DNS Infrastructure Lab

1 Overview

DNS (Domain Name System) is the Internet's phone book; it translates hostnames to IP addresses (and vice versa). This translation is through DNS resolution, which happens behind the scene. The resolution process involves many nameservers, including root servers, TLD servers, and final domain servers. These nameservers form the entire DNS system, which is an essential infrastructure for the Internet.

To help students understand how these nameservers work together to form the infrastructure, we will create a miniature DNS system inside an Internet Emulator. Even though this system is small, it has all the essential elements of a real DNS infrastructure. By building such a system, students will have a deeper understanding of how the DNS actually works. Although this lab is not a security lab, it is the basis for several SEED labs. This lab covers the following topics:

- The DNS infrastructure
- DNS forward and reverse lookup processes
- Root and TLD servers
- DNS resolver, local DNS server

Readings and videos. Detailed coverage of the DNS protocol can be found in the following:

- Chapter 23 of the SEED Book, *Computer & Internet Security: A Hands-on Approach*, 3rd Edition, by Wenliang Du. See details at `https://www.handsonsecurity.net`.
- Chapter 10 of the SEED Book, *Internet Security: A Hands-on Approach*, 3rd Edition, by Wenliang Du. See details at `https://www.handsonsecurity.net`.
- Section 7 of the SEED Lecture, *Internet Security: A Hands-on Approach*, by Wenliang Du. See details at `https://www.handsonsecurity.net/video.html`.

Lab environment. This lab has been tested on our pre-built Ubuntu 20.04 VM, which can be downloaded from the SEED website. Since we use containers to set up the lab environment, this lab does not depend much on the SEED VM. You can do this lab using other VMs, physical machines, or VMs on the cloud.

Overview of the lab tasks. The DNS infrastructure involves many servers. We will build a simplified DNS infrastructure inside the emulator. This infrastructure includes two root servers, two `com` TLD (Top-Level Domain) servers, one `edu` TLD server, and two second-level domain nameservers (See Figure 1). It should be noted that students need to replace `smith2021` with their own last names and the present year.

We need to configure the nameservers for each of the zone depicted in the figure. We will do it from the bottom up. Namely, we will configure the second-level domain nameservers first (Task 1), then the TLD server (Task 2), and then the Root server (Task 3). We also need to configure the local DNS server and the client machine (Tasks 4 and 5), so the client machine can use the DNS infrastructure. Task 6 is to complete the DNS infrastructure for the reverse lookup.

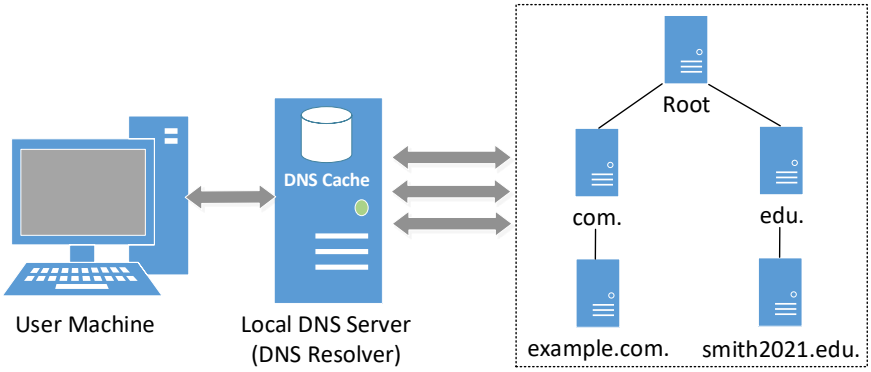

Figure 1: The simplified DNS Infrastructure

2 The Lab Setup and the SEED Internet Emulator

This lab will be performed inside the SEED Internet Emulator (simply called the emulator in this document). If this is the first time you use the emulator, it is important that you read this section. We recommend instructors to provide a lab session to help students get familiar with the emulator.

2.1 The Internet Emulator

We provide a pre-built emulator in two different forms: Python code and container files. The container files are generated from the Python code, but students need to install the SEED Emulator source code from the GitHub to run the Python code. The container files can be directly used without the emulator source code. Instructors who would like to customize the emulator can modify the Python code, generate their own container files, and then provide the files to students, replacing the ones included in the lab setup file. See the `README.md` file for instructions.

