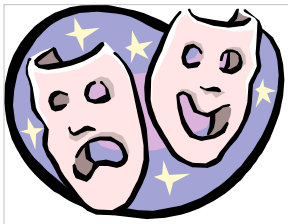


Network Address Translation

All you want to know about

(C) Herbert Haas 2005/03/11

In this chapter we discuss the idea of Network Address Translation and special issues associated to it. Invented in 1994, NAT became a quite popular technique to save official network addresses and to hide the own network topology from the Internet.

Note:

In this chapter the Cisco IOS syntax is used for configuration examples. IOS is a trademark of Cisco Systems Inc.

See http://www.cisco.com for further information.

Reasons for NAT

- Mitigate Internet address depletion
- Save global addresses (and money)
- Conserve internal address plan
- TCP load sharing
- Hide internal topology

NAT allows a router to swap packet addresses. The initial idea was to mitigate IP address depletion by masquerading internal IP addresses with (perhaps a smaller number of) official addresses. We will discuss this later on.

The first and the second point reflect the same thing, but the first statement comes from the ISP while the second point is an argument for the customer.

The third point means that the customer does not need to change her address plan when she switches to another ISP.

As stated in the fourth point, NAT additionally allows for TCP load sharing. Assume a bunch of servers represented by a single IP address to the outside.

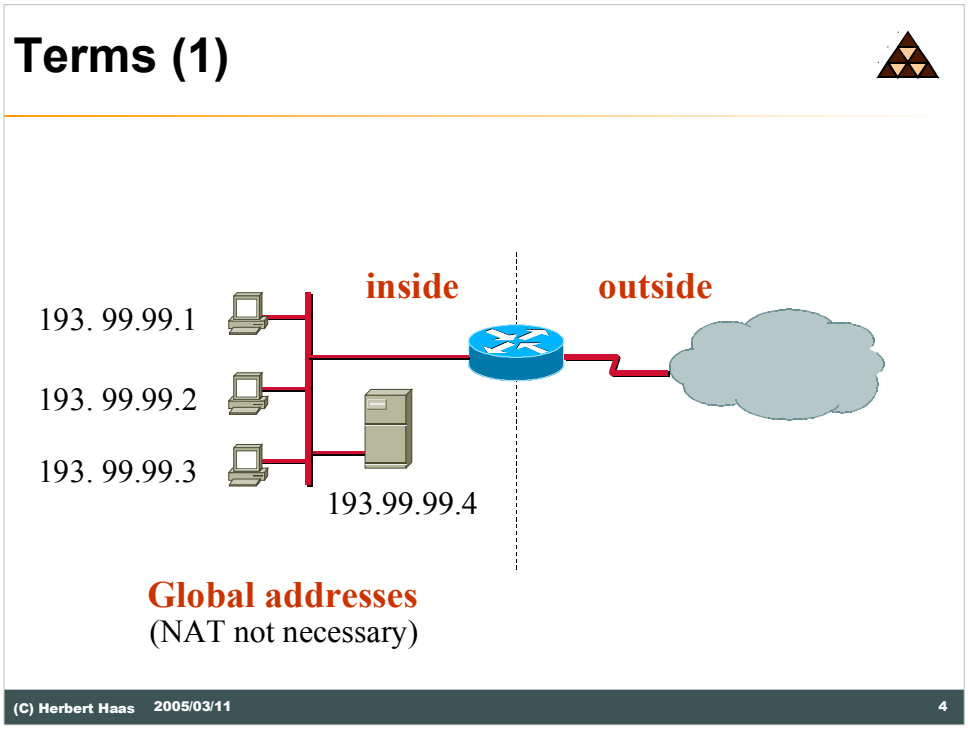
Finally, NAT improves network security by hiding the actual host addresses. Frequently NAT boxes are combined with proxy and firewalling functions.

Credits: The Creators of NAT

Paul Francis

Kjeld Borch Egevang

NAT was invented in May 1994 by Paul Francis and Kjeld Borch Egevang. Paul Francis is currently chief scientist at Tahoe-Networks. K. Egevang works at Intel Denmark. They have written RFCs about NAT, most importantly RFC-1631, "The IP Network Address Translator (NAT)".

