`.

2. One way to map hostnames to IP addresses is by hardcoding all the hostnames in the `/etc/hosts` file found on each UML. This file is a static lookup table for hostname-to-IP address bindings. While a reasonable way to setup a small network, configuring the `/etc/hosts` file is a tedious process for even a moderately sized network. This is where DNS comes in. DNS allows the user to distribute the hostname-to-IP address bindings across the network. We will begin by creating a single root DNS server, using `mercury.pe` as this root server.

3. Configure the DNS on `mercury.pe` using `named` so that it maps requests for all the hostnames. For example, if `mars.mb` wishes to connect to `neptune.qc`, it will do a DNS lookup on `mercury.pe`, which will resolve `neptune.qc`'s hostname to its IP address. To do this, you need to edit the `/etc/named.conf` file on `mercury.pe` and add a *zone* option of type *master* pointing to `named.ca`:

```
zone "." IN {
    type master;
    file "named.ca";
};
```

Then, edit the DNS zone file given by `/var/named/named.ca`. This is the main file used to resolve DNS names. Replace `<FILL_IN_HERE>` with `MERCURY.PE.`, and append the following to the file:

```
.            NS   MERCURY.PE.

MERCURY.PE.  A    <MERCURY.PE's IP Address>
PLUTO.QC.    A    <PLUTO.QC's IP Address>
NEPTUNE.QC.  A    <NEPTUNE.QC's IP Address>
MARS.MB.     A    <MARS.MB's IP Address>
VENUS.MB.    A    <VENUS.MB's IP Address>
```

Examples of DNS zone files can be found on the Internet.

Type `named` at the command line to start the DNS service. Viewing `/var/log/messages` after launching `named` will aid in debugging your configuration. To stop `named`, type `killall named`.

You will need to edit the `/etc/resolv.conf` file on each machine to know the location of `mercury.pe` by adding the line `nameserver <mercury.pe's IP address>`. You can test this setup using tools such as `dig` or `host`. If all is well, a machine will resolve a hostname by accessing the nameserver running on `mercury.pe`, which contains the proper hostname-to-IP address bindings. Once the network is properly setup, applications such as `ssh` should be able to connect from one station to another by specifying the hostname instead of the IP address.

4. You will now add reverse DNS lookups to your DNS service. For each name in `/var/named/named.ca`, add an inverse. For example, if computer `pluto.qc` has IP address `192.168.12.10`, define the server to have an entry `10.12.168.192.in-addr.arpa` which maps to the name `pluto.qc`:

   ```
   10.12.168.192.in-addr.arpa  PTR PLUTO.QC.
   ```

   You can also configure your DNS server to have additional names that are aliases for existing names (existing hosts). Choose aliases such as `www.pluto.qc` and `www.mars.mb`. For example:

   ```
   WWW.PLUTO.QC.    CNAME    PLUTO.QC.
   ```

   This will link the name `www.pluto.qc` to the server `pluto.qc`.

5. In the above experiments, one hostname was mapped to a single IP address. In certain situations such as a web cluster, it is necessary to map many IP addresses to a single name. The DNS server will select one machine from the cluster using a load-balancing heuristic (e.g., round robin selection). Enter the following at the end of the `named.ca` file.

   ```
   CLUSTER.CA.      IN      A       <MERCURY.PE's IP Address>
                            A       <PLUTO.QC's IP Address>
                            A       <NEPTUNE.QC's IP Address>
                            A       <MARS.MB's IP Address>
                            A       <VENUS.MB's IP Address><
   ```

6. Launch `ping cluster.ca` from various machines and observe the IP addresses that are mapped to the host name.

7. Repeat the above configuration with multiple domain name servers. You should setup a nameserver for each domain: make `pluto.qc` the nameserver for the `.qc` domain and `mars.mb` the nameserver of the `.mb` domain. In the previous configuration, only one nameserver was present for the whole system and it was obviously authoritative for all the TLDs. In this part, however, there will be a separate authoritative name server for each TLD. `mercury.pe` will still act as the root name server, but all machines within the `qc` domain will first contact their local name server running on `pluto.qc`. If this name server cannot resolve the request, it will connect to the root name server.

