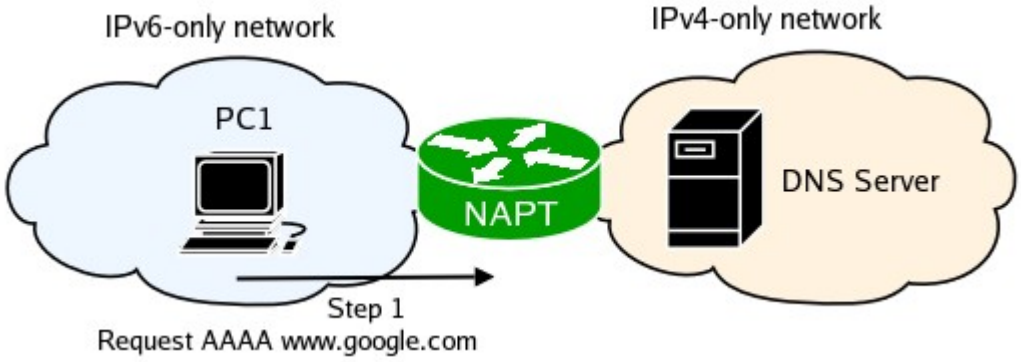
NAT-PT - Documentation

"Soon, we will be entering the true age of broadband networks, a time when electronic products will be linkerd by high-speed, high-capacity networks. With these advances will come IPv6 (Internet Protocol Version 6), which will assign a unique IP address to everything from TVs and PCs to telephone and AV products. Distinguishing individual devices over networks will become possible as a result.", **Sony Annual Report 2001**

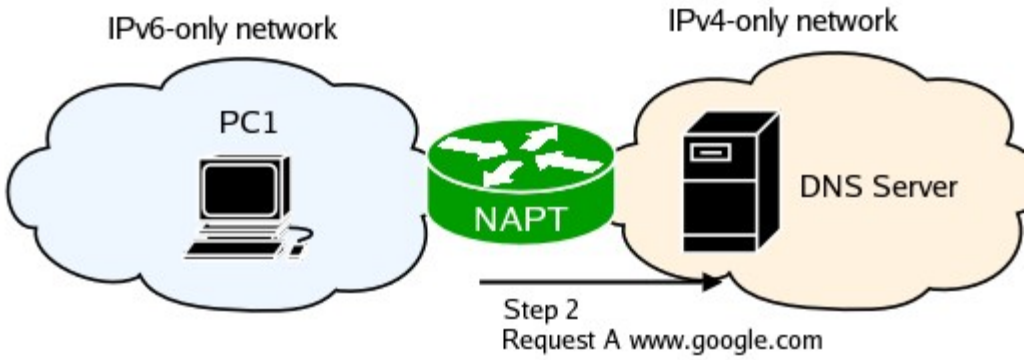
Table of contents

Operations

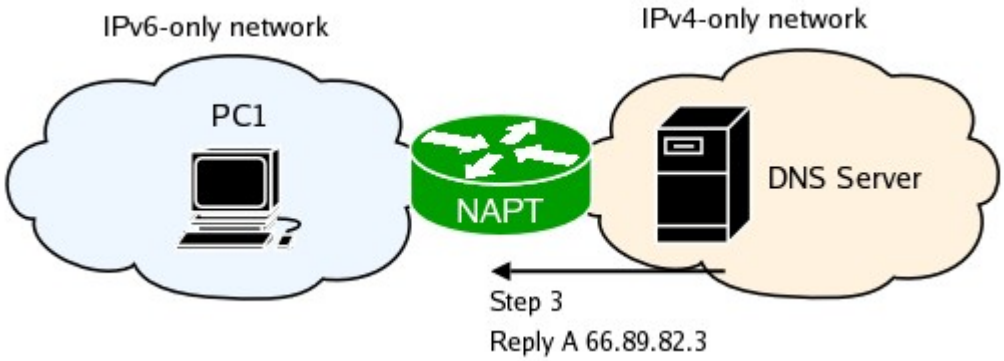
NAT-PT works by capturing, translating and sending packets from the IPv6 to the IPv4 network (and vice versa). The destination IPv4 address for the outdoing packet is determined by the last four bytes of the destination in the IPv6 packet received. Look at the diagram below. A user on PC1, an internal IPv6-only host opens an Internet browser and starts by going to www.google.com. His computer first performs a query to it's DNS server.



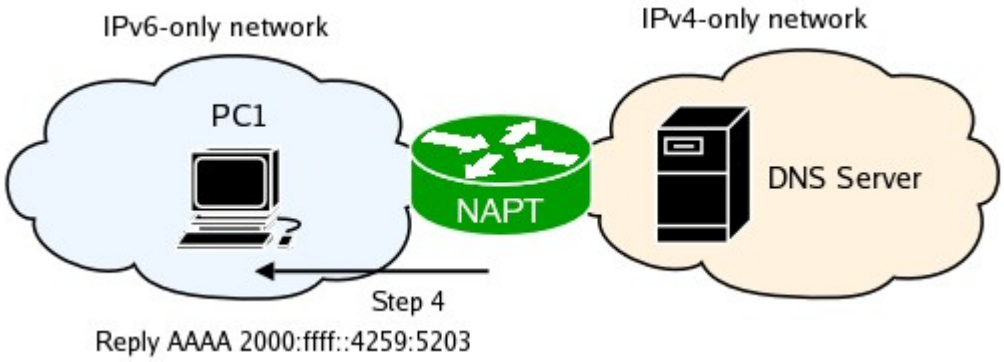
NAT-PT captures the packet and creates a new binding between the IPv6 address of PC1 and one of the IPv4 addresses it has in its pool, and translates the packet. NAT-PT sees a DNS query and translates the AAAA request to an A request. Finally it finds a route for the newly created IPv4 packet and queues it to the appropriate outbound interface.



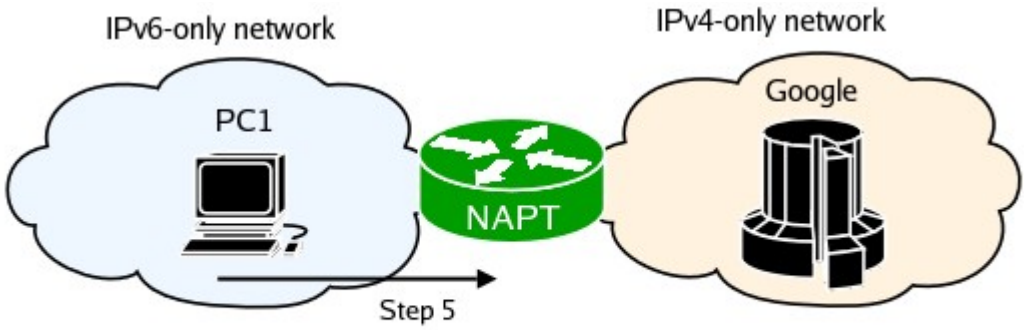
The DNS server receives the query and sends a reply.



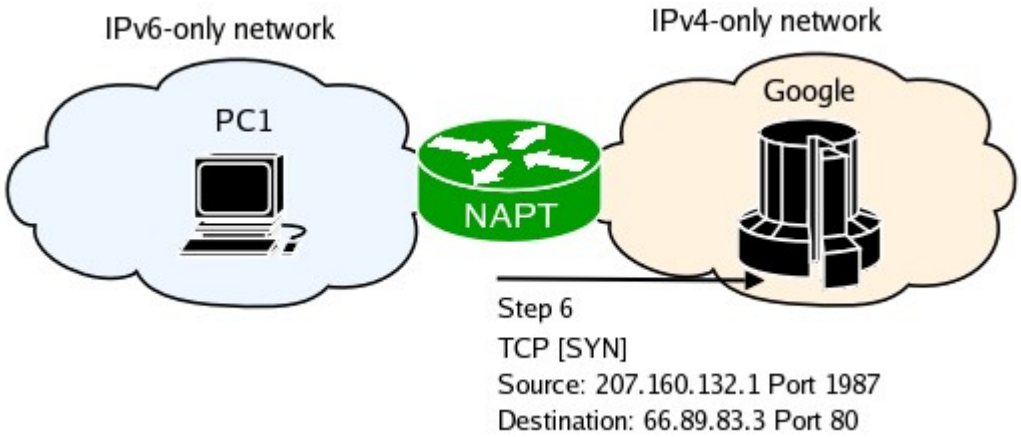
NAT-PT captures the traffic and finds a the IPv6 address for the mapping. It also sees the DNS A resource records and translates them to AAAA records. It does this by adding a special 'NAT-PT prefix' (configured through 'naptd-confmaker', default 2000:ffff::) to the beginning of the IPv4 address. Finally it finds a route for the newly created IPv6 packet and queues it to the appropriate outbound interface.



