Advanced Networking: Network Address Translation (NAT)

Renato Lo Cigno
Renato.LoCigno@disi.unitn.it
Network Address Translation

- Originally (RFC 1631 - obsolete) a “simple” method for connecting a private network to the public Internet
  - Also called *network or IP masquerading*
- Now (Traditional NAT, RFC 3022) includes also port translation and is more correctly called NAPT: Network Address and Port Translation
  - Payload (application) independent and almost transparent
- NAT evolved and evolves highly intertwined with Firewalls, Routing (a NAT is always also a Router), Traffic Monitoring, and Proxy
  - Often NAT techniques and implementations go beyond RFCs … which follow up
- NAT and NAT traversal evolves in parallel and are intertwined
Basic operation of NAT

- NAT device must map addresses
- A one-to-one translation brings little advantages
  - Not many public IP “spared” specially if computers are always on
One use of Basic NAT

- **Supporting migration between network service providers**
- **Scenario:** IP addresses are obtained from the service provider. Changing the service provider requires changing all IP addresses in the network.
  - **NAT solution:**
    - Assign private addresses to the hosts of the corporate network
    - NAT device has static address translation entries which bind the private address of a host to the public address
    - Migration to a new network service provider merely requires an update of the NAT device. The migration is not noticeable to the hosts on the network
- **The same can be done with properly configured DHCP: obsolete use!!**
IP masquerading or NAPT

- A single (or few) public IP address is mapped to multiple hosts in a private network
  - Assign private addresses to the hosts of the corporate network
  - NAT device modifies the port numbers for outgoing traffic
  - Ports should be translated as well

<table>
<thead>
<tr>
<th>Private Address</th>
<th>Public Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.1.2/2001</td>
<td>128.143.71.21/2100</td>
</tr>
<tr>
<td>10.0.1.3/3020</td>
<td>128.143.71.21/4444</td>
</tr>
</tbody>
</table>
Load balancing of servers

• Balance the load on a set of identical servers, which are accessible from a single IP address
  – servers are assigned private addresses
  – NAT device is a front-end for requests to the server from the public network
  – The NAT device changes the destination IP address of arriving packets to one of the private addresses for a server

• Many strategies for assignment
  – Simple round-robin
  – Weighted round robin
  – With feedback from servers on the actual load
Load balancing of servers

<table>
<thead>
<tr>
<th>Private network</th>
<th>Outside network</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source</strong></td>
<td><strong>Public Address</strong></td>
</tr>
<tr>
<td>10.0.1.2</td>
<td>128.143.71.21</td>
</tr>
<tr>
<td>10.0.1.3</td>
<td>128.143.71.21</td>
</tr>
<tr>
<td>10.0.1.4</td>
<td>128.143.71.21</td>
</tr>
</tbody>
</table>
Concerns about NAT

• Changing the IP address requires that NAT boxes recalculate the IP header checksum
• Modifying port number requires that NAT boxes recalculate TCP checksum
• Additional care is needed if a fragmented datagram reaches a NAT device to avoid inconsistent assignments to pieces of the same packet
• End-to-end connectivity:
  – NAT destroys universal end-to-end reachability of hosts on the Internet
  – A host in the public Internet often cannot initiate communication to a host in a private network
  – The problem is worse, when two hosts that are in a private network need to communicate with each other
Further concerns about NAT

- Applications that carry IP addresses in the payload generally do not work across a NAT
- Some NAT boxes inspect the payload of widely used application layer protocols and, if an IP address is detected in the payload, translate these addresses too
- Typical example is ftp
- Further problems with sftp because the payload is encrypted
NAT and FTP

- Normal FTP operation

FTP client

H1

public address: 128.143.72.21

---

FTP server

H2

public address: 128.195.4.120

PORT 128.143.72.21/1027

200 PORT command successful

RETR myfile

150 Opening data connection

establish data connection
NAT and FTP

- NAT device with FTP support
NAT and FTP

• FTP in passive mode and NAT
Configuring NAT in Linux

- Linux uses the Netfilter/iptables package to add filtering rules to the IP module
Iptable based NAT: examples

• First example:
  ```
  iptables -t nat -A POSTROUTING -s 10.0.1.2
  -j SNAT --to-source 128.143.71.21
  ```

• Pooling of IP addresses:
  ```
  iptables -t nat -A POSTROUTING -s 10.0.1.0/24
  -j SNAT --to-source 128.128.71.0-128.143.71.30
  ```

• ISP migration:
  ```
  iptables -t nat -R POSTROUTING -s 10.0.1.0/24
  -j SNAT --to-source 128.195.4.0-128.195.4.254
  ```

• IP masquerading:
  ```
  iptables -t nat -A POSTROUTING -s 10.0.1.0/24
  -o eth1 -j MASQUERADE
  ```

• Load balancing:
  ```
  iptables -t nat -A PREROUTING -i eth1 -j DNAT --to-destination 10.0.1.2-10.0.1.4
  ```
NAT traversal and classification