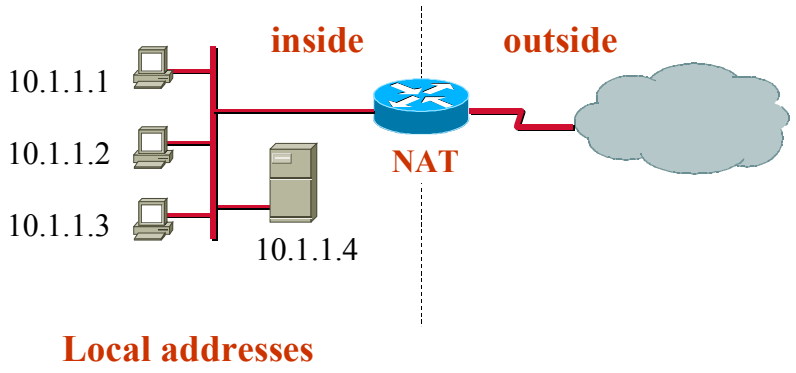


To understand standard documents such as RFCs or vendor documents such as Cisco white papers or similar, it is **very** important to understand four terms.

Firstly we have to distinguish the *inside* from the *outside* world. *Inside* is our own network (which we want to hide using a NAT-enabled router later on). Outside is the rest of the world, especially the Internet.

Secondly, suppose we do *not* use NAT. Therefore we use global addresses. That is, we use addresses that are registered by the NIC and can be seen from outside.

Terms (2)



Using a NAT enabled router we can use inside local addresses which are *not* unique in the world. This addresses are not registered and must be *translated* to global addresses.

Note that we can already distinguish between **inside-local** and **inside-global** addresses.

Terms (3)

This NAT–Table is maintained inside the router

Inside local IP address		Inside global IP address
10.1.1.1	⟷	193.99.99.1
10.1.1.2	⟷	193.99.99.2
10.1.1.3	⟷	193.99.99.3
10.1.1.4	⟷	193.99.99.4

Remember the terms **inside local** versus **inside global.**

Of course the inside global address is basically seen from the outside. But these addresses belong to **our** hosts, so we call them inside.

Simple NAT translates between these two types of addresses.

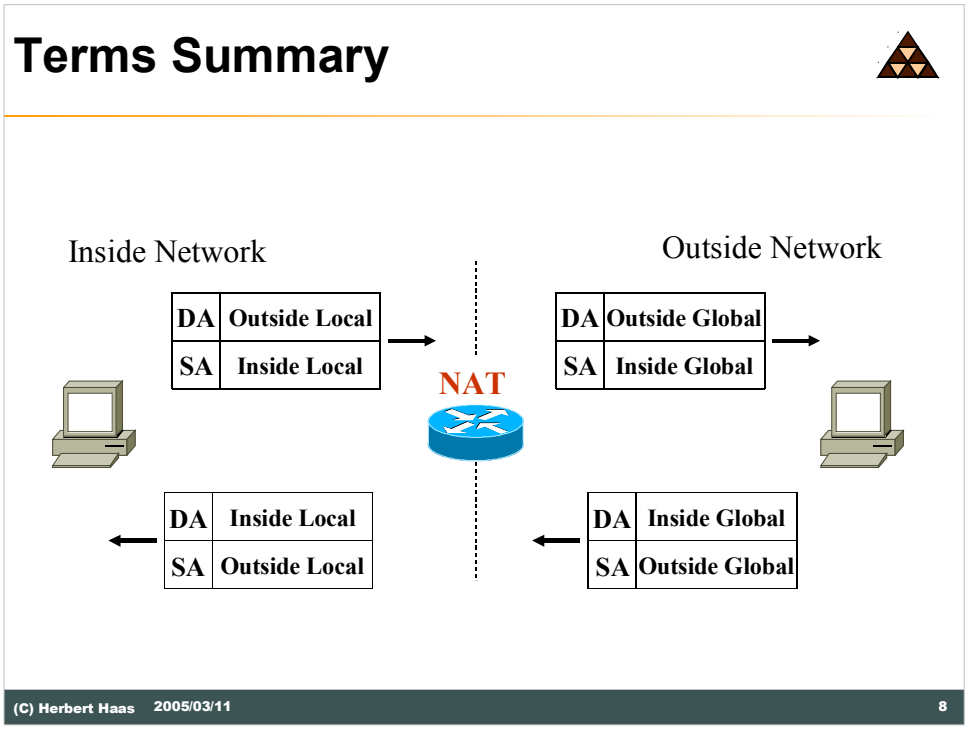
Terms (4)

- *Local* versus *global* address
  - Reflects realm of usage (inside or outside)
- *Inside* versus *outside* world
  - Reflects origin

Since these terms are so important and many people and some documents confuse them, we give a summary here.