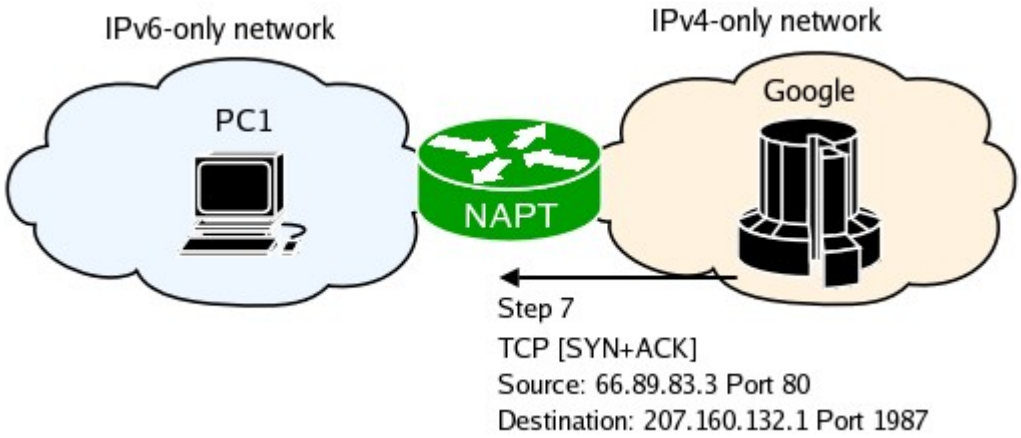
From here PC1 initiates a connection to www.google.com.



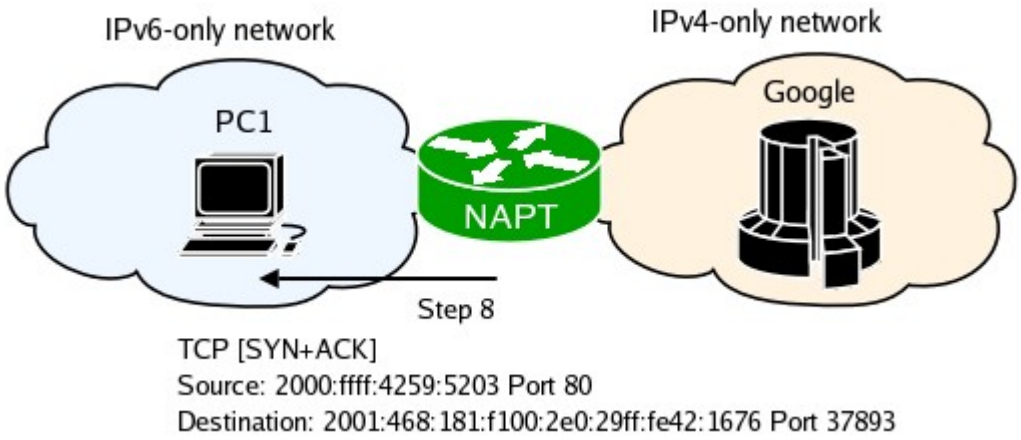
NAT-PT captures the packet and creates a new binding between the IPv6 address of PC1 and one of the IPv4 addresses it has in its pool, translates the packet, finds the route, and ...etc.



Google's webserver replies to the initial SYN packet.



NAT-PT captures the traffic and finds a the IPv6 address for the mapping. Finally it finds a route for the newly created IPv6 packet and queues it to the appropriate outbound interface.



The above summarizes the basics of how IPv6-only and IPv4-only hosts communicate. The rest of the connection occurs in the same way. NAT-PT maintains three "translation" pools, one for TCP, one for UDP and one for ICMP. By default TCP translations timeout after 5 minutes from the time RST or FIN packet was captured, or after 24 hours of innactivity. UDP translations expire after 1 hour, and ICMP after 30 seconds.

Pre-requisites

NAT-PT requires both iptables and ip6tables to work properly. This is caused by the fact that NAT-PT runs in userspace. This has two important side effect:

1. TCP Resets.
After a packet is translated from IPv6 and sent to the IPv4 side by using the IPv4 address of the outbound interface as the global IPv4 address for the translation, the kernel does not note this connection as originated by the router (despite the fact that a packet containing the router's IPv4 address was sent). When a reply is received from the IPv4 side and there is no firewall running on the router, the kernel after seeing that the packet's destination is the router, look's in it's connection table trying to find the application that this packet belongs to. Upon realizing that there is no such application the default, RFC defined behavior is to sent a TCP RST (Reset) packet to the source host. If this happens the connections will be dropped by the remote IPv4 server and all communication between in and the internal IPv6 host will be broken.

2. Route Unreachable/Invalid Routing
The IPv6 hosts see the IPv4 world as a single virtual IPv6 network. It is created by taking the NAT-PT prefix (by default: 2000:ffff::) and setting the IPv4 address of each host as the last 4 octets. This network is purely virtual and doesn't really exist. All the traffic to this network must be routed to the NAT-PT box, which in turns captures it and performs the necessary translations. The fact that we are routing to a non-existant network (excuse me for this oxymoron) causes a small problem. When the router performs routing and doesn't find a route to our virtual network, it attempts to send a ICMP Destination Unreachable (Route Unreachable) packet to the sending host. This must be prevented. If the router has a default route to any IPv6 destination, the packets destinated for our virtual network would be sent to the global IPv6 cloud. This wouldn't disrupt our NAT-PT operations, but sending packets with an non-existant destination into the global IPv6 cloud is highly undesireable and should be prevented whenever possible.

We must compensate for these shortcomings by using a firewall. The first problem can be corrected by using iptables. If you have a default DROP policy on all incoming packets you should be OK. Your iptables rules should have something similar to the example below.

```
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -j DROP
```

To solve the second problem, we must use ip6tables. We can either filter outgoing ICMPv6 Destination Unreachable packets, or filter packets destinated for our virtual IPv6 network in the FORWARD chain. The first mechanism should be used if your router has no default route for IPv6 packets.

```
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type 1 -j DROP
```

If your network does have connectivity to the global IPv6 cloud and therefore your router has a default IPv6 path, you must the second option. The example below uses the default NAT-PT prefix (2000:ffff::), if you are using a non-default prefix, adjust the example to reflect your configuration.

```
ip6tables -A FORWARD -d 2000:ffff:: -j DROP
```

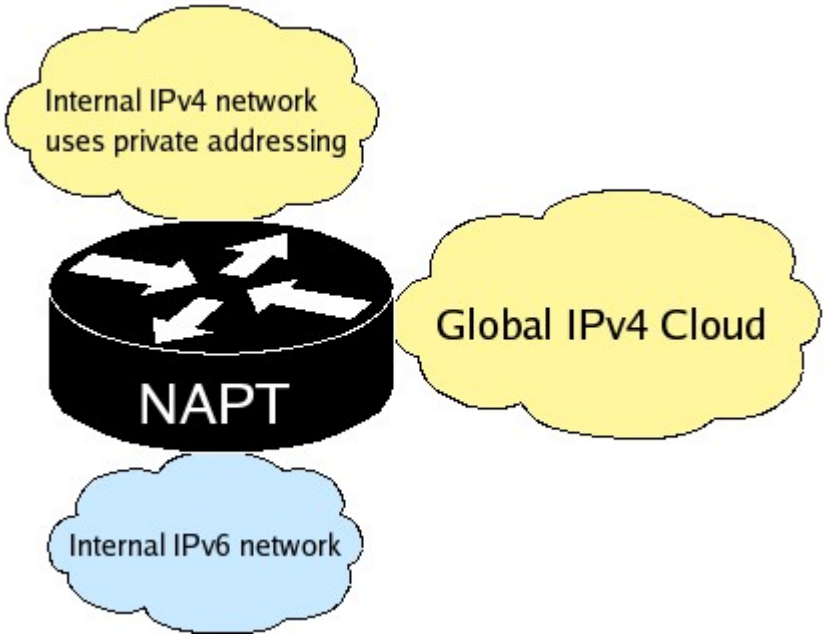
Configuration details

This section explains in detail each step of the NAT-PT configuration process. The NAT-PT translator should be configured using a the program 'naptd-confmaker' included with NAT-PT daemon itself. By default naptd will read the configuration file /etc/naptd.conf, but an alternate configuration file can be supplied by using the -c parameter.

```
naptd -c /usr/local/etc/naptd.conf
```

