CS 313 Project 3

Group 31: Network Address Translation (NAT)

Keagan Selwyn Gill: 22804897@sun.ac.za

25/03/2024

TABLE OF CONTENTS

INTRODUCTION:

The goal of this project was to implement a NAT-Box with the ability to connect multiple clients to the NAT-Box. As part of this implementation, it was also required to implement ICMP, an error-reporting protocol, as well as DHCP, a client-server protocol to dynamically allocate addresses to internal clients in a local network. Network Address Translation was thought up as a solution to the problem of there being too few IP addresses in IPv4 to be able to supply each computer connecting to the internet with a unique IP. NAT allows local computers, nodes, to form a network using non-unique IP's. This allows the nodes to be able to communicate with each other inside the network without packet modification, and without each having to have unique IP addresses. The NAT-Box is allocated one or more unique IP's by the ISP. These unique IP's are mapped to internal nodes when they want to send packets to nodes outside the local network. The packet header from the internal node is modified so it looks as if it originated from the NAT-Box.

UNIMPLEMENTED FEATURES:

I did not implement port forwarding, nor did I implement Ethernet frames in the packet construction.

ADDITIONAL FEATURES IMPLEMENTED:

- Minimal ICMP
- TCP Packet forwarding
- Internal IP releasing on lease timeout

DESCRIPTION OF FILES:

Source files:

- Client.java
- ClientConnection.java
- NATBox.java
- NATtableEntry.java
- NATUtilities.java

Shell files and make file:

- clientExt.sh
- clientInt.sh
- NatBox.sh
- Makefile

PROGRAM DESCRIPTION:

Client:

Internal Client:

The Internal client acts as a device that would like to be a part of the internal network. To connect to the NAT-Box, this device is allocated an Internal IP address via a DHCP implementation. The client can then send messages to other internal clients and any external clients that have already interacted with the NAT-Box.

External Client:

The External client acts as a device that would like to communicate with a device that is contained in the internal network. When an external client interacts with the NAT-Box, the NAT-Box remembers its IP and allows its internal clients to the communicate with it. External clients are allowed to communicate with Internal clients, but the NAT-Box drops any packets that are deemed to be communications between two external clients, as this must be routed via an external router.

NAT-Box:

The NAT-Box handles all client connections and communications, including all of the algorithms and implementations required. The NAT-Box is constantly listening for attempted connections from internal and external clients and will execute the correct protocols for the connections of these clients until a predetermined limit has been reached.

Every client is added to the NAT-Table, with External clients being mapped to their own IP address and internal clients having their internal IP address (obtained via DHCP) mapped to their externally valid IP address. For every client that connects, a ClientConnection object is created. This object handles the receiving, parsing, editing, and routing of all packets that are sent through the NAT-Box.

Routing rules:

Internal → Internal: Packet is forwarded without changing its data.

Internal → External: The packet header is modified, and an entry is written into the NAT table, or refreshed if it already exists, that binds the source IP to the destination address.

External → Internal: The packet is routed according to the entry in the NAT table, if there is no corresponding entry in the table for the source, the packet is dropped and an error packet is returned.

External → External: Packets are dropped as they should be routed by external networks.

EXPERIMENTS:

The experiments that I performed were used to test whether the packets were being routed and edited correctly. I also experimented with different manners of allocating the IP addresses in the DHCP implementation.

Routing tests:

In these tests, I sent multiple packets through every possible permutation of Internal and External clients and compared the resultant packets with what was expected.

Internal → Internal:

Expectation:

The packet does not get changed and arrives successfully at the destination. The destination receives the packet correctly and the source receives an ACK that the packet was forwarded correctly.

Result:

As expected.

Internal → External:

Expectation:

The Packet is edited to reflect the internal client's externally valid IP. The destination receives the packet correctly and the source receives an ACK that the packet was forwarded correctly.

Result:

As expected.

External → Internal:

Expectation:

The Packet is edited to reflect the internal client's externally valid IP. The destination receives the packet correctly and the source receives an ACK that the packet was forwarded correctly.