   One way of doing this is to create two zones in the domain nameserver: one zone for the local domain and the other zone which resolves unknown requests to the authoritative nameserver. Here is an example setup of `/etc/named.conf` for `mars.mb`:

   ```
   zone "." IN {
           type hint;
           file "db.root";
   };
   zone "mb" IN {
           type master;
           file "named.ca";
   };
   ```

Create a file `/var/named/db.root` containing the following lines:

```
.                3600000          IN      NS      MERCURY.PE.
MERCURY.PE.      3600000                  A       <MERCURY.PE.'s IP ADDRESS>
```

This will send all unknown requests to `mercury.pe`.

Also, within `/var/named/named.ca` for `mars.mb`, we only want to resolve hostnames associated with the `.mb` domain:

```
MB.              NS      MARS.MB.

MARS.MB.         A       192.168.4.3
VENUS.MB.        A       192.168.4.2
```

Now, when a request comes for `pluto.qc`, the nameserver running on `mars.mb` will forward that request to `mercury.pe`. Make sure to also modify your `/etc/resolve.conf` files to point to the appropriate nameserver.

The authoritative nameserver must also be modified so that requests coming for either the `.qc` or `.mb` domain will be forwarded to the correct nameserver. An example setup for `mercury.pe` is:

```
                 NS      MERCURY.PE.
QC.              NS      PLUTO.QC.
MB.              NS      MARS.MB

MERCURY.PE.      A       192.168.6.6
PLUTO.QC.        A       192.168.5.4
MARS.MB.         A       192.168.4.3
```

This points to the appropriate nameservers for the given TLDs.

Test the configuration using `dig +trace`. Be sure the DNS caches are flushed before running the above command. You can flush the cache by restarting `named`. You should be able to see through `dig` how the hostname request is being resolved.

8.4.7. Review Questions

Answer the following review questions after completing this experiment. Background reading and the knowledge gained through the experiments should be sufficient to answer the questions.

1. DNS can map a single name to multiple IP addresses as shown in one configuration. What is the cost (cost is measured in required DNS server capacity) of using multiple names for a n IP address?

2. Several examples of DNS mapping multiple names to a single IP address were included in the above experiments. What is the cost of using multiple names? Is it more efficient or less efficient than using DNS to map a single name to multiple IP addresses?

3. Fault tolerance is an important consideration in DNS. How is it implemented? Before you answer this question, use `dig` to query for the name servers for large number of academic institutions in US and Canada. Do you see a pattern in how the name servers are configured for each domain?

4. Does this approach to provide fault tolerance (identified above) provide attack tolerance as well? What type of attacks can you tolerate? Do you see any cases where the fault tolerance technique can make DNS more vulnerable?

8.5. Simple Mail Transport Protocol

8.5.1. Overview

Electronic mail is one of the most popular applications on the Internet. The mail system has three major components: mail user agent and mail transfer agent. Over the years, user agents have evolved from terminal-based tools to GUI-based to browser-based. Some of the popular terminal-based user agents include pine, mail, and elm. There are many GUI based user agents including Eudora, Microsoft Outlook, and Mozilla Thunderbird.

The core of the email infrastructure is the *mail server*, where each mail user has a mailbox containing all the mail messages destined for that user. A typical message starts its journey in the sender's user agent. The user agent hands over the mail message to the sender's mail server which transmits it to the recepient's mail server. SMTP is the principal application-layer protocol that is used by the mail servers to handle mail message transactions.

SMTP uses the reliable data transfer service of TCP to transfer mail from the sender's mail server to the recipient's mail server. As with most application-layer protocols, SMTP has two sides: a client side, which executes on the sender's mail server, and a server side, which executes on the recipient's mail server.

8.5.2. Objective

Configure and test an e-mail delivery system on a simple multiple LAN IP network using the GINI toolkit. Get familiar with how e-mail is sent, delivered, and received on the Internet, and how SMTP works.