Let's look at the steps of the configuration process:

1. **Do you want to create a new configuration? [Y/n]**
   This really doesn't require much explanation. If you do want to create a new configuration file we continue, if you don't, the program terminates.
2. **Do you want IPv4 addresses from the outside interfaces to be automatically used as part of the NAT pool? [Y/n]**
   If you answer no skip the next question in this documentation. If you answer yes then NAT-PT will use the IPv4 addresses from the outside interfaces when translating packets. If you have multiple outside interfaces this may be undesirable, as there is no assurance that after a packet is translated it will leave the interface whose IP address it carries. In other words, the translation mechanism is independent from the routing mechanism. In a situation like the one below this could cause a problem. In 99.9% of other cases there should be no problem.



If your network topology resembles the one above, you can still use the IP address of the outside interface on the side of the global IPv4 cloud. You can do this by entering it a IP pool range containing only one IP, this is explained in more detail below.

3. **Do you want to configure additional address as part of your NAT pool? [y/N]**
   You are already going to use the IPv4 addresses from the outside interfaces of your NAT-PT, but you can still configure more IP addresses to be used.
4. **You need to create a public IPv4 address pool. Enter the pool's starting IP.**
   At this point you need to specify the IP pools that will be used for translations. Enter the starting IP followed by return and the ending IP (inclusively). If this range is to contain a single IP, just press return when asked for the ending IP. The second step is to configure the port ranges for this pool. By default the first port that will be used will be 1500 and the last 65000. You can simply accept these values by pressing return or specify your own. You can create as many IP pools as you wish, make sure though that they do not overlap, as NAT-PT will not check these pools for overlaps.
5. **Do you want to create a pool of public IPv4 addresses that will allow incoming connections to be dynamically mapped to appropriate IPv6 addresses? [y/N]**
   Answering yes, will allow you to create IP ranges that will be used by NAT-PT for dynamic inbound connection mapping. You need to specify the starting and ending IP addresses of the IP ranges, and you are free to create as many of these ranges as you want. These ranges should not overlap as NAT-PT will not check for any overlaps and simply accept the values you give it. Find out more about dynamic inbound connection.
6. **Do you want to create static mappings of public IPv4 addresses that will allow incoming connections to reach IPv6 hosts? [y/N]**
   If you answer yes, you will be able to create static mappings between IPv4 and IPv6 addresses. You will need to specify the IPv4 address first, followed by the IPv6 address. You can create as many static mappings as you like. Find out more about static inbound mappings.
7. **Enter the name of the first inside (IPv6) interface that you want NAT-PT to listen on. interface (eth0 eth1 sit0):**
   NAT-PT needs to know which interfaces you wish to specify as inside (IPv6) and which as outside (IPv4). The configuration maker will list all interfaces it finds on your system excluding

lo. You enter any interface name, but make sure that such an interface exists or will exist when you'll run NAT-PT, because there is no error checking performed here. After entering the first interface, you'll be given the choice to enter another. You can enter as many interfaces as you like.

8. **Enter the name of the first outside (IPv4) interface that you want NAT-PT to listen on.** In this promt you must specify the outside (IPv4) interfaces that NAT-PT will use. The same conditions apply here as to specifying inside (IPv6) interfaces. You can use the same interface for both IPv4 and IPv6, making it possible to run NAT-PT on a router with only one network interface. If you do see the section single NIC operations.
9. **Enter the TCP translation timeout in seconds [86400]:**
   Here you can setup the number of seconds of inactivity that a TCP translation will timeout after. The default is 86400 seconds (24 hours), however whenever a RST or FIN packet is cought the connection will be timed out 5 minutes later. You can simply hit return to accept the default value.
10. **Enter the UDP translation timeout in seconds [3600]:**
    Here you can setup the number of seconds of inactivity that a UDP translation will timeout after. The default is 3600 seconds (1 hour). You can simply hit return to accept the default value.
11. **Enter the ICMP translation timeout in seconds [30]:**
    Here you can setup the number of seconds of inactivity that a ICMP translation will timeout after. The default is 30 seconds. You can simply hit return to accept the default value.
12. **Enter the IPv6 prefix that will be used as the destination for that should be translated. prefix [2000:ffff::]:**
    Here you must enter an IPv6 network prefix that you will indicate packets that must be translated. The default value should work for everybody, but you may wish to use part of you IPv6 block as an IPv6 prefix, just to be safe. I prefer to use the last available network from my block for NAT-PT purposes. If you have a block like 2001:468:181:f100::/56, this would be 2001:468:181:f1ff::. Also remember that if you are using the DNS proxy totd, you should adjust it's prefix to whatever you set it here to be.
13. **Please enter the IPv4 address of the DNS server you are currently using. IPv4 DNS server:**
    Enter the IPv4 address of the DNS server you are currently using and the configuration program will return the IPv6 address that you should use. This is new address is determined based on the NAT-PT prefix and the IPv4 address of your DNS server. This part of the configuration isn't mandatory but will make it easier for you to calculate the translated IPv6 address for your DNS server if you plan to use NAT-PT's build-in DNS translator. You can also use the script below.

Prefix:	2000:ffff::
IPv4 DNS server:	
IPv6 DNS server:	

Convert

14. **Enter the path were NAPT's logfiles should be kept. Please make this directory globaly writeable. log file path [/var/log/naptd/]:**
    NAT-PT keeps logfiles of it's activity. Here you must specify the directory to store these logfiles. They will be named according to the following format:

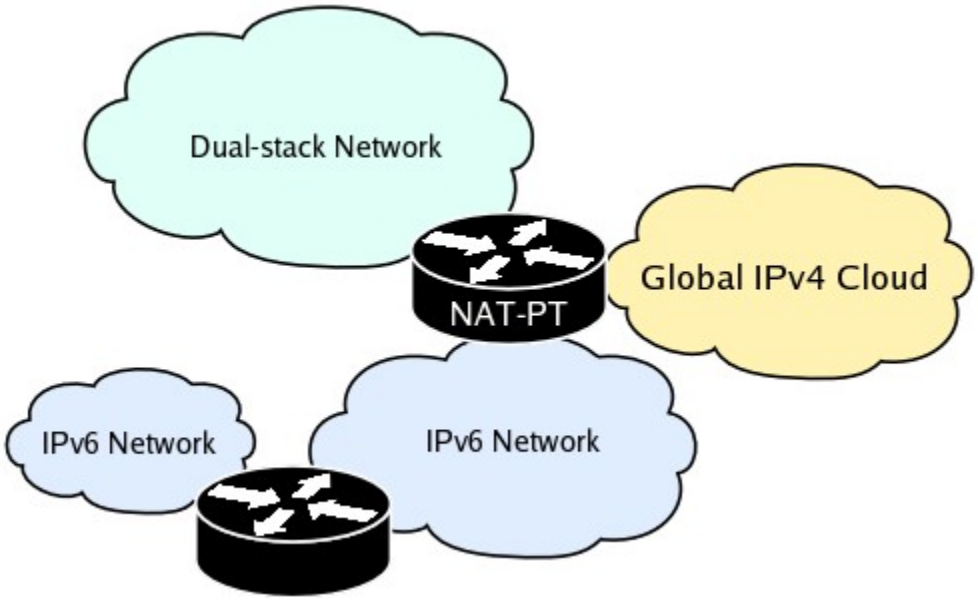
```
year.month.day.hour.minute.second.log

example:
```

```
2005.09.05.09.14.01.log
```

15. **Thank you for choosing Ataga as you IPv4/IPv6 NAT-PT solution. Setup is now complete. Type 'naptd' to start NAT-PT.**
    Congratulations, your done!