• Classification of NAT techniques come with methods to traverse NAT boxes
• STUN (Simple Traversal Utility for NAT – RFC3489)
• Same acronym modified to Session Traversal Utilities for NAT in RFC5389
• Universally supports traversal for UDP only
  – RFC5389 supports (with some limits) also TCP and TLC
• STUN is a client-server protocol, with the server on the public side
• STUN servers are identifies via srv records of DNS
  – stun for UDP
  – stuns for TCP/TLC
Full-cone NAT

- Known as one-to-one NAT
- Does a semi-static mapping \((i\text{Addr}:i\text{Port}) \leftrightarrow (e\text{Addr}:e\text{Port})\)
  - any packets from \(i\text{Addr}:i\text{Port}\) is sent through \(e\text{Addr}:e\text{Port}\)
- External hosts can send packets to \(i\text{Addr}:i\text{Port}\) by sending packets to \(e\text{Addr}:e\text{Port}\)
Address-restricted-cone NAT

• Does a semi-static mapping \((i\text{Addr}:i\text{Port}) \leftrightarrow (e\text{Addr}:e\text{Port})\)
  – any packets from \(i\text{Addr}:i\text{Port}\) is sent through \(e\text{Addr}:e\text{Port}\)
• An external host \((h\text{Addr}:\text{any})\) can send packets to \(i\text{Addr}:i\text{Port}\) by sending packets to \(e\text{Addr}:e\text{Port}\) only if \(i\text{Addr}:i\text{Port}\) has previously sent a packet to \(h\text{Addr}:\text{any}\)
  – The sending (server) port number doesn't matter
Port-restricted-cone NAT

- Like an address restricted cone NAT, but the restriction includes port numbers
- Does a semi-static mapping \((i\text{Addr}:i\text{Port}) \leftrightarrow (e\text{Addr}:e\text{Port})\)
  - any packets from \(i\text{Addr}:i\text{Port}\) is sent through \(e\text{Addr}:e\text{Port}\)
- An external host \((h\text{Addr}:h\text{Port})\) can send packets to \(i\text{Addr}:i\text{Port}\) by sending packets to \(e\text{Addr}:e\text{Port}\) only if \(i\text{Addr}:i\text{Port}\) has previously sent a packet to \(h\text{Addr}:h\text{Port}\)
Symmetric NAT

- Packets from (iAddr:iPort) to different (hAddr:hPort) are mapped to different (eAddr:ePort)
  - A host appears with different (eAddr:ePort) to different hosts!!
- Only an external host that receives a packet from an internal host can send a packet back
  - Strong firewalling, very difficult to traverse
NAT Problems (reprise)

- NAT means a 'table' binding private and public addresses and ports
- 'Bindings' can only be initiated by outgoing traffic
- NAT breaks end-to-end semantics

Methods of solving the ‘NAT Problem’

Some proposals for solving NAT traversal are:

- Simple Traversal of UDP Through Network Address Translation devices (STUN)
- Traversal Using Relay NAT (TURN)
- Universal Plug and Play (UPnP)
- Tunnel Techniques
STUN

• Lightweight protocol that allows applications to discover the presence and types of NATs and firewalls between them and the public Internet
• Provides the ability for applications to determine the public Internet Protocol (IP) addresses allocated to them by the NAT
• STUN works with many existing NATs, and does not require any special behavior from them
• A STUN server in the public address space informs STUN-enabled clients of the Public NAT IP address and port being used for that particular session
STUN

1: query to know eA:eP

2 – communicate eA:eP to potential clients (eg, SIP, peers, ...)

Client  
iA:iP 

NAT  
eA:eP 

STUN Server

Server A
Operation of STUN

- STUN identifies eA;eP by inspecting STUN messages that arrive at the STUN server
- STUN-enabled hosts send an exploratory message to the external STUN server to determine the transmit and receive ports to use
- The STUN server examines the incoming message and informs the client which public IP address and ports were used by the NAT
- These are communicated to e.g.
  - SIP proxies/buddies in the call establishment messages
TURN - Traversal Using Relay NAT

- TURN relies on a “counter” middlebox that is inserted in the communication path.
- A TURN server is located:
  - in the campus DMZ
  - in the Service Provider network
- A TURN-enabled client sends initial messages to the TURN server.
- The TURN server will forward the traffic reverting the NAT operation.
- This information is used e.g.:
  - in the SIP call establishment messages and for subsequent media streams
  - in P2P gossipping messages
- Works with symmetric NAT:
  - No change in the destination address seen by the NAT
  - Heavy protocol!!
  - Can be used as a second resort
  - See also ICE (Interactive Connectivity Establishment)
TURN setup
Tunnel Techniques

[Diagram showing the process of tunneling in a network, with steps labeled: 1 - Signalling, 2 - Media. The diagram includes entities such as Corporate Clients, Call Agent, SIP tunnel, Tunnel Origin, Tunnel Termination, and Firewall/NAT.]
Tunnel Techniques

- A tunnel can be used to cross any Firewall/NAT
- The tunnel termination can be anywhere
- The Tunnel can be also secure (see IPSec)
- Can even be used to anonymize communications (see Onion Routing and Tor)
- Blocking tunnels is difficult
  - impossible if they are TLS