Project 3: Network Address Translation

Department of Mathematical Sciences
Computer Science Division
University of Stellenbosch
7600 Stellenbosch

July 2016

1 Introduction

Network address translation (NAT) is a fundamental component used in the Internet today. It was implemented because of the limited size of the IPv4 address space. One of the key features envisioned by the designers of the Internet was that every device would have a *unique* address that identified it on the Internet. As more devices became part of the Internet it became apparent that the current address pool was a restricting resource. The solution was IPv6, which has a near inexhaustible supply of IP addresses.

Before IPv6 was released a stopgap was proposed that would solve the impending address scarcity, namely Network Address Translation. NAT allows local nodes to form a network using non-unique IP addresses and communicate internally without packet modification. When a internal node wants to send a packet to a node outside its network, the packet needs to go through a NAT routine. In the most basic situation the NAT-box would be allocated a single IP address by some external authority (such as an ISP). The NAT routine takes the internal packet and modifies its header so that at appears to originate from the NAT-box (using the allocated external address). Some additional housekeeping is done to ensure that replies and requests are matched up.

There is a trade off for using NAT. A NAT-box can significantly impact the quality of Internet connectivity and it breaks the originally envisioned model of IP end-to-end connectivity across the Internet. Additionally, it becomes difficult for hosts in a NAT protected intranet to accept incoming connections.

2 Preparation

Before starting to program a NAT-box it is advisable to study the following Request For Comments (RFCs). These give an overview of how a NAT-box operates and give the implementations details of several various types of NAT:

IP NAT Terminology and Considerations RFC2663.

Traditional NAT RFC3022.

It is also important to have a solid understanding of the following concepts:

Encapsulation
: Ethernet frames, IP packets, TCP segments, UDP packets and how they interact.

Headers and Payload
: Construction, padding, flags.

DHCP
: Dynamically allocate addresses to internal clients (refer to RFC1531 and RFC2131 for more information).

Port Forwarding
Allow external clients to initiate a session given an address and a correctly bound port number (refer to wikipedia for more information).

ICMP (Internet Control Message Protocol)
A message protocol used by routers to communicate error messages to clients when packets could not be delivered. ICMP also defines Echo request and reply messages. See wikipedia and RFC792 for more details.

NAT ICMP behaviour
The behaviour of NAT differs for TU (TCP/UDP) and ICMP packets. See RFC5508 its behaviour regarding ICMP.

3 Implementation Details

You will be required to implement a simulation router that (minimally) has NAT functionality. The router will process *paquets*[1] that are received from clients, where the clients are either marked as `internal` (hosts in the local network) or `external` (hosts marked as belonging to an external network)

When a client connects to the server it indicates whether it is an internal or external client and its IP address (obtained via DHCP). Even though this is a simulated NAT environment (so a rogue *paquet* won't cause any problems), ensure that the internal IP addresses are within the correct ranges, as allocated in RFC1918 section 3.

The clients then send simple *paquets* to each other (possibly ICMP echo request/response packets, or PING packets see), and the server processes these *paquets* depending on one of the following scenarios:

Internal → Internal
The *paquet* is forwarded without changing its data.

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

External → Internal
The *paquet* is routed according to the entry in the NAT table, if there is no corresponding entry in the table for the source, the *paquet* is dropped and an error *paquet* is returned. (Unless something like port forwarding has been implemented.)

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

Additional features that should be implemented are:

NAT Table
The address translation table should refresh dynamically, see the specification for further details.

DHCP
Internal clients should automatically request and be assigned an address by your NAT server, refer to the project specification section for more information.

Any language may be used to implement this project, but it is advised that you use either ANSI C, Java or Python.

[1] The term *paquet* differentiates between packets sent in a real network, and the simulated packets (*paquets*) being discussed.

4 Project Specification

You will need to implement both a simple client and the NAT-box/router for this project. The clients must exchange simple *structured paquets* that reflect the packet formats currently found in the Internet. A basic NAT-box is required and more marks will be awarded for different (improved) variations that are implemented.

4.1 The NAT-box

The NAT-box must have the following properties:

1. The NAT-box must be given a unique hardware (MAC) address.
2. The NAT-box must be given a unique IP address.
3. The NAT-box's MAC address and IP address must be communicated to clients (A minimal ARP implementation might work here).
4. Multiple clients must be able to be connected to a single NAT-box.
5. Internal clients must be assigned unique IP addresses by the NAT-box by using a minimal DHCP server implementation.
6. If an internal client disconnects its IP address must be released into the pool of available IP addresses.
7. The NAT-box must be able to process and modify the packets that it receives.
8. A NAT table (address translation table) must be present in your implementation.
9. The NAT table must be refreshed dynamically (removing unused entries). This can be achieved through a threaded timer and entry timestamps (for example every 10 minutes update the table).
10. The NAT table refresh time must be configurable, as this facilitates testing.
11. No GUI is required for the NAT-box.
12. Certain status *paquets* must be printed out, such as when a *paquet* has been routed, or could not be routed.
13. Clients are only able to communicate according to the rules of the chosen NAT-box implementation (refer to RFC 3022).

4.2 The Client

The clients that must be implemented must have the following properties:

1. Each client must either be designated as an Internal or External client, as discussed above.
2. The clients' IP addresses should reflect their status as either Internal or External clients.
3. Internal clients should use a minimal DHCP implementation to request an IP address.
4. Every client must have a unique (simulated) MAC address
5. No GUI is required for the client, however it should have the ability to send a simple *paquet* to a targeted client.
6. The client must provide a reply for every *paquet* received (This is not the case with ICMP error *paquets*). This may take the form of an ACK or a similar type of packet.

Your simulator must minimally support the following protocols and *paquets*:

1. ICMP *paquet* forwarding (error and echo/response *paquets*)
2. TU (TCP / UDP) *paquet* forwarding
3. A minimal DHCP implementation

5 Dates

- **Project starts:** 17 August 2016 (14:00).
- **Project deadline:** 14 September 2016 (13:00).

The report, Makefile and source codes must be submitted as a single tar or zip file whose name has the following form: `GROUP_NUMBER.tar.gz` where `GROUP` is your group name and `NUMBER` is the project number. Your project with the report must be emailed to `rw354demi@cs.sun.ac.za` before the deadline.

6 Helpful Hints

- Start early, do not leave this project to the last few days and think you will finish on time.
- Look at the marking scheme and at what needs to be done to help you gauge your progress.
- Do not think you are almost done after you have successfully had your client connect to your server.
- Ensure that you have read and understood the RFCs before choosing a NAT model to implement.
- It may be useful to construct packets using byte arrays rather than objects.
- The clients are simple and should not take long, the NAT-box is the focus of the project and thus contains the concentration of the marks.

6.1 Writing your report

Please refer to the report guidelines available on the course website.

7 Marking Scheme

Packet Construction	**5**
- *Ethernet*	2
- *IP*	2
- *Payload*	1
Simulator Design	**5**
- *DHCP*	2
- *IP Addresses and Hardware Addresses*	2
- *Address Distribution*	1
Correct Routing	**10**
- *Changing Packet Information*	5
- *Internal Routing*	1
- *Internal / External Routing*	4
Sending/Receiving Messages	**2**
House keeping and Style	**3**
Design and Stability	**10**
- *Correct use of data structures*	2
- *Threading*	3
- *Simultaneous client communications*	3
- *Multiple connections/disconnections*	2
Report	**15**
- *Problem statement*	
- *Design and Implementation*	
- *Selection of experiments discussed*	
- *Execution instructions*	
- *Conclusions drawn from the results*	
- *Language, spelling and grammar*	
TOTAL	**50**