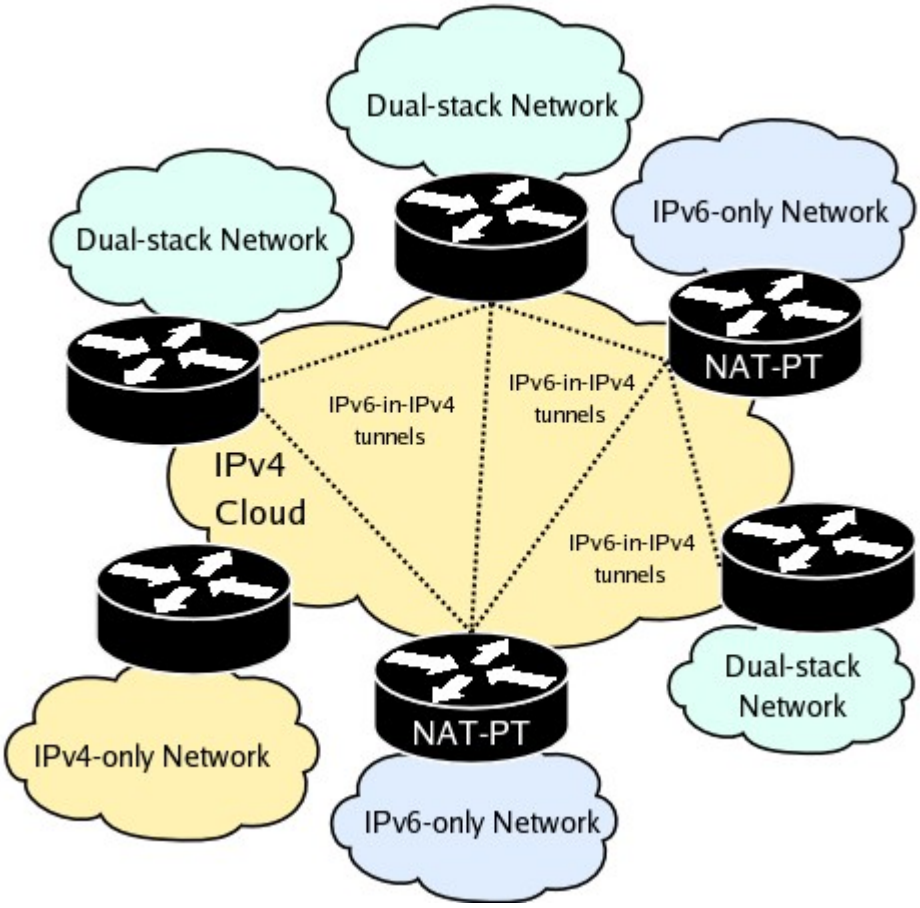
Example setup scenario

Lets look at two example setup scenarios where NAT-PT could be used. In the first we are looking to deploy an IPv6-only corporate network in which we want to take advantage of IPSec to encrypt all internal traffic. Most of your traffic will be isolated within the v6 domain, but we still need to provide a way for our employees to communicate with the rest of the world.



In the above we have two IPv6-only networks, one dual-stack network and two routers (one running dual-stack and NAT-PT and the second one being IPv6-only). The NAT-PT machine shown on this diagram can act as an IPv4 router, an IPv6 router and a IPv4/IPv6 translator all at the same time. It performs IPv4 routing between the global IPv4 cloud and the dual-stack network. It performs IPv6 routing between the dual-stack network and the IPv6-only networks. Finally it performs statefull IPv4/IPv6 translations between the IPv6-only networks and the IPv4 global cloud. The hosts on the IPv6-only networks should use NAT-PT's built-in DNS translator, while the dual-stack hosts can continue to use their old IPv4 DNS server.

The second scenario is a bit more complex and probably is a better reflections of todays complex network environment. It shows multiple network, some of them IPv4-only, some IPv6-only and some dual-stacked.



In this scenario an organization wants to migrate to IPv6, but needs to still be able to communicate with the rest of the IPv4-only world. This scenario is different from the previous one by the fact that in this case the organization has a native or tunneled IPv6 connection provided by his ISP. In this case, the DNS proxy daemon **totd** should be used. The NAT-PT machine performs IPv6 routing and IPv4/IPv6 translations. If an internal IPv6-only host tries to communicate with an IPv6-enabled host, the connection will be routed through IPv6 and communication will happen through the IPv4 cloud in a IPv6-in-IPv4 tunnel. Otherwise, if the remote host uses only IPv4, NAT-PT will translate the connection.

The setup scenario should determine how DNS queries are resolved. Use the table below for reference.

Scenario	**DNS Resolution**
Dual-Stack	Use a regular DNS server. If a remote host has an IPv6 address it will be prefered, otherwise IPv4 will be used.
IPv6-only + tunneled/native IPv6 connectivity	Use the DNS proxy daemon **totd**. If a remote host has an IPv6 address it will be used, otherwise the IPv4 address will be translated using the NAT-PT prefix (remember the NAT-PT prefix and the prefix configured with totd must be the same).
IPv6-only, no IPv6 connectivity available	Use NAT-PT's built-in DNS translator. The remote host's address will always be translated using the NAT-PT prefix.

Memory usage

NAT-PT doesn't put any significant strain on a router's memory. After initializing it should use about 1.1 MB of shared memory. This amount will increase by 48 bytes per translation created. After creating 30,000 translations, this increases memory usage by about 1.3 MB. Other source of memory usage include free IP+port pair that can be used for translations. Each of these takes up 8 bytes. This can quickly become a large amount of memory if for example your NAT pool consists of 254 IPs and 63500 ports per IP. This would total over 123 MB of memory. However NAT-PT maintains separate free IP+port pairs per each transport layer protocol. This brings the memory usage up to over 369 MB (123MB * 3 protocols; TCP, UDP, ICMP). However, NAT-PT uses a dynamic mechanism that allocates IP+port pairs only when the pools are exhausted. In this way the memory used by the IP+port pair is insignificantly small.

Security

Security is a big concern in todays computer world. NAT-PT mitigates most security risks by running by dropping root privileages soon after startup. Securing NAT-PT really comes down to securing the configuration file that it uses, because the current mechanism used to read the configuration file can be exploited if the configuration file is tampered with (this will be changed in future releases). The configuration file should be only readable by root and only modified using the naptd-confmaker program. If a remote attacker modifies the configuration file that NAT-PT uses, then he must already have gained root privileages. While NAT-PT is running it needs to have read access to the following files:

```
/proc/net/ipv6_route
/proc/net/route
```

If you are running SELinux make sure that they are readable by NAT-PT.

One DOS attack scenario that can be attempted is creating thousands of half-open TCP connections to exhaust the free IP+port pool. In this scenario the attacker must be on one of the internal (IPv6) networks to make the attack feasible. If you ever experience this (check NAT-PT's logfile) you can modify the TCP translation timeout. Refer to configuration details for more information.

Finally there is the case of Application Level Gateways that are loaded during NAT-PTs startup. One possible scenario would be for an attacker to create ALG that would be designed to crash NAT-PT, thus causing a denial of service attack. In order to mitigate this attack, NAT-PT plugin directory (/usr/lib/naptd/plugins) must be only readable and writeable by root.

