NETWORK ADDRESS TRANSLATION

September 19, 2019

Gerrit Burger - 21261687

Jacques Huysamen - 20023669

Computer Science 354 - Networking

Contents

1 Project Description

Network Address Translation is a method used to remap one IP address space to another by modifying the information in the IP header of a packet, while this packet is in transit. This method was developed because of the limited size of the IPv4 address space. In this model, two different types of clients are specified, internal and external clients. Internal clients are hosts in the local network itself and external clients belonging to an external network. In this project, a simulation router was implemented that has basic NAT functionality following the traditional NAPT model, and the ability to rout packets between clients.

The term paquet will be used to refer to packets being sent in the simulated network. In the simulated network model, the following scenarios hold:

- When an Internal client sends a paquet to another Internal client, it is sent without changing the header information.
- When an internal client sends a paquet to an external client, the paquet header information is modified and an entry is written or refreshed in the NAT table.
- When an External client sends a paquet to an Internal client the paquet is routed according to the corresponding entry in the NAT table if it exists. With the use of port forwarding, it is always possible for an external client to connect to an internal client.
- When an external client sends a paquet to another external client, the paquet is dropped.

A NAT table is used to monitor active connections between internal and external clients. When an internal client creates a new connection to an external client an entry is written to the NAT table. If there is an existing NAT table entry for this pair of clients, the entry is refreshed.

2 Project Features

2.1 Client Features

Required features included for the client:

- Ability to specify the type of client to be created, i.e. internal or external via the command line.
- Ability to send normal messages to other internal or external clients.
- Ability to ping other clients.
- Ability to send echo messages to other clients.

Additional features included for the clients:

- Command-line interface used to enter messages to be sent to other clients, and display received messages.
- ICMP error messages.

2.2 Natbox Features

Required features included for the natbox:

- The natbox's unique IP and MAC address is broadcast to all clients connecting to the natbox.
- Multiple clients can connect to the natbox.
- Internal clients are assigned an available IP address from the generated IP pool using a DHCP implementation.
- Assigned IP address of an internal client is released back into the pool if the client disconnects.
- Ability to process and modify packet header information to be sent to clients.

- NAT table to translate IP addresses.

Additional features implemented for the natbox:

- Port forwarding, to enable external clients to communicate with internal clients without an entry for the pair in the NAT table.
- Command-line interface to show information of clients connecting and disconnecting to the natbox.
- ICMP error messages.

3 Program Description

3.1 Natbox

3.1.1 Initialization and client connection

When the simulation is started a new natbox is created and the natbox will wait for clients to connect. When the natbox is initialized it will find the local machine's IP address and set it as its IP address, and generate a unique MAC address to be set as its MAC address. A pool of generated IP addresses is created to be allocated to internal clients. A UDP socket is created and a new thread is spawned that waits for clients to connect, this socket is not bound to any connection and thus open to any client to send a message. Using the ARP protocol the natbox's MAC and IP addresses are sent to the client when it connects. The DHCP protocol is then started when the natbox receives a discover message from an internal client that wants to connect, the natbox will then get an available IP address from the pool and send it to the requesting client as an offer. If there are no IP addresses available, a message will be sent to the client telling it to disconnect. If the client accepts this IP address a request message will be received by the natbox and the IP address that was offered will be removed from the pool. An acknowledge message is then again sent to the client to let the client know the connection was successful. A new thread is spawned that will handle messages for this newly connected client. When an external client connects to the natbox a generated external IP address is sent to the client and this address is stored by the natbox. A receive thread is then created for this external client that will handle all the messages for this client. After an external or internal client has been connected, and the client's receive thread has been spawned, the initial thread will again wait for a new client to connect and repeat the process.

3.1.2 Packet forwarding

On a client's receive thread a TCP socket is created and stored in a dictionary. When a client wants to send a message to another client, it sends a paquet to the natbox over this TCP socket. The header of this packet consists of the sending client's IP address and the receiving client's IP address, and the message to be sent is in the body of the packet.

The natbox will use these two IP addresses to decide how to respond as mentioned in the scenarios in section 1. The natbox will then change the packet's header information if needed, and send the packet to the receiving client. If the destination IP address is not a valid IP address or the IP address can not be found as a connected client an ICMP error code will be sent to the sending client. When the error code or message has successfully been routed the receive thread will wait for another message to be sent over the TCP connection and repeat the process.

3.1.3 NAT table refreshing

When the natbox receives a paquet that has to be routed from an internal to an external client the NAT table is checked to see if there is an existing connecting for this pair of clients. If there is no such entry in the NAT table, an entry is created. When a new entry is created a new thread is spawned that will monitor the time of this entry. If an entry did exist for the connection pair, the time is updated for the entry. Each of the time threads spawned to monitor the time of a connection pair will monitor the time elapsed since the NAT table was last used if this time reaches a certain timeout the entry will be removed from the NAT table.

3.2 Client

3.2.1 Initialization

When a client instance is started an argument is used to specify if it is an internal or external client, a 0 is used for an external client and a 1 for an internal client. When an external client connects it will generate a unique MAC address and IP address starting with "212.30" to identify as an external client. The client will then send the MAC and IP address to the natbox, and the natbox will send back a message to acknowledge if they are valid. If the client connecting is an internal client the DHCP protocol will be started by the client sending a discovery message to the natbox over the UDP socket. If there are no internal IP addresses available the natbox will relay this message to the connecting client and it will disconnect. If the pool is not empty the client will receive an offer from the natbox consisting of a generated internal IP address to be used by the client as its IP address. The client will then send a request message back to natbox telling it that it has accepted the offered IP. The client will then receive a final acknowledge message from the natbox telling it that the process was successful.

3.2.2 Sending and receiving messages

When the initialization process for a client is complete a thread is spawned that will handle the receiving of messages for the client. This main thread will make use of the command-line interface to prompt the user to enter information and a message if a message wants to be sent. The receiving thread will wait for a message from the natbox on the TCP socket, the message will then checked to see if it is an ICMP error code or a valid message. If it is an error code the corresponding error message will be displayed if it is a message the message and client from whom the message was sent will be displayed. When a message wants to be sent the user will enter the IP address of the user to whom the message must be sent, and the message to be sent via the command-line interface. A paquet is constructed containing the sending and receiving client IP addresses in the header and the message in the body of the paquet, this paquet is sent to the natbox via the TCP socket to be routed to the receiving client.

4 Data Structures

4.1 Data Structures

4.1.1 Natbox Data Structures

As the natbox has to keep track of two sets of IP address spaces and active connections of internal and external clients in the NAT table a few different data structures were used. When the natbox is initialized the pool of internal IP addresses is created. These IP addresses are stored in a list, when an internal client gets an IP assigned to them the IP can then be removed from the list. When an internal client disconnects its IP address can then be appended back into the list. Two other lists are also created that each respectively keep track of all the connected internal and external clients. When a new client's receiving thread gets spawned a TCP connection between the client and the natbox is created and stored in a dictionary to enable the natbox to communicate with all clients. The client's IP address is used as the key in this dictionary and the socket object as the value. A dictionary is also used to keep track of all connected clients MAC addresses to ensure no 2 clients in the simulation have the same MAC address. A dictionary was used for the NAT table, where the key is the internal client's IP and port number separated by a ":" and the value is the external client's IP address. Another dictionary is used to dynamically refresh the NAT table, for each entry in the NAT table a timer is added to the time dictionary. When one of the timers exceeds the time limit that entry is removed from the NAT table.