Note that local addresses have local meaning. That is: inside devices can only deal with packets having local addresses. The NAT router is responsible to translate global addresses to local and vice versa – *if necessary* ! (If you later understand the last two italic-written words, then you got it.)

Note that *outside* does not mean another (foreign) NAT-domain. Outside means simply the Internet or everything beyond the NAT-router.



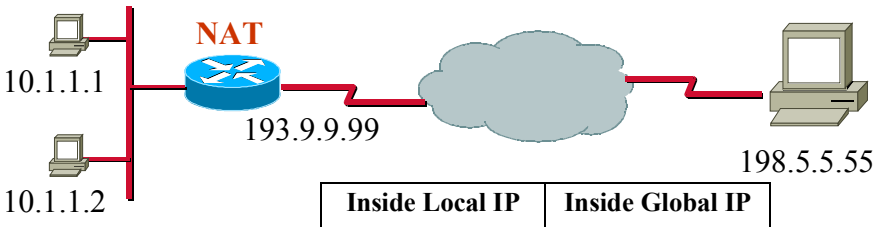
This slide summarizes all terms by showing packets flowing from inside to outside and from outside to inside. Local is what we can *use* inside our network. Inside local *source* addresses are always private addresses otherwise we won't use NAT.

Outside local addresses can be either private or registered. Mostly they are registered, but in certain cases we might want to present official registered addresses in incoming packets as being private addresses. See the slide "Outside Address Translation" for this special case. Typically the outside local address is mostly identical with the outside global address.

The inside global address is the official address of our hosts as seen in the Internet. What people mostly expect from NAT is to translate an inside local address to an inside global address. Both addresses belong to a host *inside* our network.

The outside global address is the official registered IP address of an Internet host. Mostly it is identical with our outside local address we use as destination address for outgoing packets. See the slide "Outside Address Translation" for exceptions.

Basic Principle (1a)



Simple NAT Table

Inside Local IP	Inside Global IP
10.1.1.1	193.9.9.1
10.1.1.2	193.9.9.2
....	....

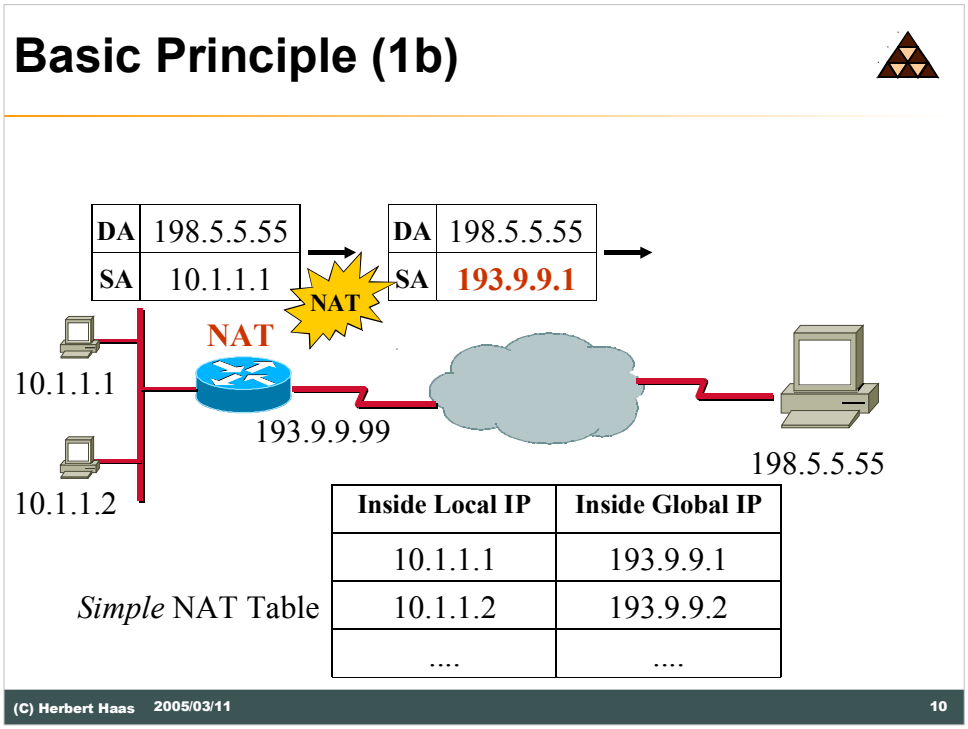
(C) Herbert Haas 2005/03/11 9

1) Suppose the user at host 10.1.1.1 opens a connection to host 198.5.5.55.
2) The first packet that the router receives from host 10.1.1.1 causes the router to check its NAT table.
3) The router replaces the source address with the inside global address found in the NAT table. If no translation entry exists, the router determines that the source address must be translated dynamically and selects a legal global address from the *predefined* dynamic *address pool* and creates a translation entry.