8.5.3. Background Material

The primary components involved in the delivery of e-mail are shown in Figure 8.4. We use a mail user agent to write the e-mail. The user agent hands over the e-mail to the mail transfer agent which uses the SMTP protocol to transfer the mail to the target mail transfer agent. When the mail is received by the target mail transfer agent, it hands over the mail to a mail delivery agent which sorts the mail according to the delivery address (in Figure 8.4, the mail transfer agent does the above functionality). The recipient uses her user agent to get the mail from the mail queue and read it.

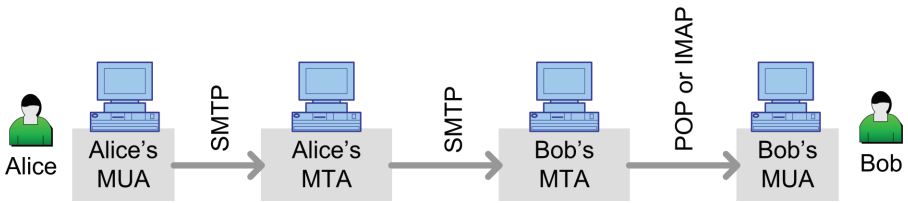

Figure 8.4: Simplified email delivery process.

8.5.4. Description

We will use the same setup as in the above DNS exercise. If you have completed that exercise successfully, you can use the same DNS setup.

Let pluto.qc and mars.mb be the mail servers for the qc and mb domains, respectively. Let Alice have a mail account alice at neptune.qc and Bob have a mail account bob at venus.mb. Setup the mail servers and DNS such that Alice and Bob can send mail messages to each other using their respective email addresses. For example, Alice should be able to send an email to Bob using the address bob@mail.mb and Bob should be able to reach Alice using the address alice@mail.qc.

8.5.5. Suggested Experimental Procedure

1. Before setting up the mail delivery system, configure the DNS for the example network, or alternatively, use the setup configured by the DNS experiment above.

2. Configure the postfix servers on pluto.qc and mars.mb. You will need to edit /etc/postfix/main.c and remove all the lines between main_owner and sendmail_path and add the following, where SERVER_NAME is either pluto.qc or mars.mb:

```
myhostname = mail.SERVER_NAME
mydomain = SERVER_NAME
myorigin = $mydomain
mydestination = $myhostname localhost.$mydomain $mydomain localhost

smtpd_banner = $myhostname ESMTP $mail_name
masquerade_domains = mail.SERVER_NAME SERVER_NAME

alias_maps = hash:/etc/postfix/aliases
alias_database = hash:/etc/postfix/aliases
```

3. Modify /etc/postfix/aliases and add the following:

```
root:           root        # This will allow root to retrieve messages
```

   Then run postalias /etc/postfix/aliases

4. Modify /var/named/named.ca to add an alias from mail.SERVER_NAME to SERVER_NAME and restart named.

5. Run postfix start to launch postfix.

6. After completing the above instructions for both mail servers, you can now test the setup by sending a message between the two root users running on the mail servers. Launch pine on one of the mail servers and go to Setup->Config. Set user-domain to SERVER_NAME, and smtp-server to mail.SERVER_NAME. Do this on both mail servers, and you should be able to send a message between the two root users.

8.5.6. Review Questions

Answer the following review questions after completing this experiment. Background reading and the knowledge gained through the experiments should be sufficient to answer the questions.

1. Give a brief security analysis of the mail setup described in this experiment. Can the current setup be easily exploited to send spam or other unsolicited e-mails?
2. In your opinion, is email spam a result of deficiencies in email delivery infrastructure, administration of a sound infrastructure, malicious users who cannot be controlled by any means, or a combination of the above. Provide valid reasons (from observations from doing this experiment) to support your position on the email spam problem.
3. Reputation management is used in e-commerce sites to rank sellers and/or commodities sold online. Buyers take the reputation measures into consideration when they make the buying the decisions. Explain how a reputation management scheme can be used in e-mail. What are the major hurdles in deploying a reputation management system to cut-down on spam? (gmail is already using some form of reputation management for spam control.)