4.1.2 Client Data Structures

As all of the routing is done by the natbox the client does not need many data structures. When a client wants to send a message the command line interface gives the user a series of prompts to enter information. The client indicates which type of messages should be sent, and the recipient's IP address and port number if it is an internal client is entered via the command-line interface. The actual message is then entered and all of the required information is then acquired. A paquet is built as a byte array to be sent via the TCP socket to the natbox.

5 Experiments

5.1 Allocation of internal client IP address from address pool

5.1.1 Hypothesis

The connecting internal client will receive an IP address from the natbox if there are any internal IP addresses available from the IP pool. If there are no IP addresses available the client will be notified to disconnect.

5.1.2 Method

A natbox instance is created and waits for clients to connect, for this experiment the pool size is set to a maximum of 2 internal clients to be connected at the same time. A few internal clients are connected to the natbox and the results are observed.

5.1.3 Results

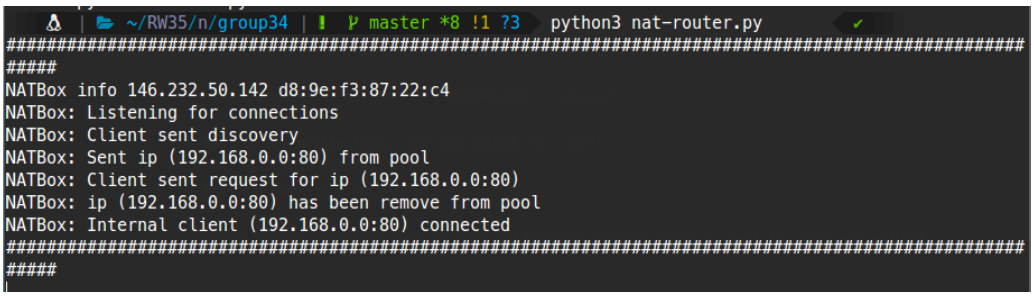
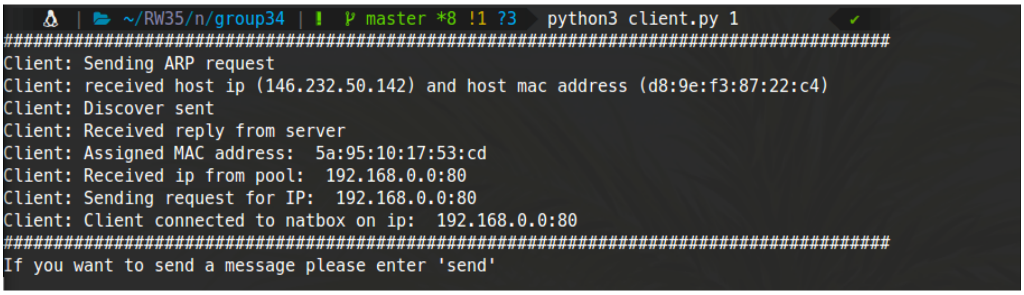
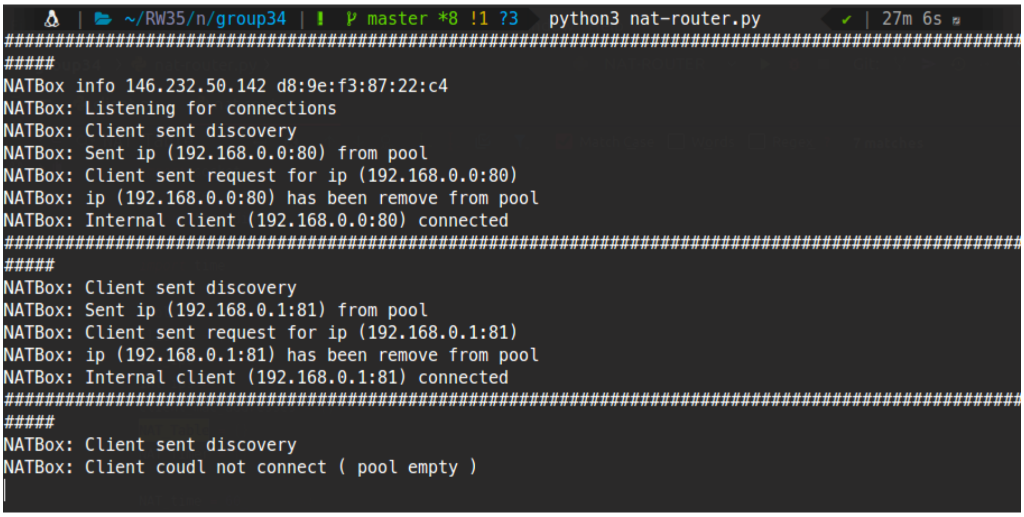


Figure 5.1: Natbox is initialized



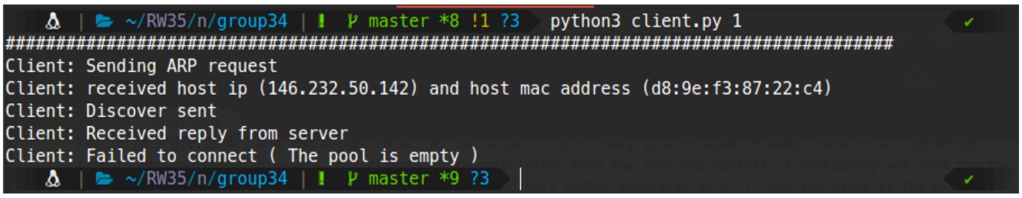
```
~/RW35/n/group34 | ! master *8 !1 ?3  python3 client.py 1
##########################################################################################
Client: Sending ARP request
Client: received host ip (146.232.50.142) and host mac address (d8:9e:f3:87:22:c4)
Client: Discover sent
Client: Received reply from server
Client: Assigned MAC address:  5a:95:10:17:53:cd
Client: Received ip from pool:  192.168.0.0:80
Client: Sending request for IP:  192.168.0.0:80
Client: Client connected to natbox on ip:  192.168.0.0:80
##########################################################################################
If you want to send a message please enter 'send'
```

Figure 5.2: First internal client is connected



```
~/RW35/n/group34 | ! master *8 !1 ?3  python3 nat-router.py   27m 6s
#####################################################################################################
#####
NATBox info 146.232.50.142 d8:9e:f3:87:22:c4
NATBox: Listening for connections
NATBox: Client sent discovery
NATBox: Sent ip (192.168.0.0:80) from pool
NATBox: Client sent request for ip (192.168.0.0:80)
NATBox: ip (192.168.0.0:80) has been remove from pool
NATBox: Internal client (192.168.0.0:80) connected
#####################################################################################################
#####
NATBox: Client sent discovery
NATBox: Sent ip (192.168.0.1:81) from pool
NATBox: Client sent request for ip (192.168.0.1:81)
NATBox: ip (192.168.0.1:81) has been remove from pool
NATBox: Internal client (192.168.0.1:81) connected
#####################################################################################################
#####
NATBox: Client sent discovery
NATBox: Client coudl not connect ( pool empty )
```

Figure 5.3: Three internal clients trying to connect to the natbox



```
~/RW35/n/group34 | ! master *8 !1 ?3  python3 client.py 1
##########################################################################################
Client: Sending ARP request
Client: received host ip (146.232.50.142) and host mac address (d8:9e:f3:87:22:c4)
Client: Discover sent
Client: Received reply from server
Client: Failed to connect ( The pool is empty )
~/RW35/n/group34 | ! master *9 ?3
```

Figure 5.4: Third client trying to connect to the natbox

5.1.4 Conclusion

As seen from the results we can accept our hypothesis, the first 2 clients could successfully connect to the natbox, when the third client tried to connect it was signaled that it could not connect and the program was terminated.