Advanced features

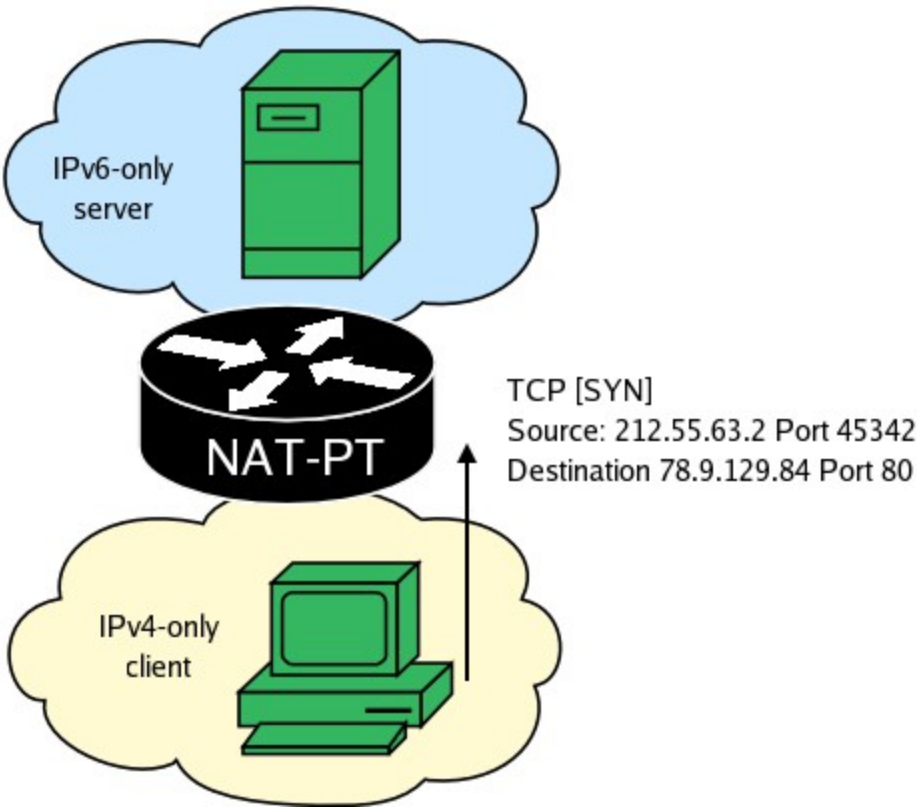
NAT-PT employs number of advanced features including static and dynamic inbound mappings, and single NIC operations. You don't need to know these in order to use NAT-PT in a simple setup scenario. However, these features can be very helpful in specific situations.

Static inbound mappings

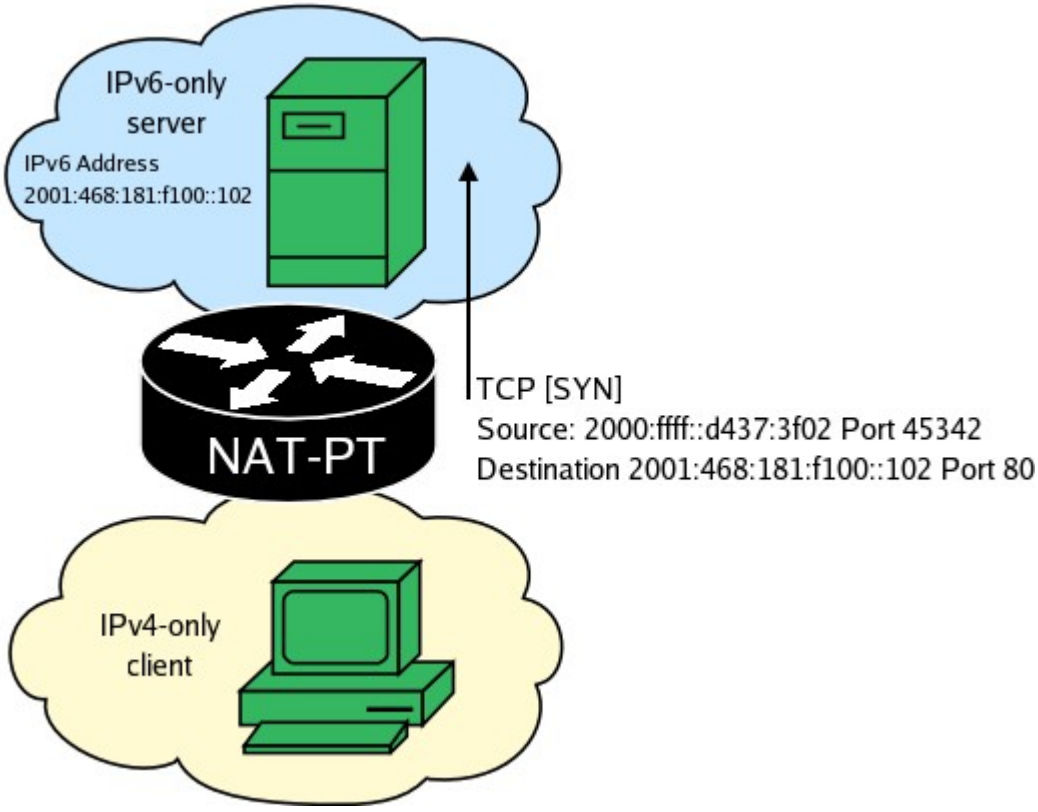
Under normal circumstances NAT-PT allows new connections to be established on from the inside of the network (IPv6 side). In such a situation, the source IPv6 address and port are translated and replaced with a IP+port pair. Depending on your network topology, it might be necessary to allow remote connections

to be established to certain internal hosts. If you are running a public web or mail server you will need a way to allow remote hosts to connect to your server. This can be done using static inbound mappings which can be configured using naptd-confmaker. When a static connection is created NAT-PT will establish a mapping between a public IPv4 address and an IPv6 address and will perform stateless translation between these addresses.

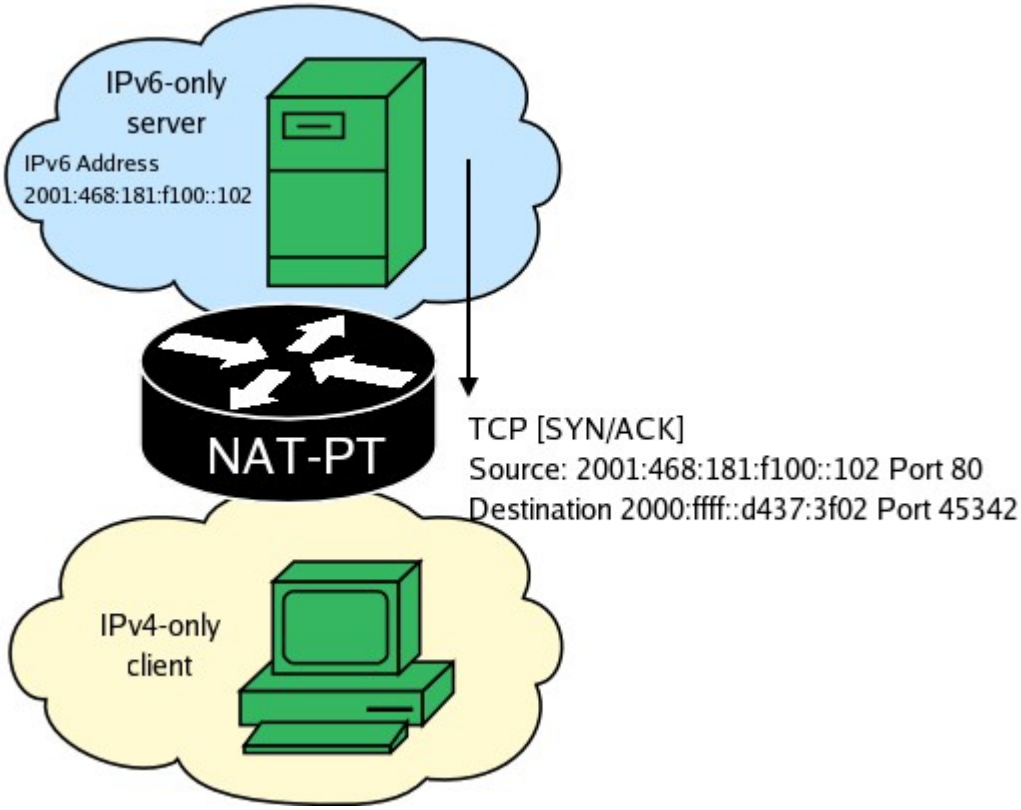
When using static mappings the communication between the hosts looks like the following.