Note: *static* versus *dynamic* entries.

Example for a *static* configuration:

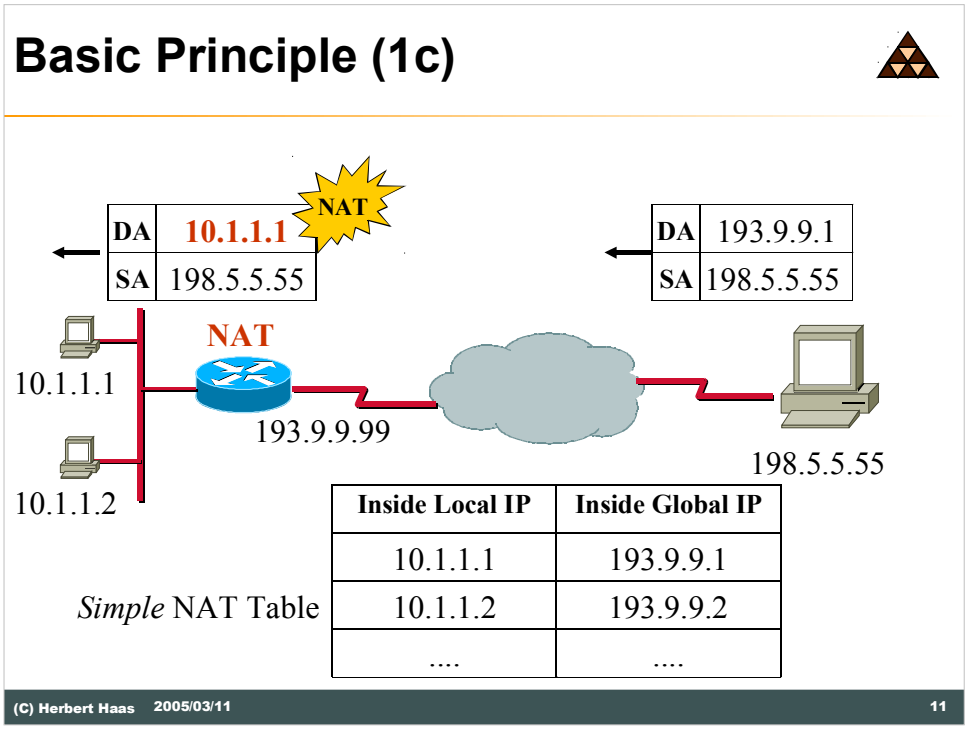
```
ip nat inside source static 10.1.1.1  193.9.9.1
interface ethernet 0
  ip address 10.1.1.99  255.0.0.0
  ip nat inside
interface serial 0
  ip address 193.9.9.99  255.255.255.0
  ip nat outside
```



In many NAT implementations the host portion of an IP address remains unchanged. Only the prefix is translated.

Example for a *dynamic* configuration:

```
ip nat pool mynatconf 193.9.9.1 193.9.9.254 netmask 255.255.255.0
ip nat inside source list 1 pool mynatconf
!
interface ethernet 0
           ip address 10.1.1.99  255.0.0.0
           ip nat inside
!
interface serial 0
           ip address 193.9.9.99 255.255.255.0
           ip nat outside
!
access-list 1 permit 10.0.0.0  0.255.255.255
```



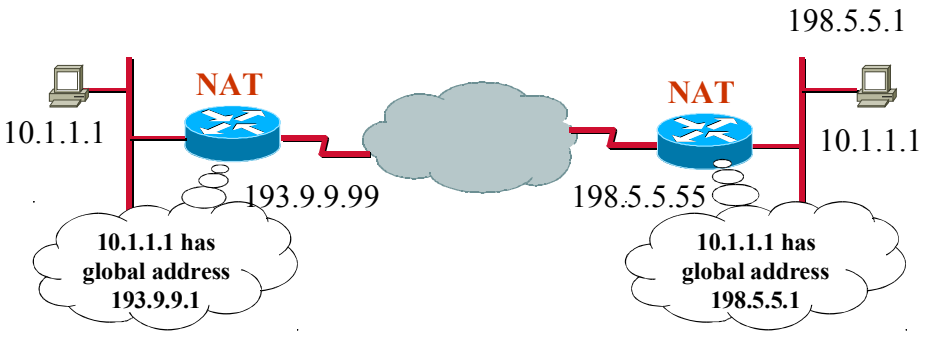
1) Host 198.5.5.55 responds to host 10.1.1.1 by using the inside global address 193.9.9.1 as destination address.
2) When the router receives a packet with the inside global address 193.9.9.1 it performs a NAT table lookup to determine the associated inside local address.
3) The router translate 193.9.9.1 to 10.1.1.1 and forwards the packet to host 10.1.1.1.