Download the emulator files. Please download the `Labsetup.zip` file from the web page, and unzip it. The files inside the container folder are the actual emulation files (container files); they are generated by the Python code. The name of the container folder is called `output/` for most labs, but if a lab has multiple emulators, it will use different folder names. The actual names will be given in the lab task.

Start the emulation. We will directly use the files in the container folder. Go to this folder, and run the docker commands to build and start the containers. We recommend that you run the emulator inside the provided SEED Ubuntu 20.04 VM, but doing it in a generic Ubuntu 20.04 operating system should not have any problem, as long as the docker software is installed. Readers can find the docker manual from this link. If this is the first time you set up a SEED lab environment using containers, it is very important that you read the user manual.

In the following, we list some of the commonly used commands related to Docker and Compose. Since we are going to use these commands very frequently, we have created aliases for them in the `.bashrc` file (in our provided SEEDUbuntu 20.04 VM).

```
$ docker-compose build  # Build the container images
$ docker-compose up     # Start the containers
$ docker-compose down   # Shut don the containers
```

```
// Aliases for the Compose commands above
$ dcbuild       # Alias for: docker-compose build
$ dcup          # Alias for: docker-compose up
$ dcdown        # Alias for: docker-compose down
```

All the containers will be running in the background. To run commands on a container, we often need to get a shell on that container. We first need to use the "`docker ps`" command to find out the ID of the container, and then use "`docker exec`" to start a shell on that container. We have created aliases for them in the `.bashrc` file.

```
$ dockps        // Alias for: docker ps --format "{{.ID}}  {{.Names}}"
$ docksh <id>   // Alias for: docker exec -it <id> /bin/bash

// The following example shows how to get a shell inside hostC
$ dockps
b1004832e275  hostA-10.9.0.5
0af4ea7a3e2e  hostB-10.9.0.6
9652715c8e0a  hostC-10.9.0.7

$ docksh 96
root@9652715c8e0a:/#

// Note: If a docker command requires a container ID, you do not need to
//       type the entire ID string. Typing the first few characters will
//       be sufficient, as long as they are unique among all the containers.
```

If you encounter problems when setting up the lab environment, please read the "Common Problems" section of the manual for potential solutions.

Set the terminal title. We may need to get into several containers using the terminal. We will likely create several terminal tabs, and switch back and forth among these tabs. We can easily get lost, because it is difficult to know which tab runs which container. To solve this problem, once we are inside a container, we can set the terminal title using one of the following commands (it sets the title to "`New Title`").

```
# set_title New Title
# st New Title       ← st is an alias of set_title
```

2.2 The Map of the Emulated Internet

Each computer (hosts or routers) running inside the emulator is a docker container. Users can access these computers using docker commands, such as getting a shell inside a container. The emulator also comes with a web application, which visualizes all the hosts, routers, and networks. After the emulator starts, the map can be accessed from this URL: `http://localhost:8080/map.html`. See Figure 2. To zoom in/out, use the mouse middle scroll. Click on any node, the detailed information of that node will be displayed on the side panel, from where, users can get a console on that node (container).

