# NAT and Tunnels

Alessandro Barenghi

Dipartimento di Elettronica e Informazione
Politecnico di Milano

*barenghi - at - elet.polimi.it*

May 25, 2011

# Recap

**By now , you should be familiar with...**

- System administration skills on the local host
- Network and system programming
- Basic structure of NetFilter/IpTables

# Lesson contents

## Overview

- Network Address Translation
- Port address Translation
- IP-over-IP and GRE tunnels

# Up to now

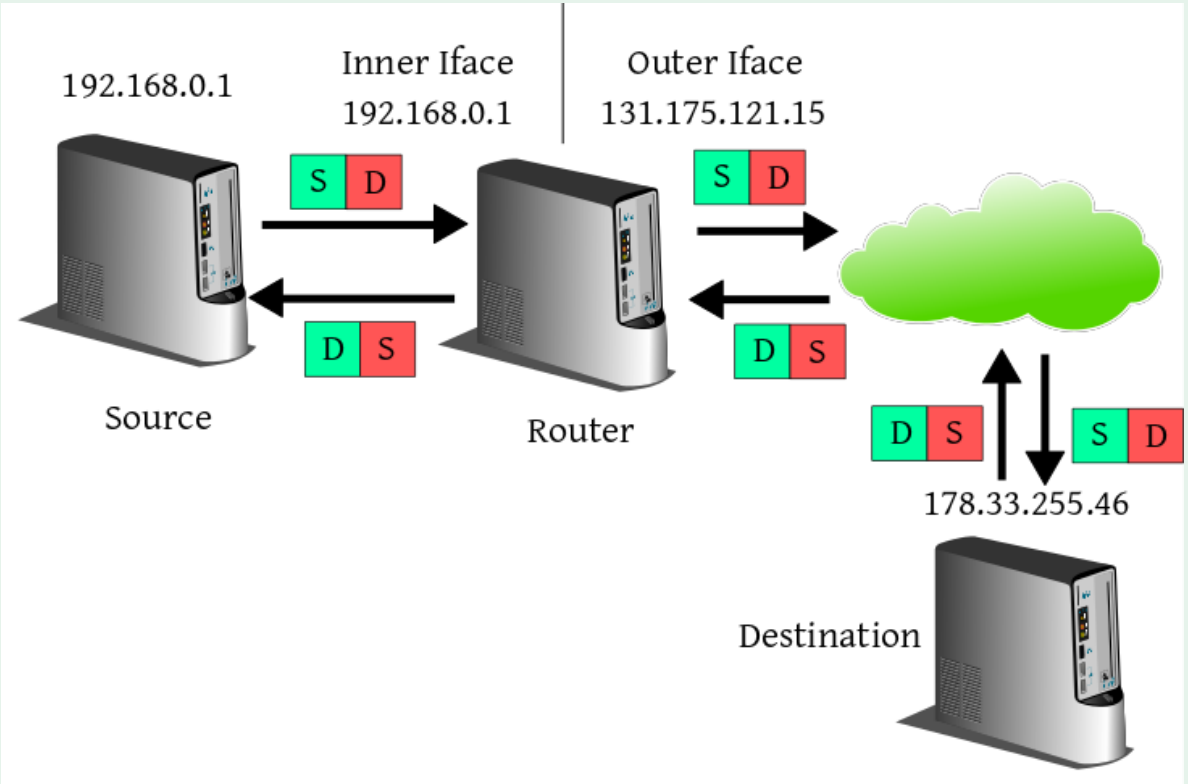
## The end-to-end transparency

- Up to now, we assumed that the “any-to-any” principle of IP was respected
- All the IP addresses were assumed to be reachable via a number of routing hops
- A single packet, once built, was always delivered as-is , without any changes
- The end-to-end transparency of a communication was never questioned