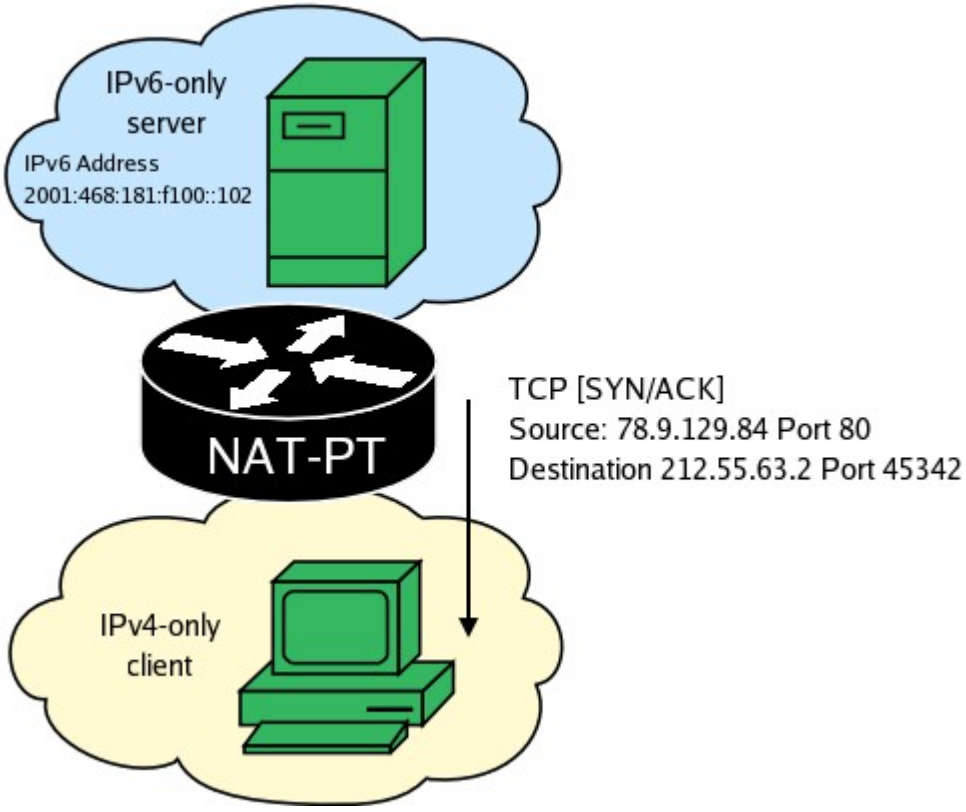
NAT-PT finds the static mapping and translates the packet.



The server responds to the TCP SYN packet.



NAT-PT finds the static mapping and translates the packet.

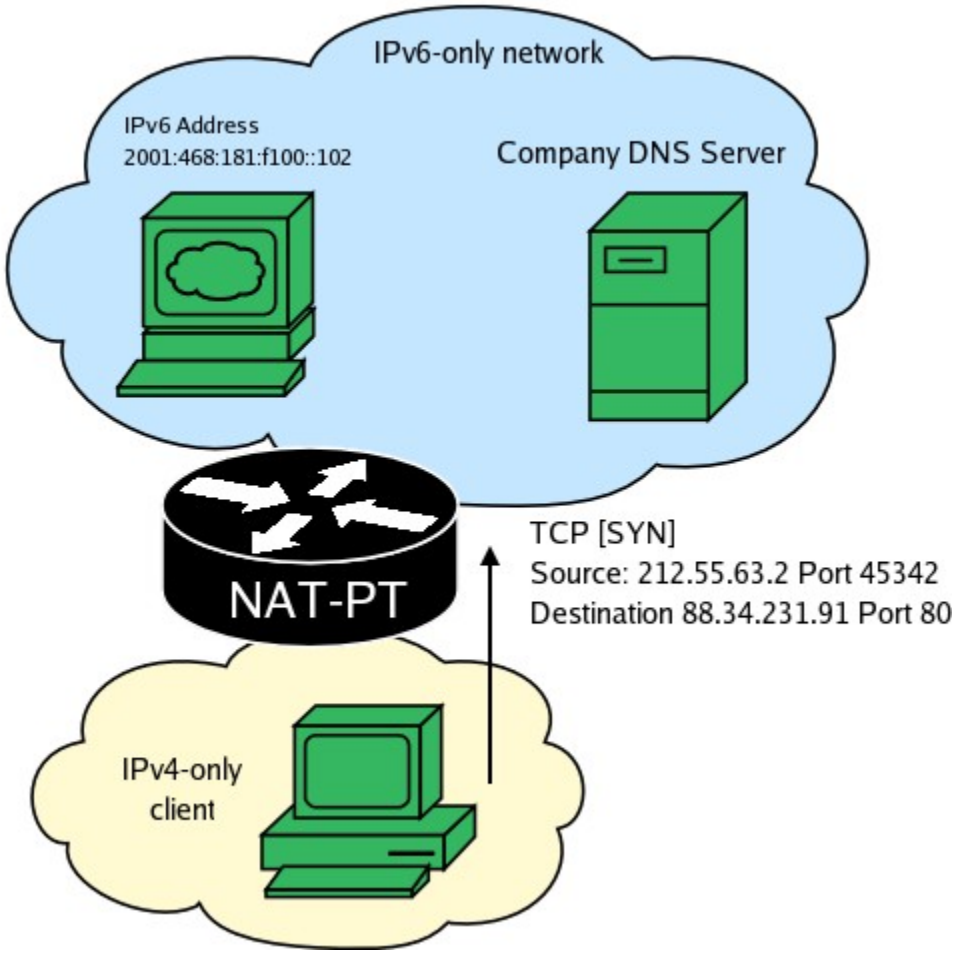


Dynamic inbound mappings

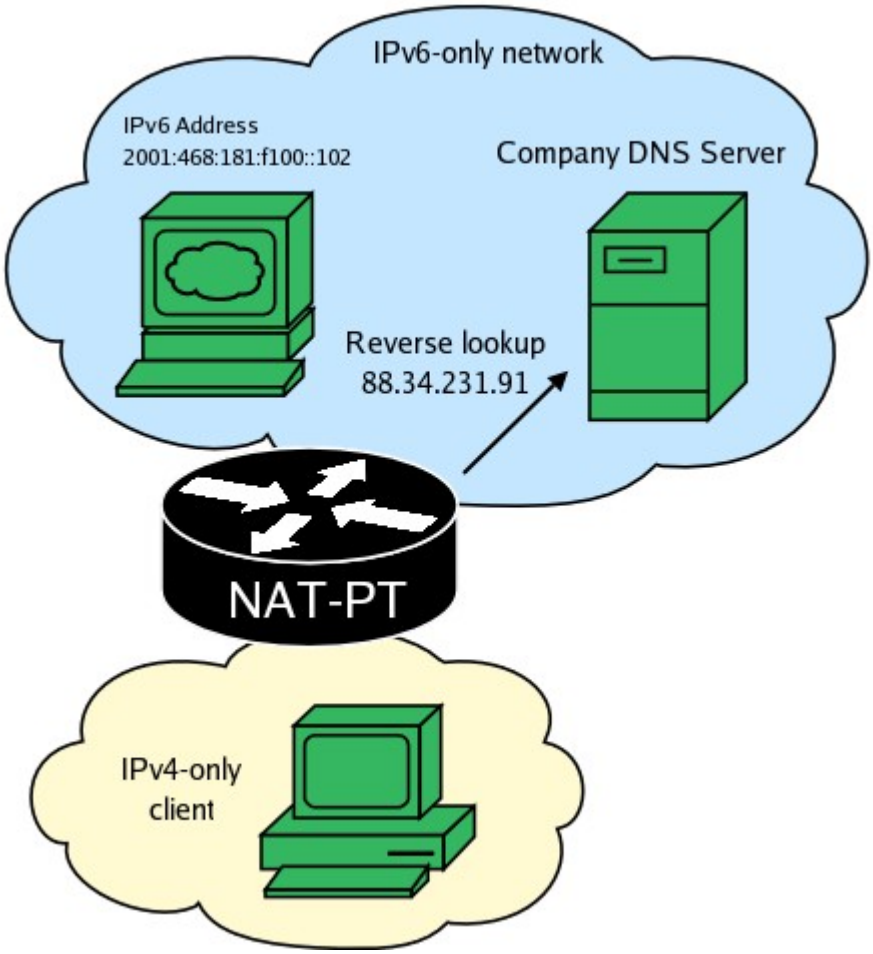
An alternative to using static inbound mappings is the use of dynamic ones. This can be especially helpful when you have a large number of servers that need to be globally reachable from IPv4-only clients. The way dynamic inbound mappings work is that NAT-PT creates inbound mappings based on DNS information. You use naptd-confmaker to first create an IPv4 address range that NAT-PT will attempt to create dynamic mappings on. You then update your DNS information so that every hostname in your zone has both its IPv6 address and its old IPv4 address.