5.2 External to external client packet dropping

5.2.1 Hypothesis

When one external client tries to communicate with another external client the packet will be dropped.

5.2.2 Method

Two external clients are connected to the natbox. One of the external clients then tries to send a ping message to the other connected external client and the results are observed.

5.2.3 Results

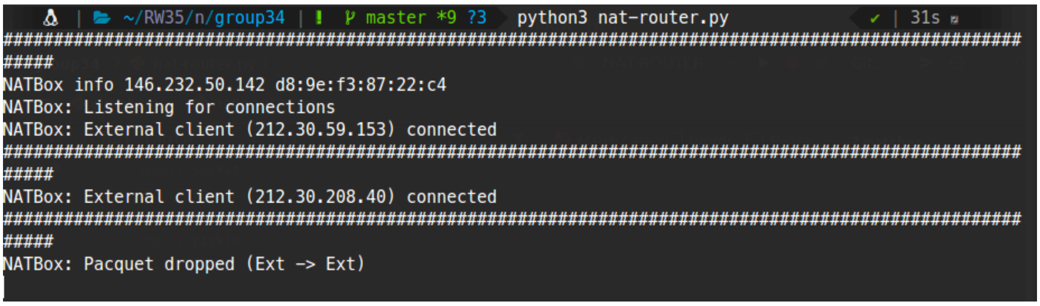


Figure 5.5: Natbox shows that packet is dropped

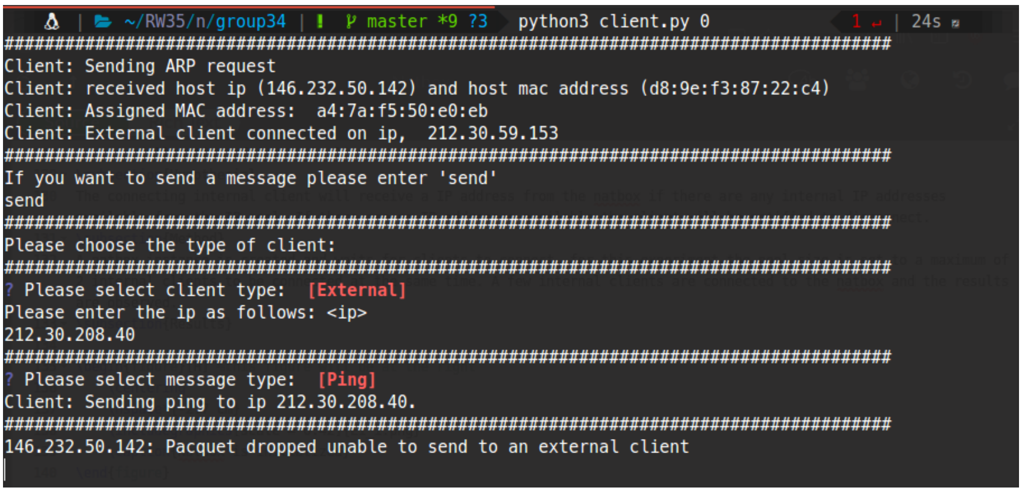


Figure 5.6: Client trying to send the message is notified that the packet was dropped

5.2.4 Conclusion

As seen from the results our hypothesis can be accepted, the packet is succesfully dropped.

5.3 Pacquet routing

5.3.1 Hypothesis

When a client sends to another client the pacquets are routed through the natbox and the pacquet data is changed, except for when an internal client sends to another internal client then the data is not changed.

5.3.2 Method

When a client connects as either internal or external and they choose to send a message they can choose to either send to an external or internal client. After choosing the message type the message is sent to the natbox to route the pacquet to the chosen client.

5.3.3 Results

Sending a ping from an internal to an external client:

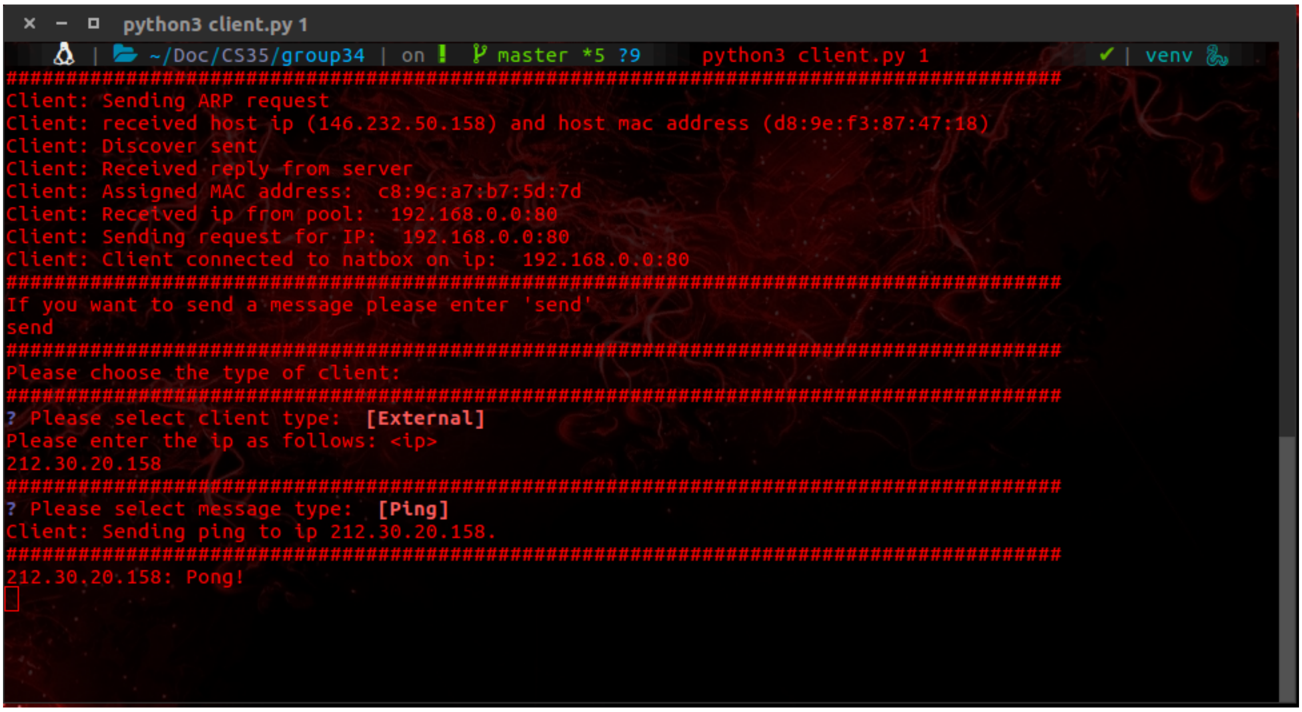


Figure 5.7: Internal client sends pacquet to external client

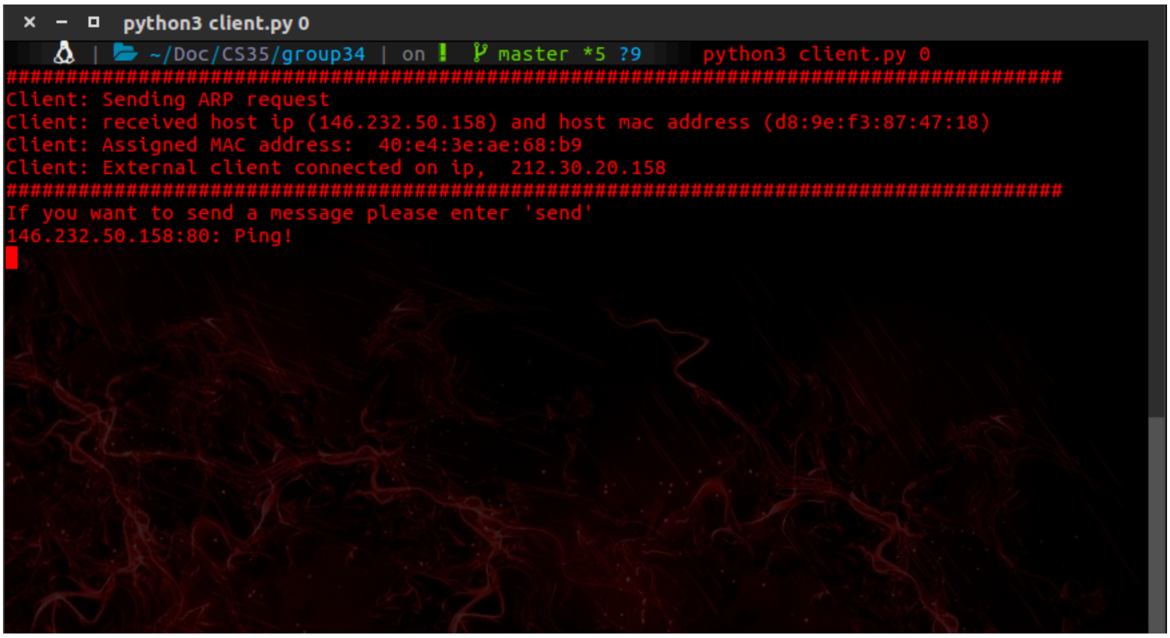


Figure 5.8: External client receives pacquet from internal client

Sending from an external to an internal client:

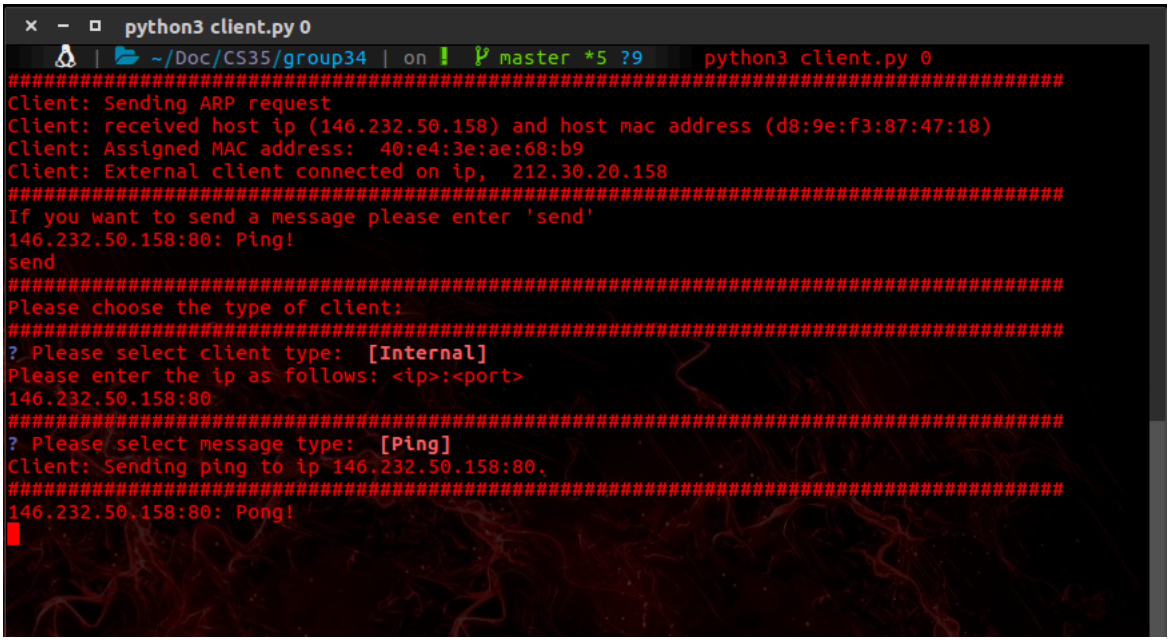


Figure 5.9: External client sends pacquet to internal client

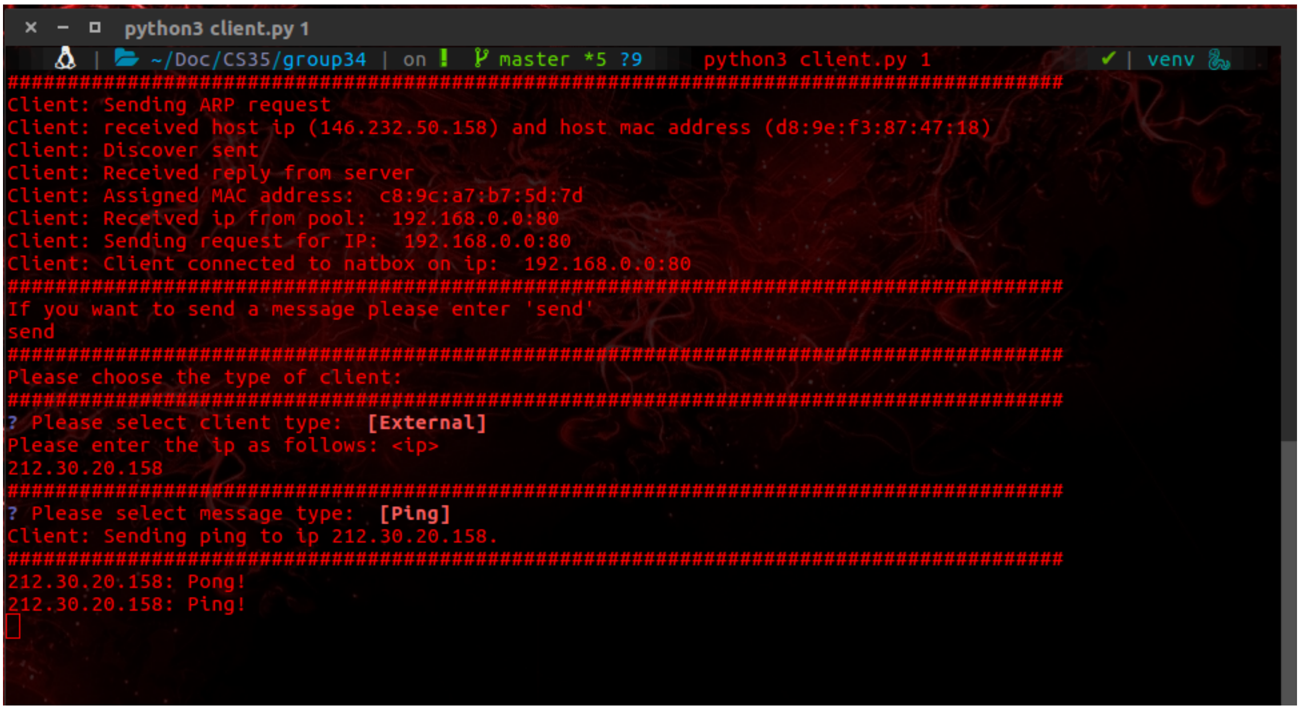


Figure 5.10: Internal client receives pacquet from external client

Sending from an internal to an internal client:

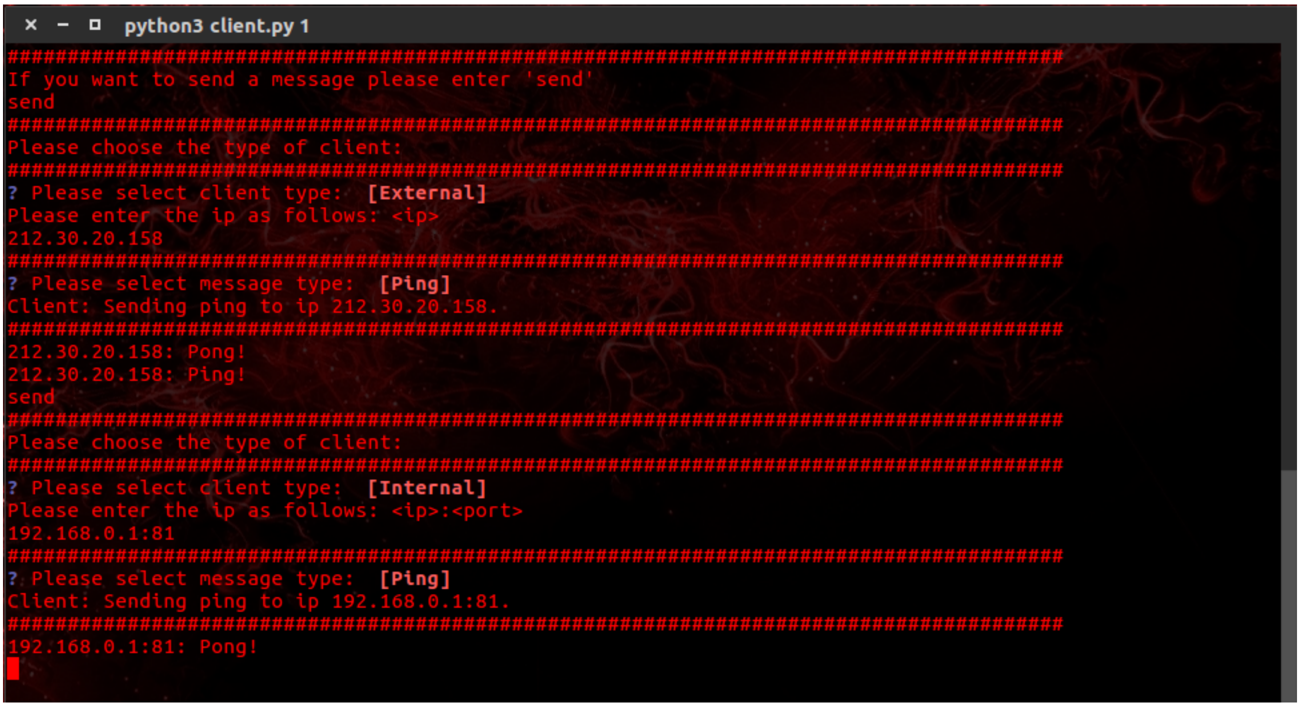


Figure 5.11: Internal client sends pacquet to internal client

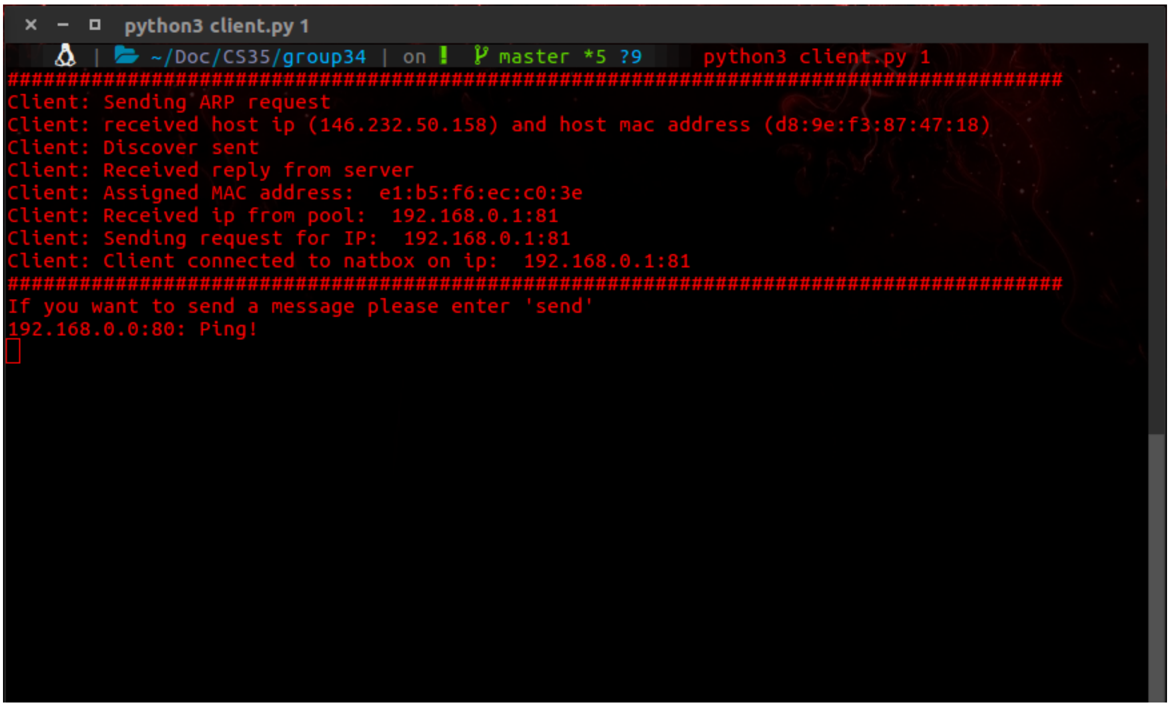


Figure 5.12: Internal client receives pacquet from internal client

5.3.4 Conclusion

From the results it is apparent that when an internal client sends to an external client the pacquet data is changed so that the pacquet is changed from "<internal-ip: port>:<external-ip>:<message>" to "<natbox-ip:port>:<message>" when the external client receives the pacquet. When an external client sends to an internal client the pacquet is changed from "<external-ip>:<natbox-ip:port>:<message>" to "<external-ip>:<message>" when the internal client receives the pacquet. Normally when an external client tries to send to an internal client a NAT table entry must have been created between the internal and external client, but since NAPT was used the internal client always has a port open on the natbox and can receive messages from external clients sending to its port, it goes without saying that if an external sends a message to an internal client and there is no NAT Table entry created it creates a NAT Table entry between the internal and external client. It is also clear that when an internal client sends a pacquet to an internal client the pacquet data is not changed. Hence the hypothesis is accepted.