# Up To Now

## Why?

Regular network communication



# Address Translation

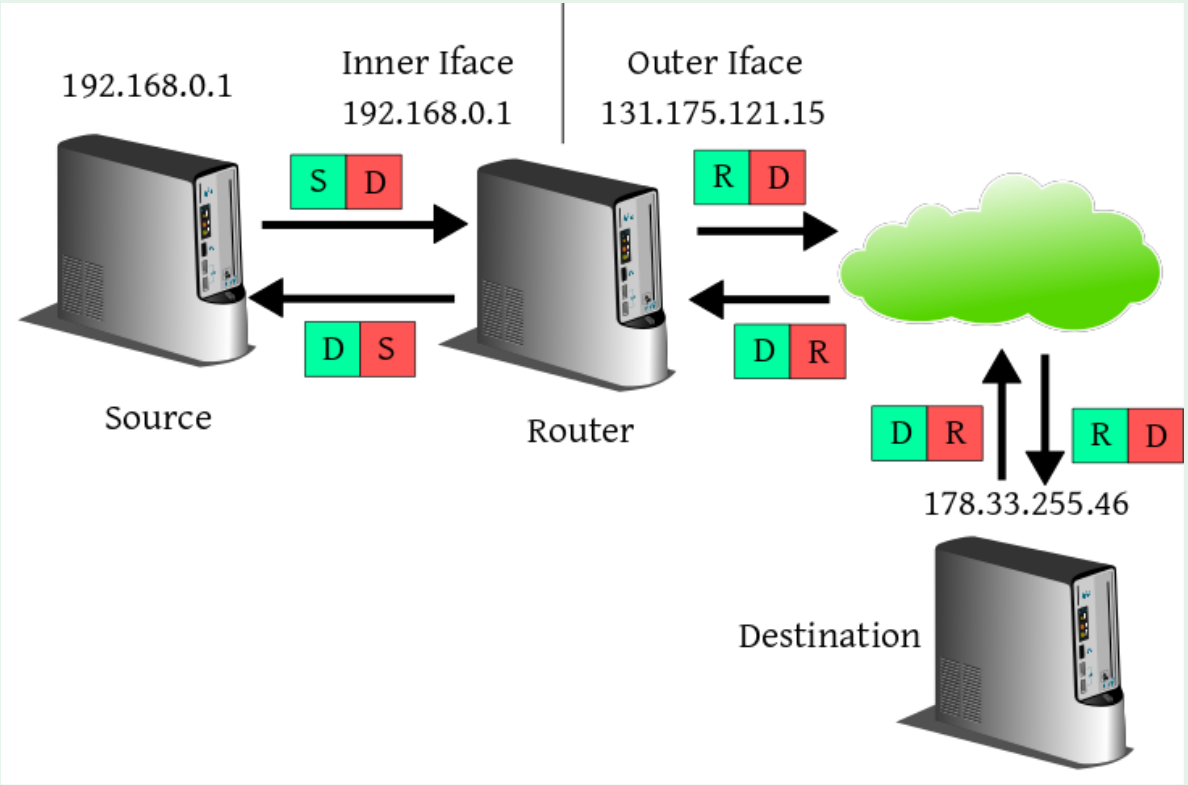
## Source NAT

- Although it raises some issues, it may be needed to mask a number of hosts under a single one
- Common when a LAN needs to access a public network but only one public IP address has been bought from IANA or the ISP
- Useful to “concentrate” accesses behind a single IP
- The best candidate to perform the packet mangling is the router

