DHCP Server
The goal of this exercise is to learn about using and configuring a DHCP server. Instead of assigning a static IP address to the ethernet interface of your Pi (as you did in the previous exercise), the address should be assigned by your DHCP server that is running on a node attached to the ethernet subnet. Install a DHCP server (the dhcpd program) on your Pi and test its basic functionality by attaching two nodes to the same subnet. The DHCP server should also provide the information for the DNS server and default gateway of the network to the client nodes. Coordinate with you classmate so that a few of you attach your Pis to the same subnet (the ethernet switch that we have in Boyana’s office now), and have the gateway routing feature (from the previous exercise) enabled on at least on of Pis, say x, then x can be used as the default gateway. Use tcpdump to capture the exchanged messages between your client and the DHCP servers, and check the content of those messages. In your report describe the type of exchanged messages between a client and a DHCP server, the content of these messages, and specify the main pieces of information in each message (e.g. type of ICMP message, MAC addresses of the client and server, provided IP address by the server, …)

Exercise 1:
In practice, there could be settings where multiple DHCP servers observe the same group of clients and can provide them with IP address (and other information). For example, if you all connect your Pis to the same ethernet segment and run a DHCP server on multiple Pis, there will be a conflict among theses servers for assigning IP address to a client on that subnet. The design of the DHCP protocol accommodates such setting and ensures that only one of the DHCP servers assigns an IP address to a client. Using tcpdump capture the exchanged messages between a client and 2 (or more) DHCP servers that run on the same subnet to explain how one (of multiple) DHCP server(s) is selected to serve the client.

Another problem in a setting with multiple visible DHCP servers to a client is to ensure that the assigned IP addresses by each DHCP server is unique. Consider a subnet with the assigned IP prefix of 192.168.15.0/24. If all the DHCP servers on the same subnet manage the same prefix, while at each point only one of them provides IP address to a client, it is still possible that a single IP address is handed out by two DHCP servers to different clients if the book keeping of the assigned addresses by different servers become inconsistent (this problem may not occur in certain OSes such as Linux where the client uses ARP ping before accepting a proposed IP address). Given a specific set of DHCP servers, you should devise a solution for this problem and show that it works in practice with multiple DHCP servers on the same subnet while fully utilizing all the addresses in the provided prefix. Hint: You should make adjustments to the configuration file for DHCP servers (at /etc/dhcpd.conf) to achieve this goal.
Exercise 2:
The goal of this exercise is to create a misbehaving node that obtains all IP addresses that are managed by a single DHCP server. Using the `ip` command from the previous exercise, you can assign a virtual MAC address to your node. Since DHCP servers identify and assign an IP address to each node on a subnet based on its MAC address, a node with multiple MAC address can request and obtain a separate IP address for each MAC address. Using this idea, develop a small script that consumes all the IP addresses of a single DHCP server.

DNS Server
DNS servers perform the important tasks of translating a domain name (**www.cnn.com**) into an IP address among other things. In this section, you conduct a few exercises with the DNS servers. First, you need to become familiar with the BIND program that implements a DNS server for Unix systems by the ISC. You should first become familiar with the syntax of zone files (DNS records) that BIND uses. A good resource on this subject is the following link: [https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/3/html/Reference_Guide/s1-bind-zone.html](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/3/html/Reference_Guide/s1-bind-zone.html).

Then you should configure two types of entry, namely an A and PTR entry. The first type of entry is used for forward address resolution (translation of a DNS name to IP address) while the second is used in reverse DNS look up. Use the BIND program to insert DNS records of various types for different domains (e.g. SNAIL.cs.uoregon.edu) to your DNS server. Turn in a tar file of all your zone files with your project report.

Exercise 3:
The `dig` command is used to query a DNS server for resolution or reverse name resolution. Practice with this command and its features. You can use `dig` to check whether the records that you added to your own DNS server are accessible and provide correct information. You can add a fake entry for a web site (**www.amazon.com**) to your DNS server and specify your own IP address so any query for that web site is directed to your node. Given that it is so easy to add a fake record to DNS and hijacking a web site, investigate and explain how this problem is avoided in practice.

Exercise 4:
Your DNS server obviously can not resolve any query for the names (**google.com**) or IP addresses that it does not know about. In these cases, your server should be configured to forward the request to an external DNS server. Such a forwarding can be done in a recursive or iterative manner (one of the configuration choices). Configure your server to forward request for any missing record to UO DNS server and practice with both (recursive and iterative) forwarding options. Then, consider the provided list of 500 URLs (that represent some of the most popular URL over the Internet) and query your server to resolve these URL. These queries should be all forwarded to the UO DNS server. Use DNS performance tools such as `dnsping`
to measure the resolution latency of your server for each query and plot the Cumulative Distribution Function (CDF) of these latency values. Note that the DNS servers cache a missing record that they obtain for a request which in turn improves the response time for the next request for the same record. Repeat your queries for some of the 500 URLs and check produce the CDF of resolution latency values for this second round. Explain whether and how much the look up performance has improved as a result of caching.