CS39006: Networks Laboratory

Assignment 6

Student Information

- Bratin Mondal - 21CS10016
- Somya Kumar - 21CS30050

Running the code

Note: Our submission is currently structured to run on the same machine. To run the code on different machines, follow the instructions given below.

1. Client and Server on the Same Machine

Setup Instructions:

1. In `simDNSClient.c`, set the `#define` for `LOCAL_IP_ADDRESS` to `127.0.0.1`, `DEST_IP_ADDRESS` to `127.0.0.2`, and `INTERFACE` to `lo`.
2. In `simDNSServer.c`, set the `#define` for `LOCAL_IP_ADDRESS` to `127.0.0.2` and `INTERFACE` to `lo`.
3. Open Terminal and find the MAC address of the machine by running the command:

   ```
   ifconfig -a
   ```

   Find the MAC address of the machine from WiFi or Ethernet.

   Example:

   ```
   wlp0s20f3: flags=4098<BROADCAST,MULTICAST>  mtu 1500
       ether ac:74:b1:ac:5c:1a  txqueuelen 1000  (Ethernet)
       RX packets 19957  bytes 14112478 (14.1 MB)
       RX errors 0  dropped 0  overruns 0  frame 0
       TX packets 12390  bytes 3547387 (3.5 MB)
       TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
   ```

   Here, the MAC address is `ac:74:b1:ac:5c:1a`.
4. The MAC address of the machine can either be `#define` in the code or passed as an argument while running the code.
5. To `#define` the MAC address in the code, set the `#define` for `SOURCE_MAC` and `DEST_MAC` in `simDNSClient.c` and `SOURCE_MAC` in `simDNSServer.c` to the MAC address found in the previous step. Now use the makefile to compile the code. Commands:

   ```
   make all
   ```

   To run the server:

   ```
   sudo ./server
   ```

   To run the client:

   ```
   sudo ./client
   ```
6. To pass the MAC address as an argument while running the code, use the following commands:

   ```
   make all
   ```

   To run the server:

   ```
   sudo ./server <SOURCE_MAC>
   ```

   To run the client:

   ```
   sudo ./client <SOURCE_MAC> <DEST_MAC>
   ```
7. To run more than one client, change the `#define` for `LOCAL_IP_ADDRESS` in `simDNSClient.c` to starting from `127.0.0.3` for client 2, `127.0.0.4` for client 3, and so on. The `DEST_IP_ADDRESS` should be set to `127.0.0.2` for all clients. The `#define` for `SOURCE_MAC` should be set to the MAC address of the machine in `simDNSClient.c` and `DEST_MAC` should be set to the MAC address of the server. Now use the makefile to compile the code.

2. Client and Server on Different Machines

1. Both the devices should be connected to the same network. Both should be connected through `Ethernet` or `GUEST_SECURED` or `CAMPUS_SECURED` or `STUDENT_SECURED`. If the devices are connected to different networks, the devices will not be able to communicate. For example, if one device is connected to `GUEST_SECURED` and the other is connected to `CAMPUS_SECURED`, the devices will not be able to communicate. But if both devices are connected to `GUEST_SECURED`, they will be able to communicate.
2. From terminal, run the command:

   ```
   ifconfig -a
   ```

   For the interface that is connected to the network, find the `inet` address and the `HWaddr` (MAC address) of the device and the interface name.

   Example:

   ```
   enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
       inet 10.117.14.196  netmask 255.255.255.0  broadcast 10.117.14.255
       inet6 fe80::d2fc:1f6f:51f6:8ffc  prefixlen 64  scopeid 0x20<link>
       ether c8:5a:cf:ab:58:a0  txqueuelen 1000  (Ethernet)
       RX packets 124664  bytes 130607377 (130.6 MB)
       RX errors 0  dropped 1473  overruns 0  frame 0
       TX packets 68703  bytes 38341807 (38.3 MB)
       TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
   ```

   Here, the `inet` address is `10.117.14.196`, the `HWaddr` is `c8:5a:cf:ab:58:a0`, and the interface name is `enp2s0`. This is the ethernet interface.

   ```
   wlp0s20f3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
       inet 10.145.26.11  netmask 255.255.128.0  broadcast 10.145.127.255
       inet6 fe80::f41e:1d70:4e2b:1cf5  prefixlen 64  scopeid 0x20<link>
       ether ac:74:b1:ac:5c:1a  txqueuelen 1000  (Ethernet)
       RX packets 20588  bytes 14143425 (14.1 MB)
       RX errors 0  dropped 0  overruns 0  frame 0
       TX packets 12461  bytes 3558093 (3.5 MB)
       TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
   ```

   Here, the `inet` address is `10.145.26.11`, the `HWaddr` is `ac:74:b1:ac:5c:1a`, and the interface name is `wlp0s20f3`. This is the WiFi interface.
3. In this method find out the `inet` address, the `HWaddr`, and the interface name of both server and client. Let's call the `inet` address of the server as `SERVER_IP_ADDRESS`, the `HWaddr` of the server as `SERVER_MAC`, the interface name of the server as `SERVER_INTERFACE`, the `inet` address of the client as `CLIENT_IP_ADDRESS`, the `HWaddr` of the client as `CLIENT_MAC`, and the interface name of the client as `CLIENT_INTERFACE`.
4. In `simDNSClient.c`, set the `#define` for `LOCAL_IP_ADDRESS` to `CLIENT_IP_ADDRESS`, `DEST_IP_ADDRESS` to `SERVER_IP_ADDRESS`, and `INTERFACE` to `CLIENT_INTERFACE`.
5. In `simDNSServer.c`, set the `#define` for `LOCAL_IP_ADDRESS` to `SERVER_IP_ADDRESS` and `INTERFACE` to `SERVER_INTERFACE`.
6. The MAC address of the machine can either be `#define` in the code or passed as an argument while running the code.
7. To `#define` the MAC address in the code, set the `#define` for `SOURCE_MAC` and `DEST_MAC` in `simDNSClient.c` to `CLIENT_MAC` and `SERVER_MAC` respectively and `SOURCE_MAC` in `simDNSServer.c` to `SERVER_MAC`. Now use the makefile to compile the code. Commands:

   ```
   make all
   ```

   To run the server:

   ```
   sudo ./server
   ```

   To run the client:

   ```
   sudo ./client
   ```
8. To pass the MAC address as an argument while running the code, use the following commands:

   ```
   make all
   ```

   To run the server:

   ```
   sudo ./server <SERVER_MAC>
   ```

   To run the client:

   ```
   sudo ./client <CLIENT_MAC> <SERVER_MAC>
   ```

Additional Information:

- The probability of packet loss can be set in the code. The default value is 0.5. It can be changed by changing `#define` for `PROB` in `simDNSServer.c`.
- The VERBOSE mode can be enabled by setting the `#define` for `VERBOSE` in `simDNSServer.c` to see the response of the server.
- The debugging statements can be enabled by setting the `#define` for different types of debugging statements in `simDNSServer.c` and `simDNSClient.c`.