# Address Translation

## SNAT

The general structure of a source network address translation based architecture



# Address Translation

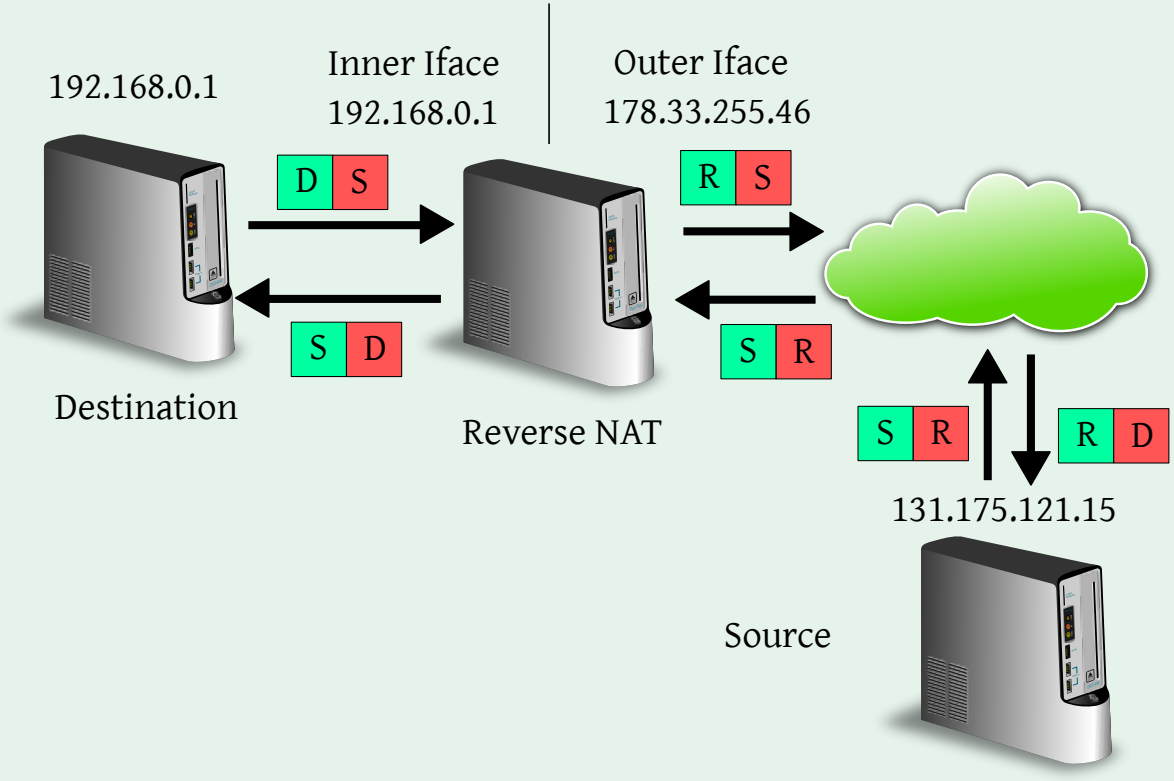
## Destination NAT

- Symmetrically, it may be helpful to split the network load managed by a server
- This can be done through dynamically modifying the destination of the communication
- The operation must be performed by the alleged target of the communications
- Bonus: it allows to replace machines behind the DNAT without interrupting a service

# Address Translation

## DNAT

The general structure of a destination network address translation based architecture



# NAT Features/Issues

## Opacity

- Once a NAT strategy is actuated, the IP domains on the two sides of the NAT are effectively split
- It is possible to mitigate the IPv4 address exhaustion
- The hosts behind a NAT are perfectly opaque[a]
- The host performing NAT must actively alter the packets, no end-to-end transparency

[a] almost perfectly actually

# NAT Strategy

## Address Translation Table

- In order to perform a correct NAT, a table containing all the connections must be kept
- Every time a new connection is requested, a line is added to the table
- The address translation mechanism will consistently map back the returning packets to the correct host
- Once a connection is torn down, the line in the mapping is removed

# NAT Issues

## Potholes

- Stateless protocols do not have a proper session
- The number of connections which can be opened between two hosts is lower than the one between $n$ and one host
- Some upper level protocols contain redundant information on the network layer[a] which must be mangled too

[a] yes, this is a terrible practice

# NAPT

## Potholes

- The most straightfoward extension of the NAT mechanism involves also the mangling of the 4th level datagram
- The Network Address and Port Translation is employed when a protocol needs to communicate exactly on a port
- Destination NAPT is the most common form, in order to split the incoming communications on different servers according to the service needed
- Source NAPT is rather uncommon, as usually the source port is an ephemeral port with no particular meaning

# NAT: How?

## Overview

- The Netfilter infrastructure allows natively to perform [S|D]NA(P)T on all the connections
- The rules specifiying the NAT policies are inserted in the `nat` tables present on the `PREROUTING` and `POSTROUTING` hooks adding `-t nat` to the rule
- There is also a `nat` table in the `OUTPUT` hook, if you want to perform pre-output routing
- The connection state tables are automatically kept by the system