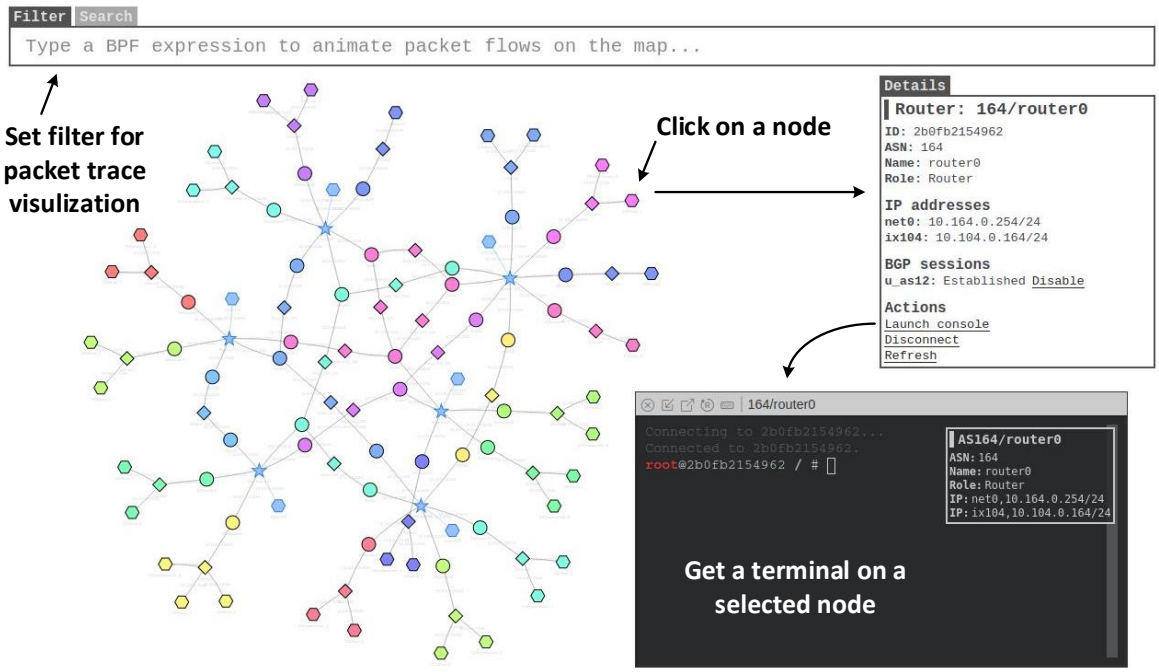

Figure 2: The map of the emulated Internet

2.3 Filtering and Replying

Users can also set filters to visualize network traffic. The syntax of the filter is the same as that in `tcpdump`; actually, the filter is directly fed into the `tcpdump` program running on all nodes. When a node sees the packets that satisfy the filter, it sends an event to the map, which will highlight the node briefly on the map.

Sometimes, a sequence of events happen too fast to see the actual order among them. In this case, we can use the Replay panel (see Figure 3) to record the events and then replay them at a slower pace. The speed of replaying can be adjusted by changing the event interval.

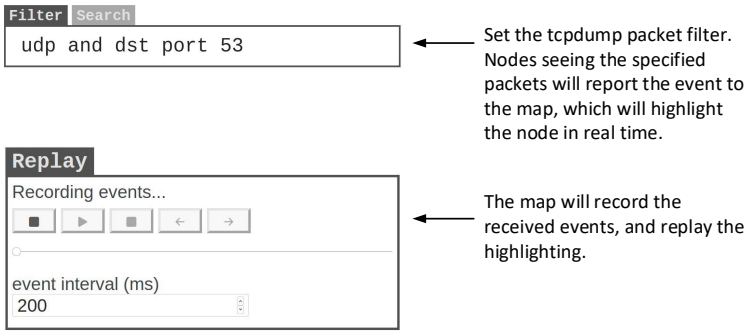

Figure 3: Capturing and replaying events

2.4 All the Nameservers

In this emulator, the DNS infrastructure is formed by a number of the containers running DNS nameservers. We have customized their container names, so they can be easily found. All of the containers have the term `DNS` in their names, and their IP addresses are also included in the names. The following command can list all of them (the order is re-arranged for the displaying purpose).

```
$ dockps | grep DNS
dedc6676421e  as150h-DNS-Root-A-10.150.0.72
de5dc343224f  as160h-DNS-Root-B-10.160.0.72
84cf7c7f337d  as151h-DNS-COM-A-10.151.0.72
20c134dceee4  as161h-DNS-COM-B-10.161.0.72
f20fe7ed115f  as152h-DNS-EDU-10.152.0.71
cfabce676bed  as154h-DNS-Example-10.154.0.71
5801404d45d2  as162h-DNS-AAAAA-10.162.0.72
c357ff76bda6  as153h-Global_DNS-1-10.153.0.53
d65e76c44437  as163h-Global_DNS-2-10.163.0.53

// Get a shell on a selected container
$ docksh cfab   ← container ID
root@cfabce676bed / #
root@cfabce676bed / # st Example    ← change the terminal title
```

2.5 Copy DNS Zone Files

In this lab, we need to modify the DNS zone files (also some of the configuration files) inside several containers. We can do that inside those containers. The problem is that once the containers are stopped and removed, those changes will be lost. It is better to copy those files to the hosting machine, make change on the host, and then copy them back to the container. We can use the "`docker cp`" command to do that. Since we need to run this command very often, we wrote the following shell scripts, one for getting a zone file from container to the host, and the other for copying the zone file back to the container. Both files are included in the lab setup folder.