5.4 Packet types

5.4.1 Hypothesis

For each of the three communication types, the paquets will successfully be routed from the sending client to the receiving client.

5.4.2 Method

Two internal and one external client is connected and each of the communication types are tested.

5.4.3 Results

Ping packet is sent from the internal client to an external client:

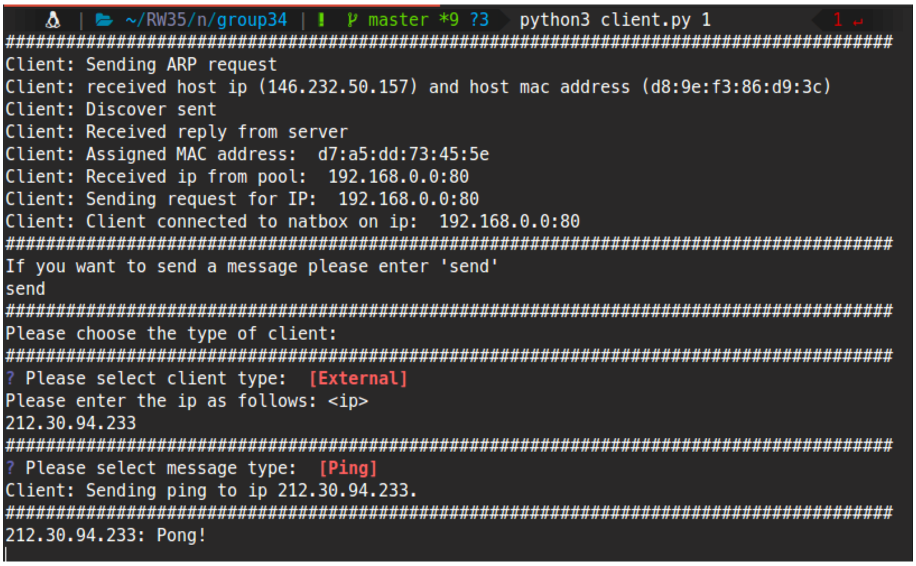


Figure 5.13: Internal client sends ping message and pong reply is received.

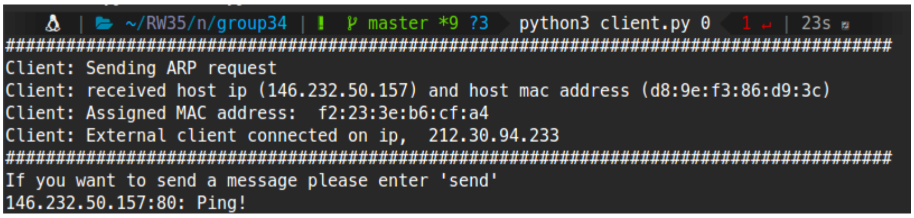


Figure 5.14: External client receives ping from the internal client

Echo packet is sent from the internal client to other internal client:

```
 ~/RW35/n/group34 |  master *9 !2 ?3  python3 client.py 1
##########################################################################################
Client: Sending ARP request
Client: received host ip (146.232.50.157) and host mac address (d8:9e:f3:86:d9:3c)
Client: Discover sent
Client: Received reply from server
Client: Assigned MAC address:  b4:fb:a4:a7:35:f9
Client: Received ip from pool:  192.168.0.2:82
Client: Sending request for IP:  192.168.0.2:82
Client: Client connected to natbox on ip:  192.168.0.2:82
##########################################################################################
If you want to send a message please enter 'send'
send
##########################################################################################
Please choose the type of client:
##########################################################################################
? Please select client type:  [Internal]
Please enter the ip as follows: <ip>:<port>
192.168.0.1:81
##########################################################################################
? Please select message type:  [Echo]
Client: Please enter the message you want to send.
This is a test!
Client: Echoing ip 192.168.0.1:81.
##########################################################################################
192.168.0.1:81: This is a test!
```

Figure 5.15: Internal client sends echo message and echo reply is received.

```
 ~/RW35/n/group34 |  master *9 !2 ?3  python3 client.py 1
##########################################################################################
Client: Sending ARP request
Client: received host ip (146.232.50.157) and host mac address (d8:9e:f3:86:d9:3c)
Client: Discover sent
Client: Received reply from server
Client: Assigned MAC address:  ed:ed:10:7d:fd:4b
Client: Received ip from pool:  192.168.0.1:81
Client: Sending request for IP:  192.168.0.1:81
Client: Client connected to natbox on ip:  192.168.0.1:81
##########################################################################################
If you want to send a message please enter 'send'
192.168.0.2:82: This is a test!
```

Figure 5.16: Internal client receives echo from the sending client

Normal message packet is sent from the internal client to other internal client:

```
################################################################################
Client: Sending ARP request
Client: received host ip (146.232.50.157) and host mac address (d8:9e:f3:86:d9:3c)
Client: Discover sent
Client: Received reply from server
Client: Assigned MAC address:  15:d0:73:7b:85:5c
Client: Received ip from pool:  192.168.0.3:83
Client: Sending request for IP:  192.168.0.3:83
Client: Client connected to natbox on ip:  192.168.0.3:83
################################################################################
If you want to send a message please enter 'send'
send
################################################################################
Please choose the type of client:
################################################################################
? Please select client type:  [Internal]
Please enter the ip as follows: <ip>:<port>
192.168.0.4:84
################################################################################
? Please select message type:  [Normal message]
Client: Please enter the message you want to send.
yEeT!
Client: Sending message to ip 192.168.0.4:84.
################################################################################
```

Figure 5.17: Internal client sends normal message to other internal client.

```
################################################################################
Client: Sending ARP request
Client: received host ip (146.232.50.157) and host mac address (d8:9e:f3:86:d9:3c)
Client: Discover sent
Client: Received reply from server
Client: Assigned MAC address:  12:6e:e2:e4:bb:58
Client: Received ip from pool:  192.168.0.4:84
Client: Sending request for IP:  192.168.0.4:84
Client: Client connected to natbox on ip:  192.168.0.4:84
################################################################################
If you want to send a message please enter 'send'
192.168.0.3:83: yEeT!
```

Figure 5.18: Internal client receives normal message from the sending client

5.4.4 Conclusion

As seen from the above results for all three of the communication types paquets are successfully built, sent to the natbox and routed to the correct receiving client. We can thus accept our hypothesis.

5.5 NAT Table entries and timeouts

5.5.1 Hypothesis

An internal client sends to an external client and a NAT table entry is created that links the internal and external client. The NAT table entry time's out after a configurable amount of time. Should the external client send a message to the internal client or vice versa the timer is reset for that entry in the NAT table.

5.5.2 Method

An internal client sends a pacquet to an external client and a NAT table entry is created between the two clients. After the entry times out the NAT table entry is removed, but should the clients communicate the timer is reset.