# Structure

## Overview

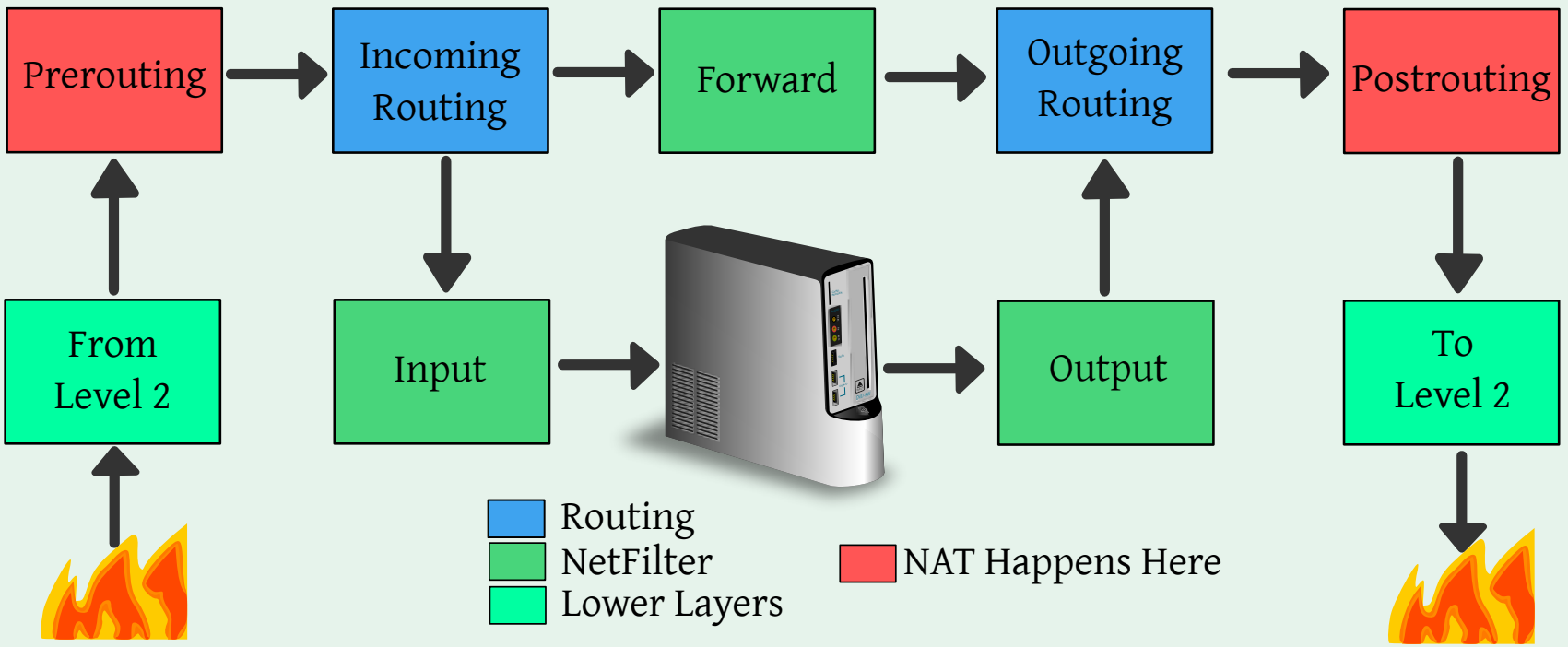



Figure: NAT communication and tables

# Source NAT

## Overview

- Source NAT is performed in the `POSTROUTING` hook, when the packet is about to leave
- The corresponding translation for the returning packet is automatically managed
- A simple `-t nat -A POSTROUTING -j SNAT --to <address>` rule sets all the packets matching it to be masked
- The output interface specifying option `-o` and comes in handy to specify which connection to mask
- The special target `-j MASQUERADE` instructs Netfilter to choose automatically the outgoing address according to the egress interface

# Destination NAT

## Overview

- Destination NAT is performed (symmetrically) in the `PREROUTING` hook, before anything is done to the packets
- The bidirectional communication of an established connection is also automatically managed
- The `-t nat -A PREROUTING -j DNAT --to-destination <address>` rule indicates the address where the packet should be redirected
- The input interface specifying option `-i` allows rough balancing in multi-interface routers
- Obviously, no automatic destination selection can be performed here

# Destination [S|D]NAPT

## Overview

- Both the Source and Destination NAT in NetFilter can be performed taking also into account ports
- The destination port of a NAT retargeted packet can simply be specified adding `:port` to the translated address
- A port range for both destination and source can be specified as `:port-port`
- By default the ports are mapped 1:1 on the range
- For privacy reasons, it is also possible to specify a `--random` option forcing the mapping to a random port for each connection

# Caveats

## Overview

- Do not set the policies of the `PREROUTING` and `POSTROUTING` hooks to anything but `ACCEPT`, filter later
- Remember that the packets will still pass through the `FORWARD` chain most of the times
- Take care that the DNAT-ed packets will flow through with a different destination IP and thus should be filtered accordingly
-

# Bidirectional traffic

## Overview

- NATs have the great advantage of being transparent to either the source or the destinations of the connection
- However, the main issue with NATs is that only one of the sides can start a connection.
- If two separate networks are required to communicate, regardless of the fact there are not enough public IP for all hosts tunnels can be used
- The general idea of a tunnel is to employ a connection as a virtual Level 2 link

# IP-over-IP tunnels

## Overview

- The simplest strategy is to encapsulate an IP datagram into another IP datagram
- The outer IP communication acts as the L2 link, while the actual IP communication is the one going on inside
- This methodology is described in RFC2003 and was the first proposal for tunnels
- It has a couple of issues, among which the bad handling of multicast connections