Listing 1: Getting the zone file from container: `getzone.sh`

```
#!/bin/bash

keyword=$1   # Command-line argument 1: used to identify container
filename=$2  # Command-line argument 2: zone file name on container
filename_to=${keyword}_${filename}                                    ①
containerID=$(docker ps | grep $keyword | awk '{print $1}')          ②
if [[ ! -f $filename_to ]]; then
   docker cp $containerID:/etc/bind/zones/$filename ./$filename_to   ③
else
   echo "** File $filename_to already exists; will not overwrite."
fi
```

In the emulator, zone files are stored inside the `/etc/bind/zones` folder. Each DNS nameservers have a unique keyword in its container name, for example the nameserver for the `example.com` zone has the keyword `Example` in its name. We use this keyword to find the corresponding container's ID (see Line ②). We also prepend the keyword to the name of the zone file, so we can easily identify them once they are copied to the host machine (see Lines ① and ③). The following examples show how we get the zone file

from the two root nameservers and the `example.com` nameserver.

```
$ ./getzone.sh Root-A root           ←  New file: Root-A_root
$ ./getzone.sh Root-B root           ←  New file: Root-B_root
$ ./getzone.sh Example example.com.  ←  New file: Example_example.com
```

Copying files back to the container is quite similar, except that we need to restart the nameserver after the zone file is copied (see Line ④).

Listing 2: Sending the zone file to container: `sendzone.sh`

```
#!/bin/bash

keyword=$1   # Command-line argument 1: used to identify container
filename=$2  # Command-line argument 2: zone file name on container
filename_from=${keyword}_${filename}
containerID=$(docker ps | grep $keyword | awk '{print $1}')
if [[ -f $filename_from ]]; then
   echo "== Copy zone file to container"
   docker cp ./$filename_from $containerID:/etc/bind/zones/$filename

   echo "== Restart the nameserver"
   docker exec $containerID service named restart      ④
else
   echo "** File $filename_from does not exists."
fi
```

Here is an example of how to use `sendzone.sh`. It sends the Root-A's zone file (`Root-A_root`) back to Root-A's container, and restart the nameserver.

```
$ ./sendzone.sh Root-A root
== Copy zone file to container
== Restart the nameserver
 * Stopping domain name service... named
waiting for pid 92 to die
   ...done.
 * Starting domain name service... named
   ...done.
```

3 Task 1: Configure the Domain Nameserver

In this task, we will configure the nameservers for two domains, one is `example.com`, and the other is a customized name based on student's name.

3.1 Task 1.a: Configure the Nameserver for `example.com`

In this task, we will configure the nameserver for a domain called `example.com`. The nameserver is hosted on one of the containers inside the emulator. The keyword `Example` is included in the container name, as well as the IP address.

```
$ dockps | grep Example
46ab89e26738  as154h-DNS-Example-10.154.0.71
```

Step 1. Add the zone entry. To host a server for the `example.com` domain, we need to add a zone entry to the BIND's configuration, so it knows what zones it is going to host. In the emulator, the zones hosted by a nameserver are placed in the `/etc/bind/named.conf.zones` file (multiple zones can be placed in this file). Here is an example, where the `file` entry specifies the actual file containing the zone information. If we want to host another zone on this nameserver, we can add a corresponding zone entry in this file.

```
zone "example.com" {
    type master;         ← this is the master server
    allow-update { any; };
    file "/etc/bind/zones/example.com.";  ← the actual zone file
};

zone "example.net" {
    ...
}
```

The above entry for the `example.com` zone indicates that the current nameserver is the master server for this domain, and the zone file is specified in the `file` entry. In the emulator, the zone files are placed inside the `/etc/bind/zones` folder. It should be noted that most of the zone files (except the root zone) have the `.` at the end of the filename. This is the main file that we need to modify in this lab.

Step 2. Modify the zone file. Add at least four records (type `A`) to the zone file to map hostnames to IP addresses. You can use the asterisk (`*`) as a wildcard hostname.

```
$TTL 300               ← The default time to live:  300 seconds
$ORIGIN example.com.   ← The start of this zone file in the namespace
@ SOA ns1.example.com. admin.example.com. 1635647622 900 900 1800 60

@                  NS    ns1.example.com.
ns1.example.com.   A     10.154.0.71

www                A     10.154.0.72
abc                A     10.154.0.73
```

The zone file must specify the Start of Authority (SOA) record, which contains the name of the authoritative master name server for the zone, the email address of someone responsible for management of the name server, The parameters of the SOA record also specify a list of timing and expiration parameters (serial number, slave refresh period, slave retry time, slave expiration time, and the maximum time to cache the record). If we modify the zone file, we should change the serial number (the highlighted number in the SOA entry), so the change can be synchronized to the slave nameservers.