Result:

As expected.

External → External:

Expectation:

The packet is dropped, and an error packet is returned to the source.

Result:

As expected.

External → Unknown External:

Expectation:

The packet is dropped, and an error packet is returned to the source.

Result:

As expected.

External → Un-allocated Internal:

Expectation:

The packet is dropped, and an error packet is returned to the source.

Result:

As expected.

Internal → Un-allocated Internal:

Expectation:

The packet is dropped, and an error packet is returned to the source.

Result:

As expected.

Conclusion from experiments:

From my testing and experimentation, I can conclude that my implementation behaves as expected in all situations. This shows that all my routing rules have been implemented correctly to my understanding.

ISSUES ENCOUNTERED:

Most of the time spent on this assignment was researching the theory and piecing together how to apply the theory into an implementation. This was tedious and took a lot of trial and error.

DESIGN:

Algorithms:
DHCP

Dynamic Host Configuration Protocol (DHCP) is a client server protocol to dynamically allocate IP addresses to internal clients in a local network. It enables the NAT protocol and allows computers to be added and removed from networks with minimal manual configuration to be done. In my implementation, when an internal client connects to the NAT box, it sends a discover message to all nodes in the network using DatagramSockets and DatagramPackets. When the NAT box notices the discover message, it reads the client's IP that was sent along with the discover message and sends an offer message back. The offer message contains the first unique IP that is available in the NAT table. The client then sends an accept message back saying that it would like that IP. The NAT box adds this mapping to the NAT table and then sends a acknowledge message back to the client. The client can then connect to the external internet using this allocated IP.

ICMP

Internet Control Message Protocol (ICMP) is an error-reporting protocol network devices like routers use to generate error messages to the source IP address when problems prevent delivery of IP packets. ICMP creates and sends messages to the source IP address indicating that a gateway to the Internet that a router, service, or host cannot be reached for packet delivery. In my implementation, an extra error payload is added to the failed packet before it is returned.

Data Structures:
For the NAT-table and DHCP pool, linked lists of NATtableEntry objects were used. These objects contain an externally valid address, and an internal IP. Each client connection to the NAT-Box is mapped to an entry in the NAT-Table but is represented by a ClientConnection object. This object handles the routing functionality for that specific client. The packets are byte array headers, that can be easily sent through sockets.

COMPILATION:

Either type “javac NAT/*.java” and press enter, or type “make” and press enter.

EXECUTION:

To run with pre-set configurations (pre-set args) from .sh files:

- To run the NATBox with 600000ms table refresh time, type './NATBox.sh' in the terminal and press enter.

- To run a new client on localhost with port 6969, type './clientInt.sh' for an Internal client or './clientExt.sh for an External client.

To run with manual configurations (custom args):

- To run the NATBox, type the following:
  - java NAT/NATBox <**Table_refresh_time**>
- To run a new client, type the following:
  - java NAT/Client <**host**> 6969 <**int/ext**>

(6969 is the port and int/ext is for an internal client or external client)

To send messages between clients, type the following into console:

- send <**IP_of_destination**>

LIBRARIES USED:

No other libraries other than the built-in Java libraries were used.

- import java.io.BufferedInputStream
- import java.io.DataInputStream
- import java.io.DataOutputStream
- import java.net.Socket
- import java.io.IOException
- import java.io.PrintStream
- import java.net.DatagramPacket
- import java.net.DatagramSocket
- import java.net.InetAddress
- import java.util.Arrays
- import java.io.PrintStream
- import java.net.ServerSocket
- import java.util.LinkedList
- import java.util.Timer
- import java.util.TimerTask
- import java.util.Random

CONCLUSION

The researching and implementing of this project helped me understand what NAT, IP, DHCP, ICMP are, what they are used for and how they work exactly. It was much better practically implementing the project rather than all the research and theory. I believe that I have implemented all the necessary features according to the specification (besides port forwarding and Ethernet frames in the packet construction). All my testing confirms that I have implemented it correctly. I am very pleased with how my project turned out.