When an IPv4-only client attempts to contact one of your servers, NAT-PT first looks for a inbound mapping for the given destination IPv4 address. If it can't find it, but the destination IPv4 address belongs to the dynamic inbound mappings range, NAT-PT will attempt to create a mapping using DNS. This happens in the following steps.

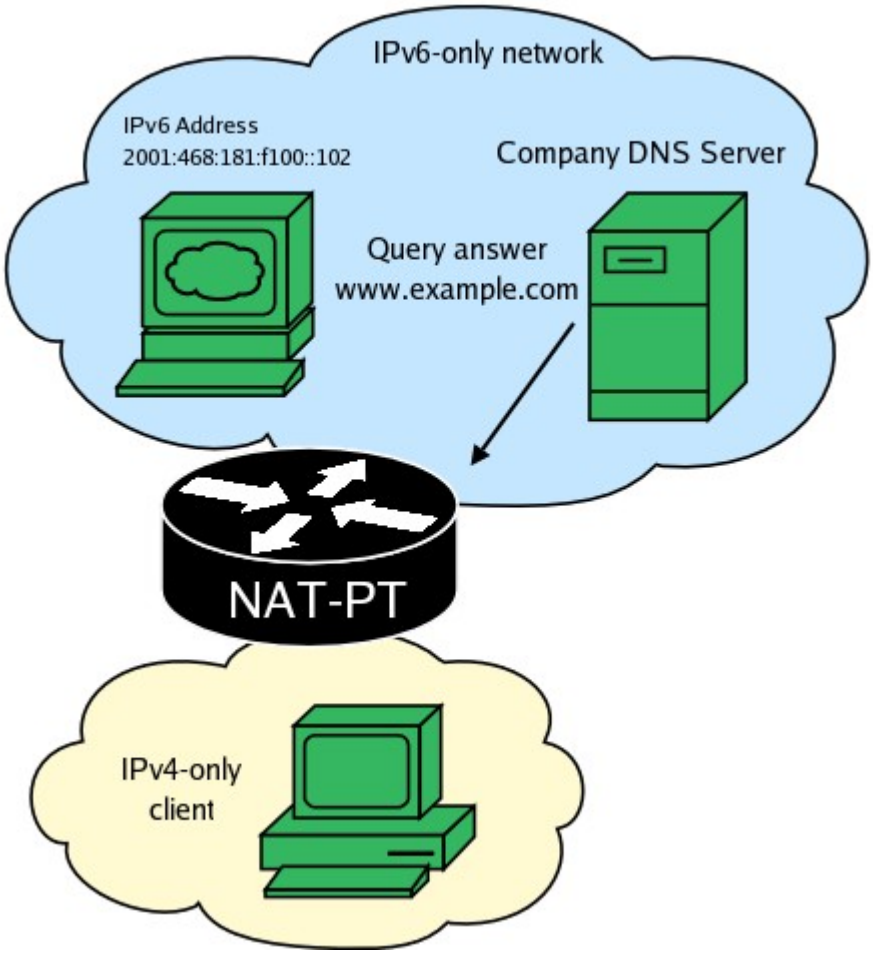
First NAT-PT captures a packet with a destination belonging in the dynamic inbound mappings range.



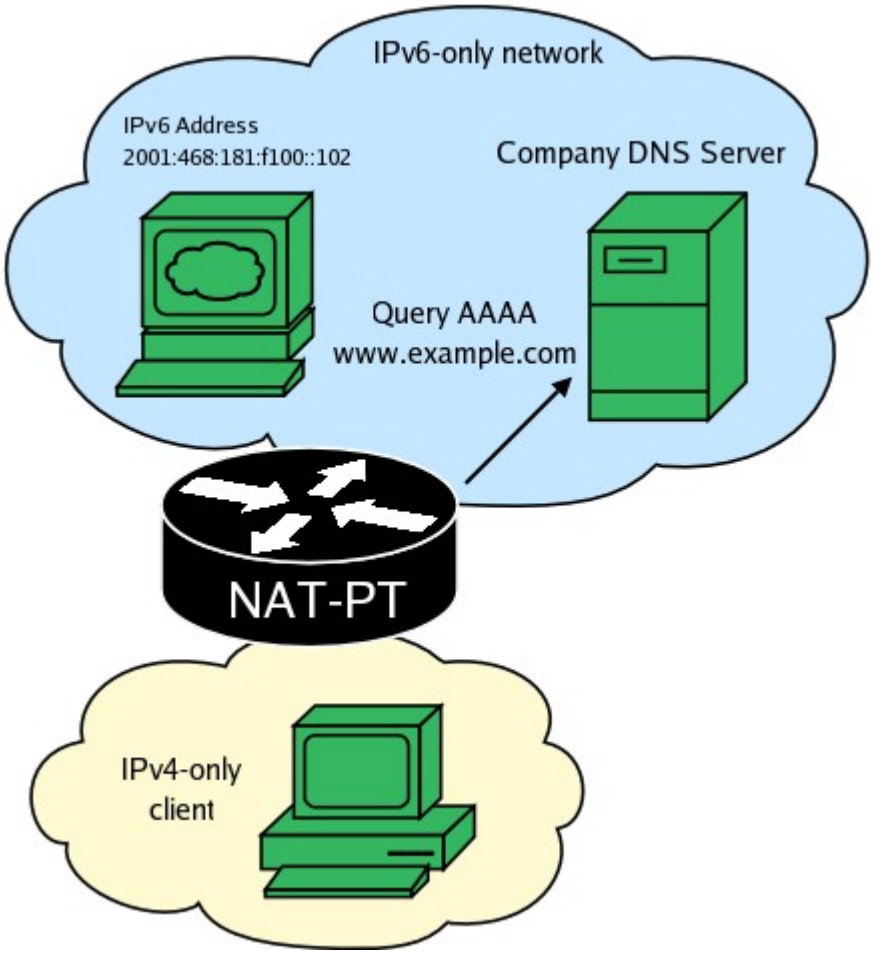
NAT-PT attempts to perform a reverse DNS resolution on the packet's destination. This is handeled by a separate thread while all other packets are being continously translated.



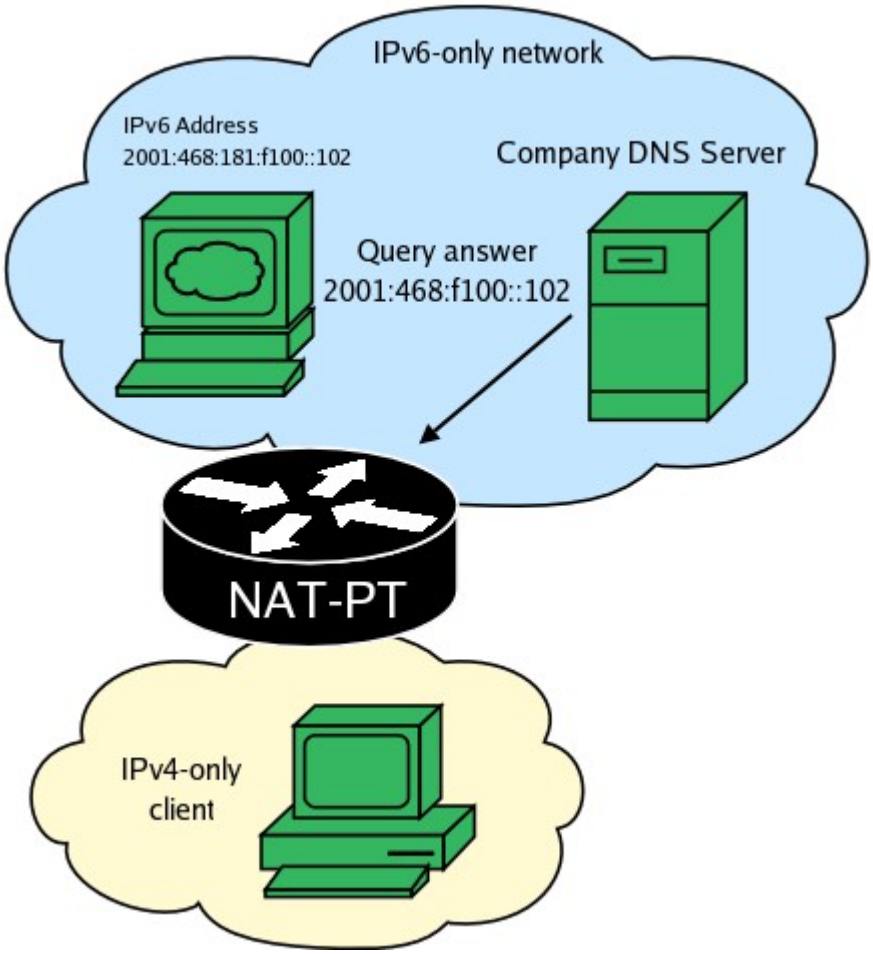
The DNS server responds with the destination's FQDN.