# GRE tunnels

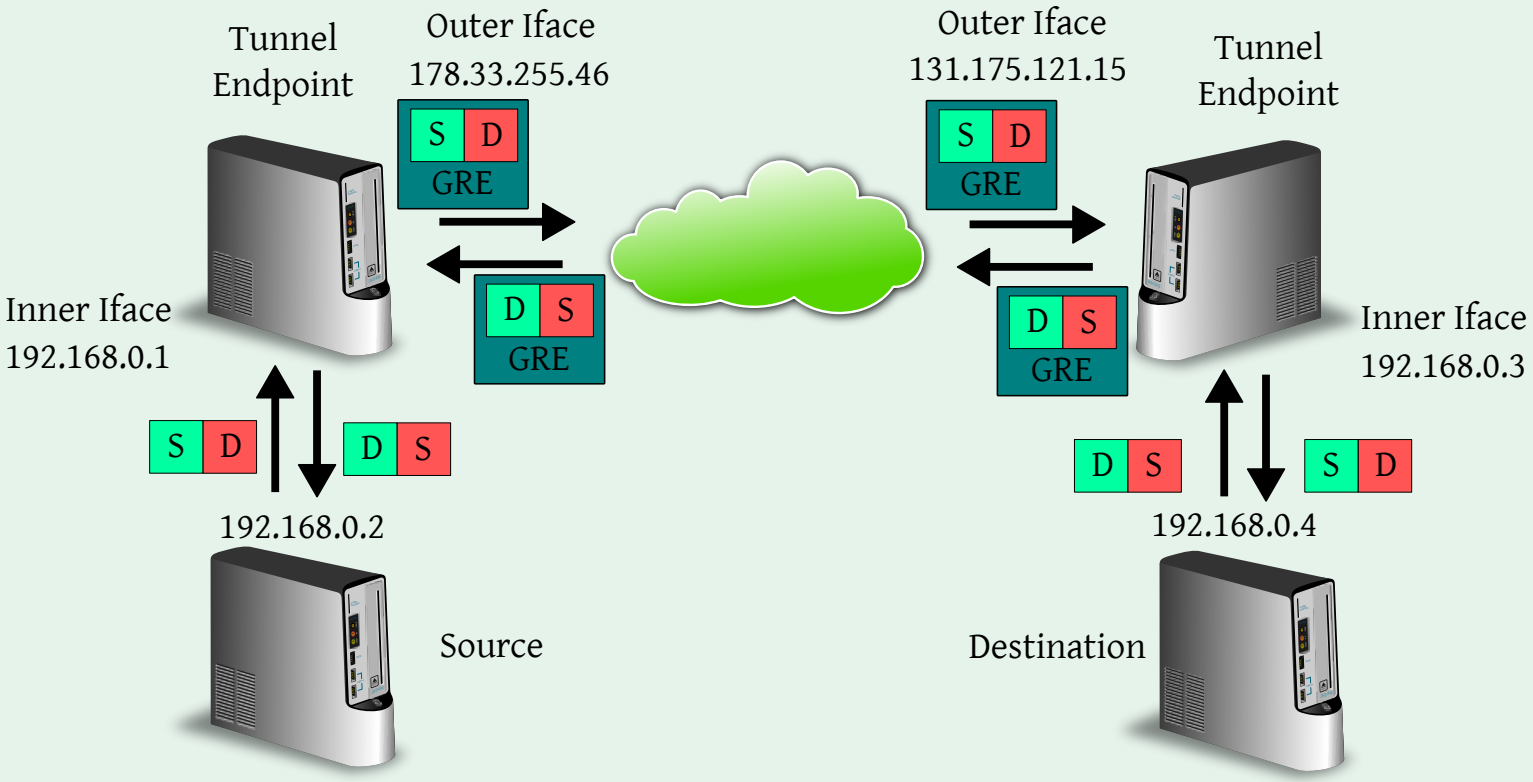
## Overview

- The Generic Routing Encapsulation protocol (GRE) was developed by Cisco as a replacement for IP-over-IP
- The protocol supports multicast messages, IPv4 and IPv6 payloads natively
- It has become the de-facto standard in tunneling
- It handles tunnel loops (the destination address of a tunneled datagram is through the tunnel itself)
- Coupled with IPSec , it forms the basic structure for the large majority of the VPNs around

# GRE tunnel

## DNAT

### The general structure of typical GRE based tunnel



# iproute2 suite GRE tunnels

## How-To

- The iproute2 suite provides also full support for tunnels
- Tunnels are treated as virtual, point-to-point, interfaces on which the host communicates
- Setting up a tunnel is as simple as : `ip tunnel add <interface> mode gre remote <endpoint> local <endpoint>`
- The same command, with swapped addresses should be employed on the other endpoint
- After the tunnel has been pulled up via `ip link <interface> up`, it is possible to route through it as a regular interface