In the zone file, domain names that end with a full stop character (i.e., the dot), are fully qualified while those that do not end with a full stop are relative to the current origin. For example, in the above example, `ns1.example.com.` is a full name, while `www` example refers to `www.example.com`.

Step 3. Restarting the nameserver. Every time we modify the BIND configuration files, we should to restart the nameserver, so the changes can take effect. If there are mistakes in the configuration files, the nameserver will fail to start.

```
# service named restart
```

```
 * Stopping domain name service... named
                               waiting for pid 92 to die

                         [ OK ]
 * Starting domain name service... named
```

Step 4. Testing. Since we have not finished setting up the entire DNS infrastructure, we need to use the `@10.154.0.71` option to send our DNS query directly to `10.154.0.71`, which is the IP address of the `example.com` nameserver in our emulator. If everything is done correctly, you can get the IP address specified in your zone file. Please report your observations.

```
$ dig @10.154.0.71 www.example.com
...
;; ANSWER SECTION:
xyz.example.com.    300    IN    A    1.2.3.6
...
```

3.2 Task 1.b: Configure Nameserver for Another Domain

In this task, we will configure the nameserver for another domain. The name of the domain should use the following format <NAME><YEAR>.edu, where <NAME> should be replaced with your last name, and <YEAR> be replaced with the current year. For example, if your last name is `Smith`, and it is year 2021, the domain will be `smith2021.edu`. In the emulator, an empty nameserver has already been created; the keyword `AAAAA` is encoded in the container name. Please use this nameserver to host your domain.

```
$ dockps | grep AAAAA
70f2e8a57a96  as162h-DNS-AAAAA-10.162.0.72
```

4 Task 2: Configure the TLD servers

In this task, we will configure the nameservers for two TLD domains: `com` and `edu`. We have already created three nameservers for these two TLD domains.

```
$ dockps | grep DNS
c3040914667c  as151h-DNS-COM-A-10.151.0.72    ← The master com server
3fa8fa41afb6  as161h-DNS-COM-B-10.161.0.72    ← The slave com server
6c7132feb7d4  as152h-DNS-EDU-10.152.0.71      ← The edu server
...
```

There are two nameservers for the `com` zone. One is configured as the master server, and the other as the slave server. We only need to modify the zone file on the master (`COM-A`), as the slave server will automatically synchronize with the master server. See the following zone configuration on these two servers.

```
// For master (COM-A)
zone "com." { type master; allow-transfer { any; }; ...};

// For slave (COM-B)
zone "com." { type slave; masters { 10.151.0.72; }; ... };
```

All the nameservers within a TLD domain must register their nameservers with this TLD server; otherwise, nobody can find them. For each domain, such as `example.com`, we need to add two records in the `com` server's zone file: an `NS` record and an `A` record. The `NS` record specifies the nameserver for the `example.com` domain, while the `A` record specifies the IP address of the nameserver.

Lab Task. Please register your `example.com` and `<NAME><YEAR>.edu` nameservers with their corresponding TLD servers. Remember to run `"service named restart"` to restart the nameserver after the changes. Also remember to change the serial number of the zone file; otherwise, the changes on the master server will not be synchronized to the slave server. After making the changes, please run the following commands to test the servers. Report and explain your observations.

```
# dig @10.151.0.72 www.example.com       ← Query the COM-A server
# dig @10.161.0.72 www.example.com       ← Query the COM-B server
# dig @10.152.0.72 www.<NAME><YEAR>.edu  ← Query the EDU server
```

It should be noted, the TLD servers are configured not to conduct the recursive query, so they will only tell you the nameserver for the domain name in the query; it will not resolve the query for you. See the following configuration.

```
// File: named.conf.options
options {
        directory "/var/cache/bind";
        recursion no;
        dnssec-validation no;
        empty-zones-enable no;
        allow-query { any; };
        allow-update { any; };
};
```

5 Task 3: Configure the Root servers

In this task, we will configure the nameservers for the root zone. In the real world, there are 13 nameservers for the root zone, and they are synchronized through the root zone file maintained by IANA. In our emulator, we have only created two root servers, and their container names contain the keyword `Root`. We will manually synchronize them by putting the identical content in their zone files.

```
$ dockps | grep Root
9a96092c7c85  as150h-DNS-Root-A-10.150.0.72
60f171f92287  as160h-DNS-Root-B-10.160.0.72
```