5.5.3 Results

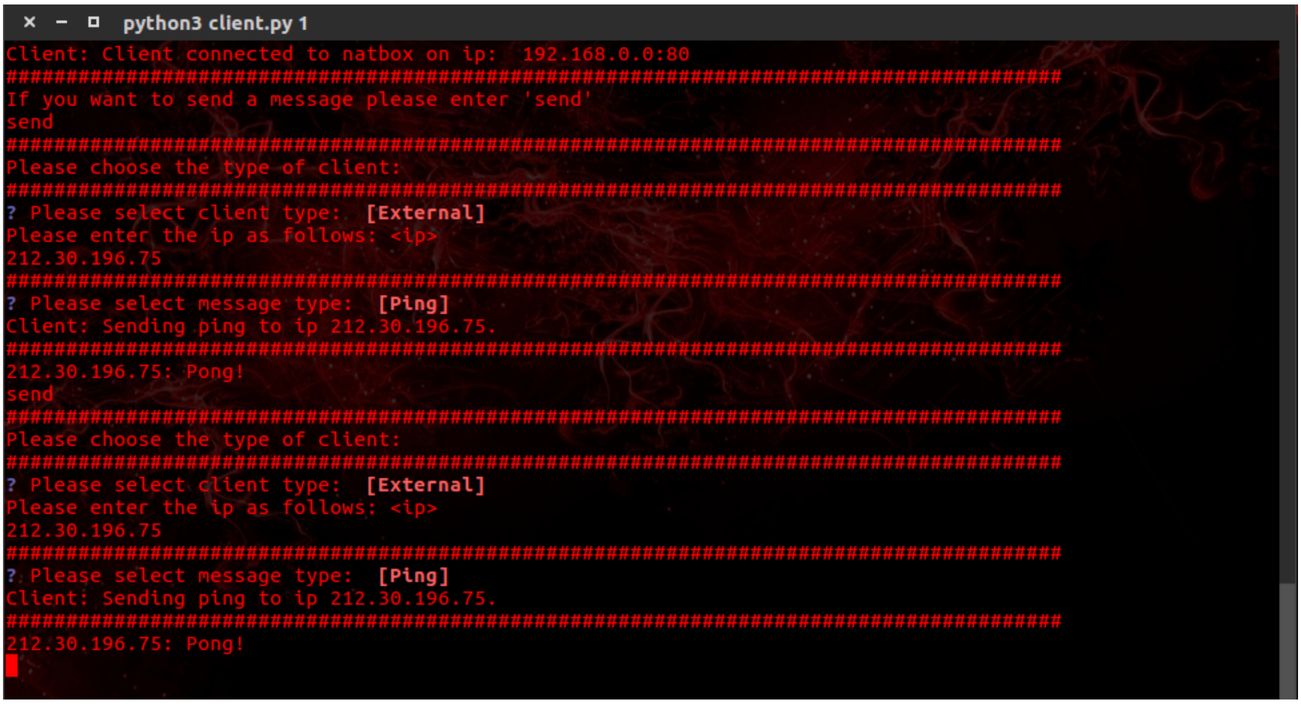


Figure 5.19: Internal client sends pacquet to external client

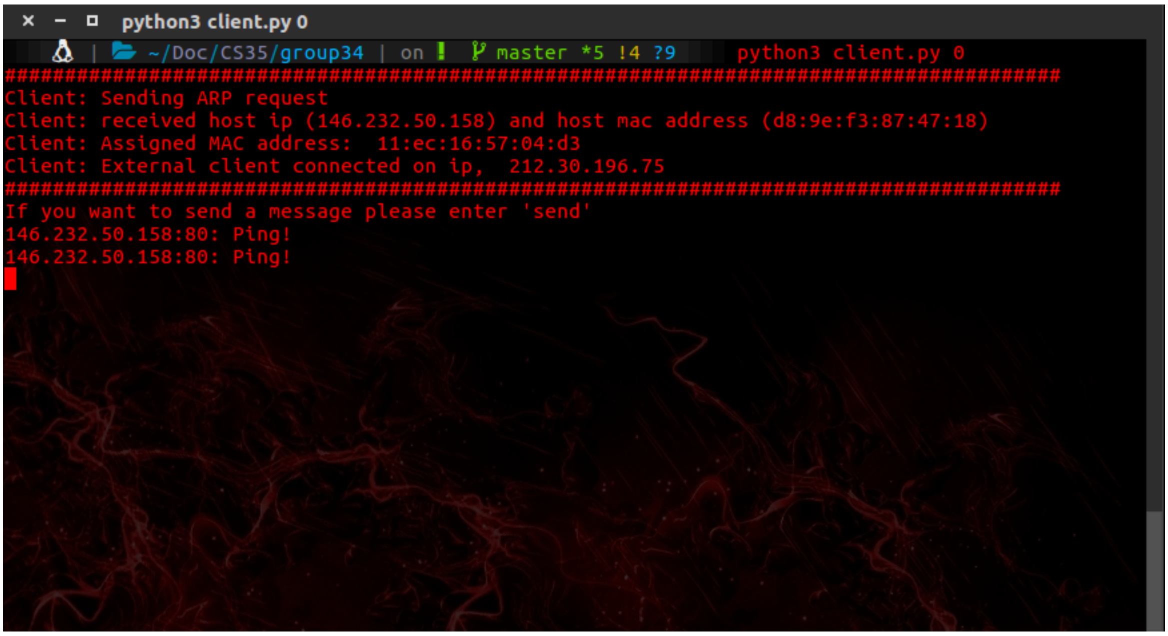


Figure 5.20: External client receives pacquet from internal client

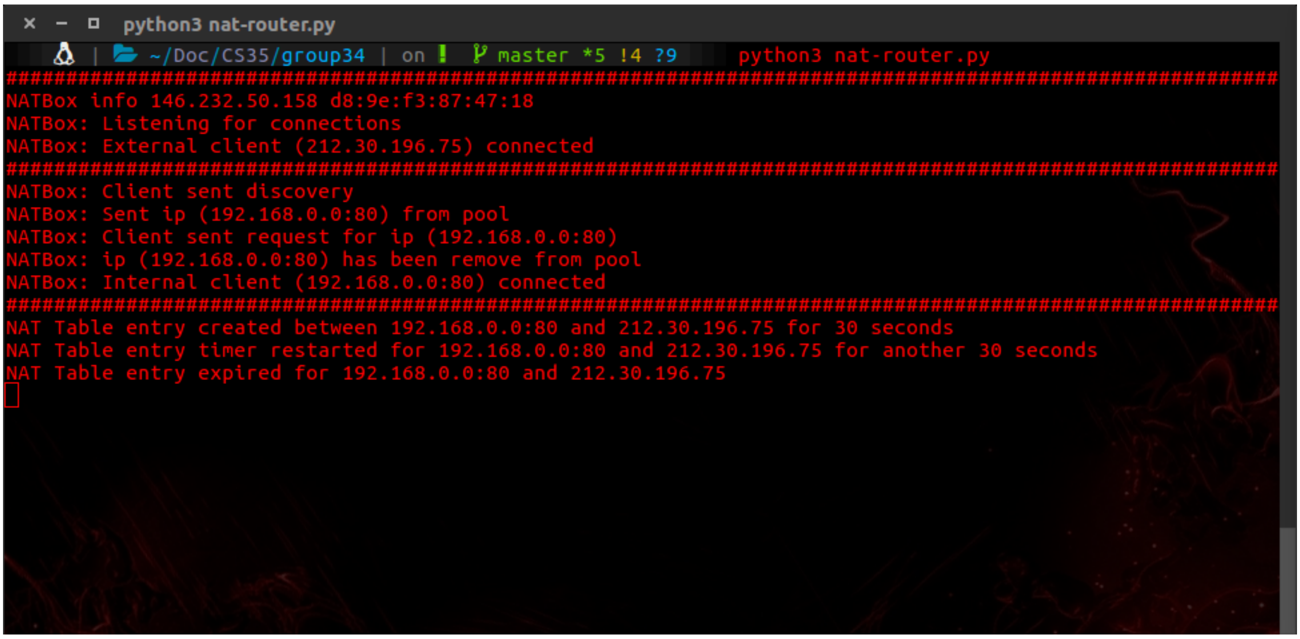


Figure 5.21: What happens to the NAT Table entry during the test

5.5.4 Conclusion

From the results above it is apparent the NAT table entry that was created when the internal client sends a pacquet to the external client. The entry's timer was set to a 30 seconds timeout, but before the timer ran out another message was sent between the clients and the timer was reset to 30 seconds, after the 30 seconds ran out the NAT table entry was removed. Hence the hypothesis can be accepted.

5.6 ICMP errors

5.6.1 Hypothesis

When an illegal paquet is received by the natbox it will find the ICMP error code for the error and relay this error code to the sending client to let the client know what the illegal operation was.

5.6.2 Method

Two clients are connected and 4 different types of situations are replicated that will each trigger a different ICMP error.

5.6.3 Results

An incorrect IP address is entered when a message is being sent, the IP address is too short and ICMP error code 12:2 is returned:

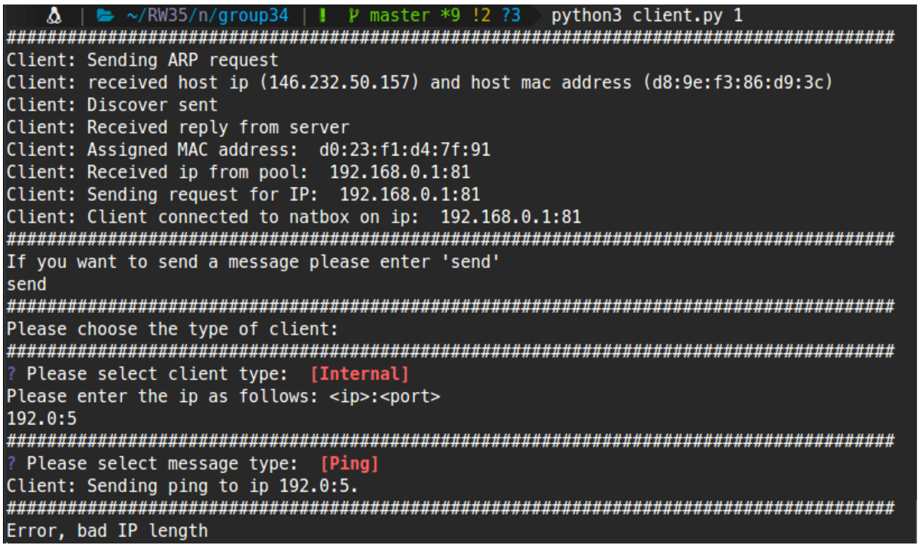


Figure 5.22: Illegal IP address entered

When a message is being sent to an internal client no port number is specified, and ICMP error code 12:1 is returned because there are missing: arguments.

```
 | ~/RW35/n/group34 | ! master *9 !2 ?3  python3 client.py 1                        1
#################################################################################
Client: Sending ARP request
Client: received host ip (146.232.50.157) and host mac address (d8:9e:f3:86:d9:3c)
Client: Discover sent
Client: Received reply from server
Client: Assigned MAC address:  c8:63:9b:4c:b7:5c
Client: Received ip from pool:  192.168.0.2:82
Client: Sending request for IP:  192.168.0.2:82
Client: Client connected to natbox on ip:  192.168.0.2:82
#################################################################################
If you want to send a message please enter 'send'
send
#################################################################################
Please choose the type of client:
#################################################################################
? Please select client type:  [Internal]
Please enter the ip as follows: <ip>:<port>
192.168.0.2
#################################################################################
? Please select message type:  [Ping]
Client: Sending ping to ip 192.168.0.2.
#################################################################################
Error, missing arguments
```

Figure 5.23: First internal client is connected

When a message is being sent to an internal client that is not currently connected to the natbox ICMP error code 3:1 is returned:

```
 | ~/RW35/n/group34 | ! master *9 !2 ?3  python3 client.py 1
#################################################################################
Client: Sending ARP request
Client: received host ip (146.232.50.157) and host mac address (d8:9e:f3:86:d9:3c)
Client: Discover sent
Client: Received reply from server
Client: Assigned MAC address:  39:ca:8e:df:43:b5
Client: Received ip from pool:  192.168.0.3:83
Client: Sending request for IP:  192.168.0.3:83