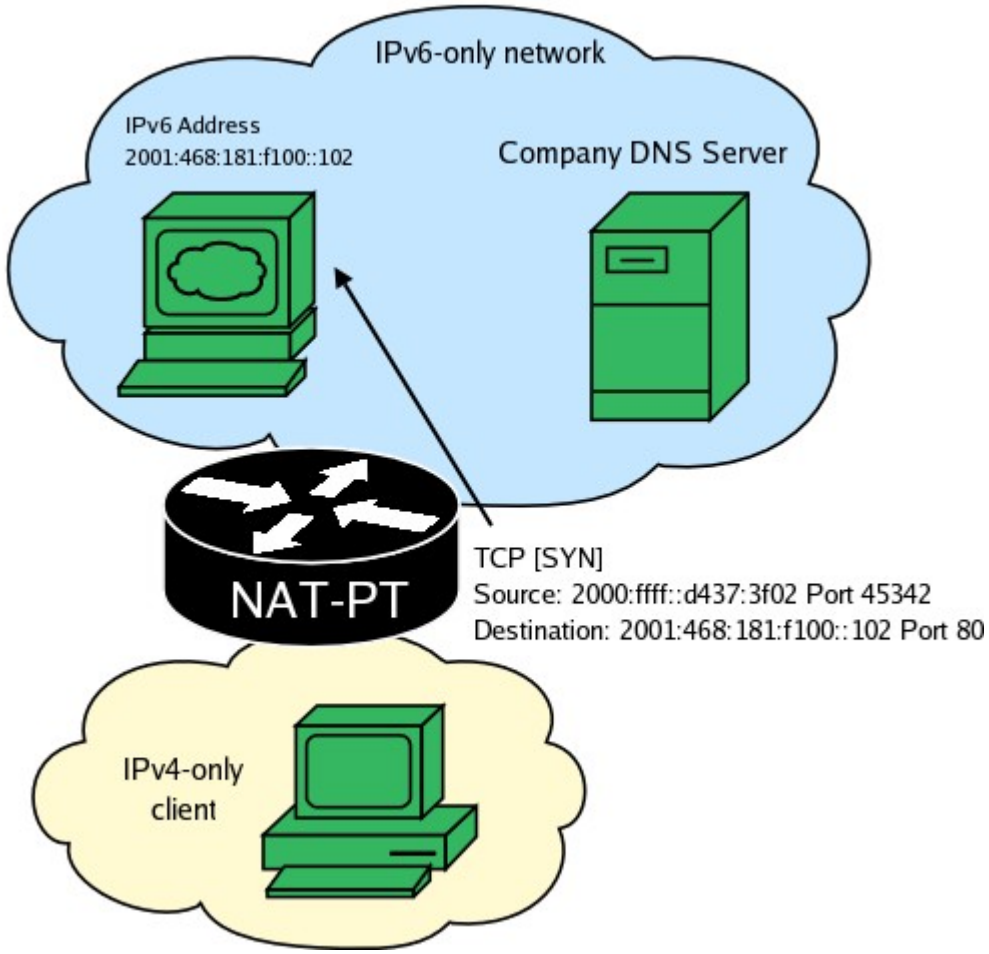
NAT-PT performs a AAAA lookup on the FQDN.



The DNS server return the AAAA record.

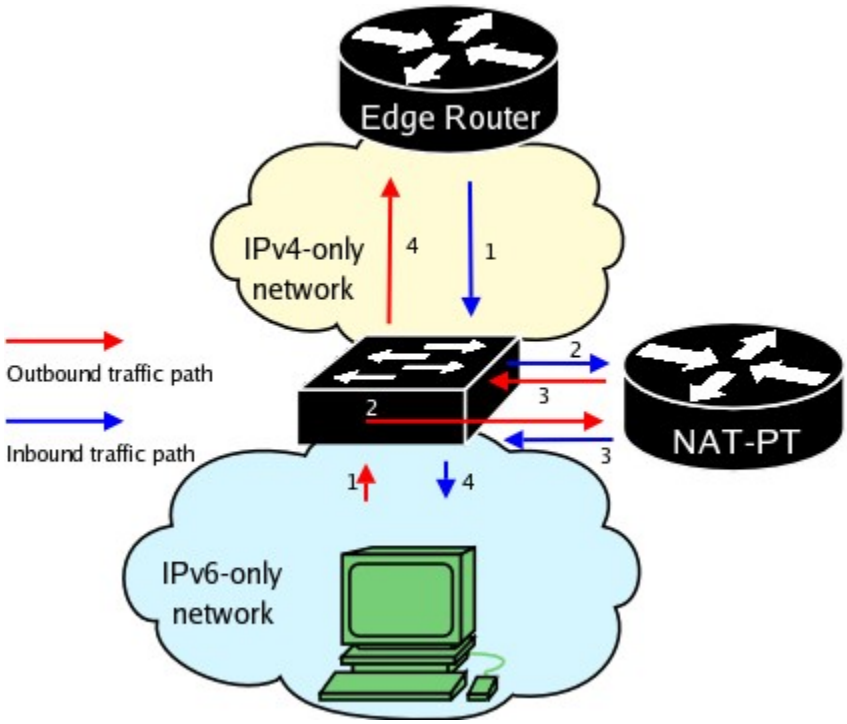


NAT-PT creates a mapping between the returned IPv6 address and the packet's original IPv4 destination, it then translates the packet and sends it to its destination.



Single NIC operations

In order to make the transition to IPv6 easier, NAT-PT has been designed by to run on commodity hardware and with minimum system requirements. One of its features is the ability to run correctly on a system that has only one Network Interface Card (NIC). In order to have NAT-PT use a single NIC, simply specify the same interface as both the inside and outside of your NAT. As long as your machine has routes to both IPv4 and IPv6 network, everything should work fine. A single NIC setup is show below.



Application Level Gateways

Under normal circumstances, NAT-PT is limited to translating IP headers and transport layer header. In some cases, this may not be sufficient to ensure full end-to-end transparent communications, because many protocols carry IP and port information in their packet's payload, therefore creating a need for deep packet inspection and translation. NAT-PT implements such functionality by using a plugin based system of Application Level Gateway (ALGs) that extend it's functionality to incorporate deep packet inspection and translation. The current release of NAT-PT (v.0.1) is shipped with two such plugins, designed to translate the FTP and DNS protocols. NAT-PT will open /usr/lib/naptd/plugins at startup and attempt to load any ALG it finds there. In the future new plugins will be added to NAT-PT on the basis to copying them to the plugin directory and simply restarting NAT-PT.

File Transfer Protocol

This ALG works on the basis of inspecting and translating certain FTP response codes and commands. It also keeps track and translates TCP sequence numbers and acknoledgement numbers, because the packet payload's size changes when crossing the IPv4/IPv6 boundry. The following translations occur:

IPv4 Side	IPv6 Side
150	150
227	229
PASV	EPSV

This is an early implementation of the FTP plugin, and may not work correctly in all possible scenarios, it has been confirmed to work in most cases.

Domain Name System

The DNS ALG works only with UDP DNS connections as of now. This will change to include TCP based DNS connections in the future. This ALG works by inspecing and translating queries and resource records (RR). It will translate a AAAA request into an A request and later map the results to AAAA RR and translate the query type back to AAAA. This ALG is still subject to change.

Troubleshooting

If the above documentation does not answer your questions, email me.