All TLD nameservers need to register with the root nameserver, so they can be found in the DNS query process. For every TLD zone that we would like to include in our miniature DNS system, we need to add at least two records in the zone file, including an `NS` record and an `A` record. In this task, students need to modify both root server's zone files to support the `com` and `edu` TLDs inside the emulator. The zone file is located inside the `/etc/bind/zones` folder. After making the changes, restart both nameservers, and then run the following commands from another container (replace the `<root-ip>` with the root server's IP address; please try both root servers). Report and explain your observations.

```
$ dig @<root-ip>    <ANYNAME>.com   ← choose an arbitrary name
$ dig @<root-ip>    <ANYNAME>.edu
```

```
$ dig @<root-ip>    com
$ dig @<root-ip>    edu
```

6 Task 4: Configure the Local DNS Server

When a computer needs to resolve the IP address from a hostname (or vice versa), it sends a request to its helper, which is called local DNS server, also called DNS resolver. This DNS resolver will conduct the entire DNS resolution process, and then send the result back to the computer. Although it is called "local", this server does not need to be local. In the past, when a computer is set up, it usually uses the DNS servers on its own local network, but nowadays, there are many non-local DNS servers that can be used as "local" DNS servers. For example, the Google Public DNS is a DNS service offered by Google, serving any host on the Internet.

When we configure the root, TLD, and domain nameservers, we configure them to be non-recursive, i.e., they will only tell you what they know, and they will not conduct the entire resolution process to get the final answer for you. When we configure the local DNS server, we turn on the recursive option (see the following), so it will get the answer for you.

```
options {
    recursion yes;
    ...
};
```

If the local DNS server cannot find the answer from its cache, it will go through an iterative process to get the answer from outside nameservers. The process starts from the root servers. Therefore, the local DNS server needs to know the IP addresses of the root servers.

In `/etc/bind/named.conf.default-zones`, which is included in BIRD's configuration file `/etc/bind/named.conf`, there is an entry for the root zone. This entry specifies a hint file for the root zone, and that is how the local DNS server knows the IP addresses of the root servers.

```
zone "." {
   type hint;
   file "/usr/share/dns/root.hints";
};
```

In the real world, there are 13 nameservers for the root zone, so their IP addresses are provided in the `root.hints` file. In our emulator, we have only two root servers. Their information is already added to the hints file.

```
.       NS    ns1.
.       NS    ns2.
ns1.    A     10.150.0.72
ns2.    A     10.160.0.72
```

The task. We have created two DNS resolvers in the emulator. We put `Global_DNS` in their container names. Their IP addresses are also encoded in the names. You can use the following command to find out these two DNS resolvers.

```
$ dockps | grep Global_DNS
c357ff76bda6  as153h-Global_DNS-1-10.153.0.53
```

```
d65e76c44437  as163h-Global_DNS-2-10.163.0.53
```

Please go to their `root.hints` file. Inside the file, please change the IP addresses of the root nameservers to something else (random, e.g., changing `72` to `75`), and then show how that affects the DNS resolution process (run the following commands). Please report and explain your observations. Please do remember to change them back after this task, because the subsequent tasks depend on them.

```
# dig @10.153.0.53  example.com
# dig @10.163.0.53  example.com
```

7 Task 5. Configure the Client

So far, we need to use `@<ip>` in our `dig` command to indicate what DNS server the `dig` command should talk to. While this is not an issue for `dig`, it is a problem for other software that depends on DNS. We need to tell the operating system what local DNS server that it should use. This is achieved by changing the resolver configuration file (`/etc/resolv.conf`) of the user machine, so the container's IP address is added as the first `nameserver` entry in the file, i.e., this server will be used as the primary DNS resolver.

We have already configured all the machines to use `10.153.0.53` as the primary DNS resolver, except for the machines in AS-155. Your job is to configure the host machines in AS-155, so they can use one of the DNS resolvers (or both of them). You can add the following entries (one or both) to the `/etc/resolv.conf` file.

```
nameserver 10.153.0.53
nameserver 10.163.0.53
```

Testing the entire DNS Infrastructure. At this point, the DNS infrastructure is completely set up, we can run the `dig` without using the `@<ip>` option; `dig` will use the local DNS server set by the system. Please run the following commands from a host inside AS-155. If everything is set up correctly, you should be able to get the answer as expected.

```
# dig www.example.com
# dig <NAME><YEAR>.edu
```

Please use the map to show the packet trace. First set the filter to `"udp and dst port 53"` to capture the DNS request traffic. Then run the above `dig` commands. If the entire process happens too fast, you can use the Replay feature of the map: After setting the filter, press the Record button, and then run the above `dig` commands. Stop the recording, and replay the recorded events. Adjust the time interval if needed. It is hard to take screenshots of the packet trace, describing the packet trace in your report is sufficient.