Client: Client connected to natbox on ip:  192.168.0.3:83
#################################################################################
If you want to send a message please enter 'send'
send
#################################################################################
Please choose the type of client:
#################################################################################
? Please select client type:  [Internal]
Please enter the ip as follows: <ip>:<port>
192.168.0.3:85
#################################################################################
? Please select message type:  [Ping]
Client: Sending ping to ip 192.168.0.3:85.
#################################################################################
146.232.50.157: Internal client not reachable.
```

Figure 5.24: Receiving internal client not reachable

When a message is being sent to an external client that is not currently connected to the natbox ICMP error code 3:1 is returned:

```
 ~/RW35/n/group34 | ! master *9 !2 ?3   python3 client.py 1
###############################################################################
Client: Sending ARP request
Client: received host ip (146.232.50.157) and host mac address (d8:9e:f3:86:d9:3c)
Client: Discover sent
Client: Received reply from server
Client: Assigned MAC address:  61:99:85:c4:8e:03
Client: Received ip from pool:  192.168.0.4:84
Client: Sending request for IP:  192.168.0.4:84
Client: Client connected to natbox on ip:  192.168.0.4:84
###############################################################################
If you want to send a message please enter 'send'
send
###############################################################################
Please choose the type of client:
###############################################################################
? Please select client type:  [External]
Please enter the ip as follows: <ip>
212.30.230.245
###############################################################################
? Please select message type:  [Ping]
Client: Sending ping to ip 212.30.230.245.
###############################################################################
146.232.50.157: External client not reachable.
```

Figure 5.25: Receiving external client not reachable

5.6.4 Conclusion

As seen from the above results for each of the scenarios the correct ICMP error code is picked up by the natbox and replayed to the sending client. When the client receives the ICMP error code a message is displayed to let the user know what the error is. We can thus accept our hypothesis.

6 Conclusion

From the results gathered over the span of the project, (refer to the conclusions at the end of each experiment to see the results of the project), it is apparent that the NAT is a good solution to the problem of the limited size of the IPv4 address space, since a local network can easily dedicate internal IP's to clients connected within the same network and connect all of them to a natbox (which will handle communication from external clients to internal clients within the private network). The NAT also has the benefit of reusable private IP addresses, enhanced security by making the internal client IPs hidden from external clients and connecting a large number of clients to the global internet using a smaller number of public IP addresses and thereby conserving IP address space. NAT will also prevent any internal IP changes when you change your service provider.