IPV4 address shortages and expanding internet routing tables are still problems

* RFC - 1917 is an appeal to return unused address blocks to Internet Assigned Numbers Authority (IANA) for redistribution.

Address allocation for private internets RFC - 1918 suggests organizations use private address space with translation performed on a smaller “routable” pool of addresses at edge of network. IANA has reserved:

- 10.0.0.0 - 10.255.255.255 (10.0.0.0/8 prefix)
- 172.16.0.0 - 172.31.255.255 (172.16.0.0/12 prefix)
- 192.168.0.0 - 192.168.255.255 (192.168.0.0/16 prefix)

- Private not routable on internet
- Can be used simultaneously by many organizations
- Requires a network address translator (NAT) for internet access.
- Easier for customer to change ISP’s.
- NAT breaks IP security (IPSEC) because changing IP address in between end points invalidates cryptographic transforms.

* Address allocation from the reserved class A address space.
RFC-1797 explores allocation of upper half of class A by using CIDR blocks from the 64.0.0.0/2 address space
Network Address Translation

(Source: http://www.suse.de/~mha/linux-ip-nat/diplom/node4.html)

• Can do address translation either statically or dynamically
• Static- a given fixed original IP address is always translated into the same NAT IP
• Dynamic- NAT IP depends upon runtime conditions and may different each time
Example of Static NAT:

This NAT strategy is easy to implement, since the entire translation process can be written as one line containing a few simple logic transformations:

new-address = new-network OR (old-address AND (NOT netmask))

In addition, no information about the state of connections that are being translated needs to be kept, looking at each IP packet individually is sufficient. Connections from outside the network to inside hosts are no problem, they just appear to have a different IP than on the inside, so static NAT is (almost) completely transparent.
Dynamic Address Translation

Dynamic NAT is more complex than static NAT, since we must keep track of communicating hosts and possibly even of connections which requires looking at TCP information in packets.

Some people use this as a security measure: it is impossible for someone outside a network to get useful IP numbers to connect to of hosts behind a NAT router doing dynamic address translation by looking at connections that take place, since next time the same host may connect using a completely different IP.

Connections from outside are only possible when the host that shall be reached still has a NAT-IP assigned, i.e. if it still has an entry in the dynamic NAT table, where the NAT router keeps track of which internal IP is mapped to which NAT IP.
Example of Dynamic NAT:

• NAT rule: dynamically translate all IPs in (class B) network 138.201 to IPs in (class C) network 178.201.112

• each new connection from the inside gets assigned an IP from the pool of class C addresses, as long as there are unused addresses left

• if a mapping already exists for the internal host this one is used instead

• as long as the mapping exists the internal host can be reached via the IP that has been (temporarily) assigned to it.

• In the next figure, the left side is internal, connections there cause the result on the right side which is the external network.
Dynamic NAT

<table>
<thead>
<tr>
<th>Internal IP</th>
<th>NAT IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>138.201.148.32</td>
<td>178.201.112.34</td>
</tr>
<tr>
<td>138.201.148.151</td>
<td>178.201.112.11</td>
</tr>
</tbody>
</table>
Dynamic NAT with Single IP Address

Local Host

172.18.3.1

NAT Router

200.24.5.8

Internet

SA = 172.18.3.1
DA = 25.8.2.10

SA = 200.24.5.8
DA = 25.8.2.10

Translation Table

<table>
<thead>
<tr>
<th>DA</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.8.2.10</td>
<td>172.18.3.1</td>
</tr>
</tbody>
</table>
Dynamic NAT: More Complicated Situations

• to allow multiple connections to same external IP address
  • use k IP addresses for NAT router to allow k simultaneous connections to same external address
• to allow many to many connections
  • add port numbers to further distinguish internal addresses
Masquerading NAT

• A very special case of dynamic NAT is many:1-translation, a.k.a. masquerading which became famous under that name because Linux can do it.

• It is probably the kind of NAT-technique that is used most often these days.

• Here many IP numbers are hidden behind a single one. In masquerading an almost arbitrary number of connections is multiplexed using TCP port information. The number of simultaneous connections is limited only by the number of TCP-ports available.

• Incoming connections are impossible with masquerading, since even when a host has an entry in the masquerading table of the NAT device this entry is only valid for the connection being active. While it is true that incoming connections are impossible we can take additional measures to enable them, but they are not part of the masquerading code.

• The greatest advantage of masquerading for many people is that they only need one official IP-address but the entire internal network can still directly access the Internet.
Example of Masquerading:

• NAT rule: masquerade the internal network 138.201 using the NAT routers own address

• for each outgoing packet the source IP is replaced by the routers (external) IP, and the source port is exchanged against an unused port from the range reserved exclusively for masquerading on the router

• if the destination IP of an incoming packet is the local router IP and the destination port is inside the range of ports used for masquerading on the router, the NAT router checks its masquerading table if the packet belongs to a masqueraded session; if this is the case, the destination IP and port of the internal host is inserted and the packet is sent to the internal host

• In the next figure, the left side is the internal network, the right side is the external network. All data appears from source 195.112.12.161 but different ports.
Example of Masquerading

<table>
<thead>
<tr>
<th>Internal IP / Port</th>
<th>local NAT-Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>138.201.148.32 1257</td>
<td>63451</td>
</tr>
<tr>
<td>138.201.160.201 4192</td>
<td>63452</td>
</tr>
</tbody>
</table>
Linux Implementation Detail:

• Masquerading usually uses ports in the upper range, in Linux this range starts at port 61000 and ends at 61000+4096, which is the default and can easily be changed by editing linux/include/net/ip_masq.h.

• This also shows that the Linux implementation by default only allows 4096 concurrent connections.

• To allow masqueraded connections on ports outside of such a port range requires keeping and managing even more information about the state of connections.

• Linux, for example, simply treats all packets with destination IP = local IP and destination port is inside the range used for masquerading as packets that have to be demasqueraded, i.e. they are answers to packets that have been masqueraded on their way out.