8 Task 6: Reverse DNS Lookup

Reverse DNS lookup is the process of finding out the hostname from an IP address. This process is actually quite similar to the forward lookup. We use an example to illustrate the process. Given an IP address, such as `128.230.171.184` (an IP address belonging to `syr.edu`), the DNS resolver constructs a "fake" name `184.171.230.128.in-addr.arpa`, and then send queries through an iterative process, just like the forward lookup. Namely, it starts from the ROOT server, to the `in-addr.arpa` server,

128.in-addr.arpa server, and eventually reach the 230.128.in-addr.arpa server, which is the same nameserver as that hosting the syr.edu zone.

Tasks 1 - 4 only set up the DNS infrastructure for the forward lookup. In this task, we will set up the infrastructure for the reverse lookup. For the sake of simplicity, We will only support the reverse lookup for the IP addresses belonging to the example.com domain in the emulator (i.e., the IP addresses in the 10.154.0.0/24 network). You need to set up the following nameservers:

- Step 1: Configure the root server. The root servers need to host the NS records for the in-addr.arpa zone. We need to add the records to the /etc/bind/zones/root file on both root servers.
- Step 2: Configure the in-addr.arpa server. This server should host the in-addr.arpa. zone. We can use the com nameserver to host this zone by adding the zone to the /etc/bind/named.conf.zones. A nameserver can simultaneous host multiple zones.

```
zone "com." {
       ... already there ...
};

zone "in-addr.arpa." {
       type master;
       notify yes;
       allow-transfer { any; };
       allow-update { any; };
       file "/etc/bind/zones/in-addr.arpa";  ← the zone file
};
```

We need to create the zone file in-addr.arpa. specified in the entry above. We add the NS record for the 154.10.in-addr.arpa zone in the zone file. For the sake of simplicity, we directly provide the NS for the 154.10 sub-zone, instead of only providing the NS record for the 10.in-addr.arpa zone (and then reply on the 10.in-addr.arpa server to provide the information for the 154.10 sub-zone. A skeleton zone file is provided in the following:

```
$TTL 300
$ORIGIN in-addr.arpa.

@        SOA ns1.com. admin.com. 1565237345 900 900 1800 60
@        NS  ns1.com.
@        NS  ns2.com.
ns1.com. A 10.151.0.72
ns2.com. A 10.161.0.72

; Please replace ____ with the correct information
154.10.in-addr.arpa.  NS  ____   ← The nameserver of the domain
____                  A   ____   ← The IP address of the nameserver
```

- Step 3: Configure the 154.10.in-addr.arpa server: this nameserver can be hosted on the same nameserver where the example.com zone is hosted. We should add a zone entry in its /etc/bind/named.conf.zones file (similar to Step 2), and then create the corresponding zone file, which should contain the reverse lookup records for the example.com domain. We give an example of the zone file in the following:

```
$TTL 300
```

```
$ORIGIN 154.10.in-addr.arpa.
@ SOA ns1.example.com. admin.example.com. 1635647622 900 900 1800 60

@               NS ns1.example.com.
ns1.example.com. A  10.154.0.71

71.0     IN      PTR     ns1.example.com.
72.0     IN      PTR     www.example.com.
73.0     IN      PTR     abc.example.com.
```

Testing. After making the changes to a nameserver, make sure run the "`service named restart`" to restart the nameserver. If everything is set up correctly, we can go to any of the hosts in the emulator, run the following reverse query commands. Please report and explain your observations. Please use the map to show the packet trace for one of the commands, and describe the packet trace in your report.

```
# dig -x 10.154.0.71
# dig -x 10.154.0.72
# dig -x 10.154.0.73
```

9 Submission

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits.

Acknowledgment

This lab was developed with the help of Honghao Zeng and Keyi Li, who contributed to the development of the DNS module in the SEED Internet Emulator. The SEED project was funded in part by the grants from the US National Science Foundation and Syracuse University.