University of East London

AIN SHAMS UNIVERSITY (FACULTY OF ENGINEERING)

I-CHEP

CESS (2024-2025)

DHCP Phase Two Report

CSE351: COMPUTER NETWORKS

Name	ID
Saifeldin Haitham	21P0307
Mohamed Ehab Badr	21P0375
Omar Baher Hussein	21P0315
Saifeldin Mohamed Yousry	21P0362

Course Coordinators:

Dr. Ayman Bahaa eldin Eng. Noha Wahdan

Contents

Table of Figures

Introduction

Overview

For the second phase in our project, we implemented a server that listens on the broadcast channel of a network, this means that all DHCP clients on that network which begin their requests by sending DHCPDISCOVER on the broadcast channel will be seen by the server. This phase is only about giving a responsive server that responds to real-life clients, in further phases the server will then filter the messages on the broadcast channel to then deal with DISCOVER and REQUEST messages accordingly.

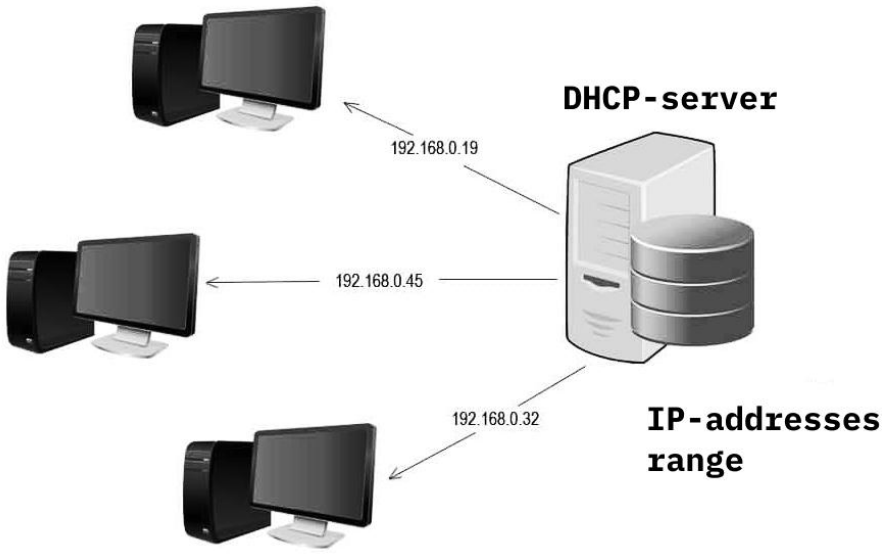

Figure 1 DHCP assignment

Implementation

1-

```
1   import socket
2   import binascii
3
4   # Server configuration
5   SERVER_PORT = 67  # Port for the DHCP server
6   BROADCAST_ADDRESS = "255.255.255.255"
7
```

Figure 2 Imports

Here we imported the two libraries:

- Socket:
- Binascii: For decoding the DHCP messages by clients with the <hexlify> function since they do not come in standard utf-8 format.

Then, We defined the port on which the server is listening (67) as it is the standard for DHCP servers, and the broadcast address on which the server replies

2-

```
9   def main():
10      # Create a UDP socket
11      server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
12
13      # Enable broadcast
14      server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
15
16      # Bind to all interfaces (0.0.0.0) and port 67
17      server_socket.bind(("0.0.0.0", SERVER_PORT))
18      print(f"Server is running and listening on all interfaces (port {SERVER_PORT})")
```

Figure 3 Server setup using IPv4, UDP, and broadcast on port 67.

In this main we define that the server will use ipv4 address and UDP as the transport layer protocol, we then allow broadcast messages, then bind the server on 0.0.0.0 (ie. Binds to all network interfaces) so it listens to all clients and binds it to port 67 (will be the target from clients). **the print statement is for debugging purposes to check if this part executes successfully.

3-

```
while True:
    try:
        # Wait for a message from the client
        data, client_address = server_socket.recvfrom(1024)
        print(f"Received message from {client_address}: {binascii.hexlify(data).decode()}")
```

Figure 4 Message decoding

This part continuously checks for messages on the socket, decodes it with the , decodes it with the <hexlify> function from the binascii library.

4-

```
# Check if the client address is invalid (0.0.0.0)
if client_address[0] == "0.0.0.0":
    print(f"Client has no valid IP address. Broadcasting response to the network.")
    # Send the response via broadcast
    response_message = b"DHCP server response"
    server_socket.sendto(response_message, (BROADCAST_ADDRESS, client_address[1]))
else:
    # Normally response to the specific client
    response_message = b"Test server response to your request"
    server_socket.sendto(response_message, client_address)
    print(f"Response sent to {client_address}")
```

Figure 5 Handling client address and message filtering

If the client address is “0.0.0.0” this means it wasn’t assigned an IP yet, ideally the server will further filter it as a discover or request message and treat it accordingly by sending a response over the broadcast channel.

If the client has an IP already but still sent a message to the server’s socket (eg. For lease renewal, release notification, etc....) it responds to the client’s address only not on the broadcast challenge.

Run Example

```
[Running] python -u "f:\ASU\YEAR 4\Semester 1\CSE351 - Networks\Project\Networks\Phase 2\server_utils.py"
Server is running and listening on all interfaces (port 67)
Received message from ('0.0.0.0', 68):
01010600c332176f00000000000000000000000000000000000000008a4b1bc3e7660000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000638253633501013d07018a4b1bc3e766390205dc3c0f616e64726f69642d646863702d31330c145265646d692d4e6f74652d31312d50726f2d35
47370c0103060f1a1c333a3b2b726c5000ff
Client has no valid IP address. Broadcasting response to the network.
Received message from ('0.0.0.0', 68):
01010600c332176f00010000000000000000000000000000000000008a4b1bc3e7660000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000638253633501033d07018a4b1bc3e7663204c0a801d93604c0a80101390205dc3c0f616e64726f69642d646863702d31330c145265646d692d4e
6f74652d31312d50726f2d3547370c0103060f1a1c333a3b2b726cff
Client has no valid IP address. Broadcasting response to the network.
```

Figure 6 Live code execution and client-server interaction

- Here we see a live version of the code running, the first printed statement indicates the socket has been configured and the server is now listening.
- On our mobile which is connected to the same network as the laptop on which this server is running, we forget the network then try to connect to it again, this results in 2 messages on the socket (Ideally, these are DISCOVER and REQUEST messages. Since the DHCP server of the router itself is probably replying to the client).
- The client addresses are 0.0.0.0 which is correct and from port 68 which is correct too. Then the message is printed (HEX-format)