FYI:

Inside-to-outside translation occurs after routing

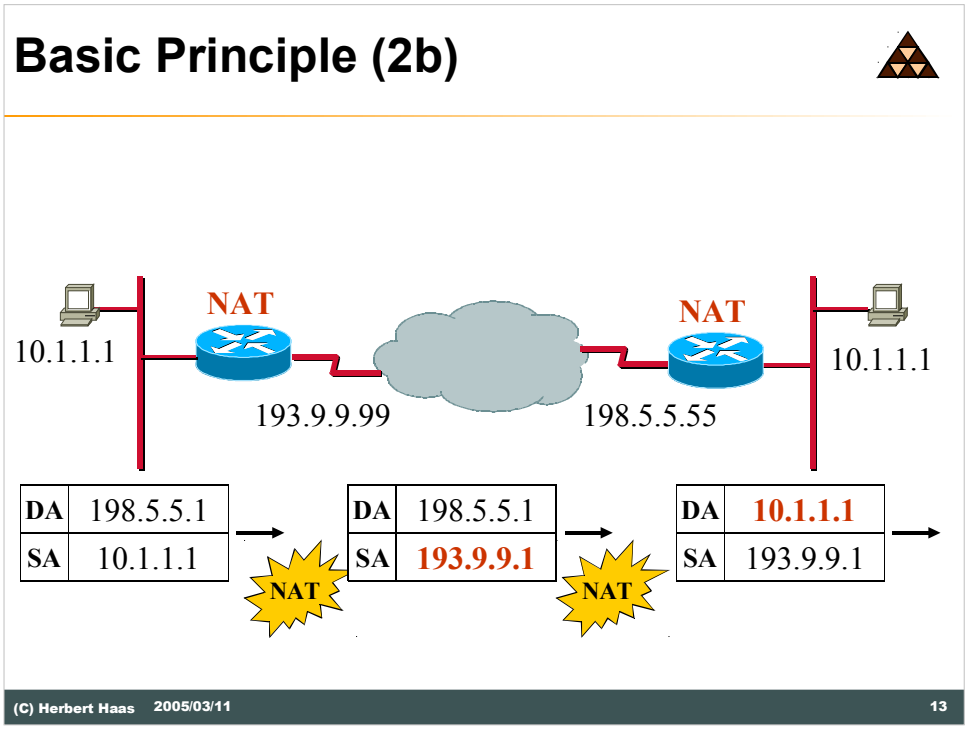
Outside-to-inside translation occurs before routing

Basic Principle (2a)



In this example we assume that the PC in the left network wants to send an IP packet to the PC in the right network. Note that both networks use NAT. *Outside* is everything between the two NAT-enabled routers.

By accident they use the same inside-local addresses. But this does not matter anyway. You can also imagine using two completely different inside-local addresses.



Observe these translations as depicted above:

1) The left host (10.1.1.1) send a packet to the right host (also 10.1.1.1). Of course the right host is known by its outside-local address (198.5.5.1), which is used as destination address.
2) The left NAT-enabled router translates only the source address (which was an inside-local address) to an inside-global address (193.9.9.1). The destination address (which is an outside-local address) remains unchanged and is now called outside-global, while the packet traverses the Internet.
3) The right NAT-enabled router only changes the destination address (which he regards as inside-global) by translating it to an inside-local one. The source address is regarded as outside-global and remains unchanged but is now called outside-local.

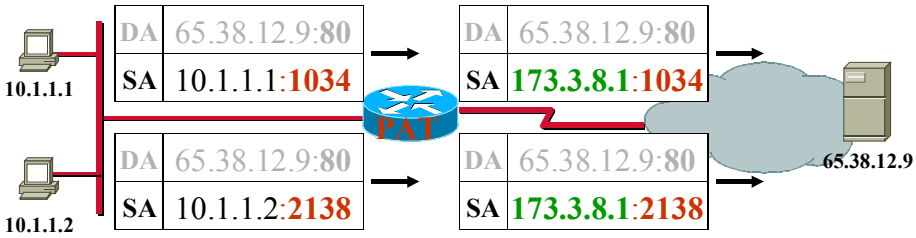
Overloading (PAT)

- Common problem:
  - Many hosts inside
  - But only one or a few inside-global addresses available
- Solution:
  - Many-to-one Translation
  - Aka *"Overloading Inside Global Addresses*"
  - Aka "*PAT"*

Many-to-one translation is acomplished by identifying each traffic according to the source port numbers. This method is commonly known as Port Address Translation (PAT). In the IETF documents you will also see the abbreviation **NAPT**. In the Linux world it is known as *masquerading*.

When N inside hosts use the same source port numbers, the PAT-routers will increase N-1 of these identical source port numbers to the next free values.

Overloading Example (1)



Prot.	Inside Local	Inside Global	Outside Local	Outside Global
TCP	10.1.1.1:**1034**	**173.3.8.1**:**1034**	65.38.12.9:**80**	65.38.12.9:**80**
TCP	10.1.1.2:**2138**	**173.3.8.1**:**2138**	65.38.12.9:**80**	65.38.12.9:**80**

Extended Translation Table

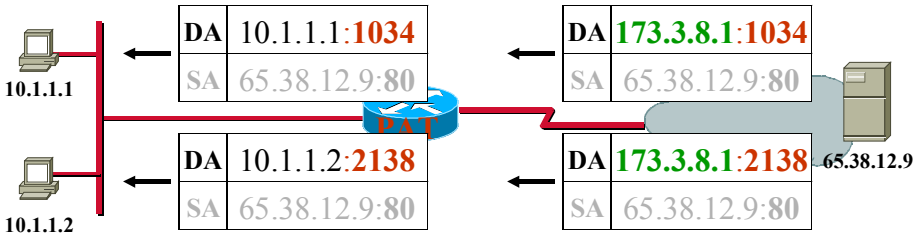
(C) Herbert Haas 2005/03/11 15

The port number is the differentiator. Note that the TCP and UDP port number range allows up to 65,536 number per IP address. This number is the upper limit for simultaneous transmissions per inside-global IP address.

If the port numbers run out, PAT will move to the next IP address and try to allocate the original source port again. This continues until all available ports and IP addresses are utilized. If a PAT router run out of addresses, it drops the packet and sends an ICMP Host Unreachable message.

Generally, NAT/PAT is only practical when relatively few hosts in a stub domain communicate outside of the domain at the same time. In this case, only a small subset of the IP addresses in the own domain must be translated into globally unique IP addresses.

Overloading Example (2)



Prot.	Inside Local	Inside Global	Outside Local	Outside Global
TCP	10.1.1.1:1034	173.3.8.1:1034	65.38.12.9:80	65.38.12.9:80
TCP	10.1.1.2:2138	173.3.8.1:2138	65.38.12.9:80	65.38.12.9:80

Extended Translation Table

(C) Herbert Haas 2005/03/11 16

In this example both inside hosts (10.1.1.1 and 10.1.1.2) connect to the same outside webserver. The outside local addresses are mostly identical to the outside global addresses, but in some situations we might want to translate them also (see next slides for examples).

The dynamic translation table (or *translation matrix*) ages out after some time. The default timeouts are:

Non-DNS UDP	5 minutes	(ip nat translation udp-timeout <seconds>)
DNS	1 minute	(ip nat translation dns-timeout <seconds>)
TCP	24 hours	(ip nat translation tcp-timeout <seconds>)
TCP RST/FIN	1 minute	(ip nat translation finrst-timeout <seconds>)

If overloading is *not* configured the timeout period is 24 hours per default.

```
(ip nat translation timeout <seconds>)
```

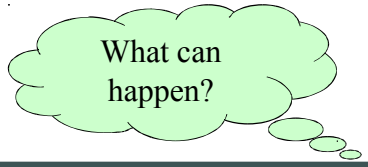
Above Configuration:

```
ip nat pool mypool 173.3.8.1 173.3.8.5 netmask 255.255.255.0
ip nat inside source list 1 pool mypool overload
interface ethernet 0
              ip address 10.1.1.99 255.0.0.0
              ip nat inside
interface serial 0
              ip address 173.3.8.9 255.255.255.0
              ip nat outside
access-list 1 permit 10.0.0.0 0.255.255.255
```

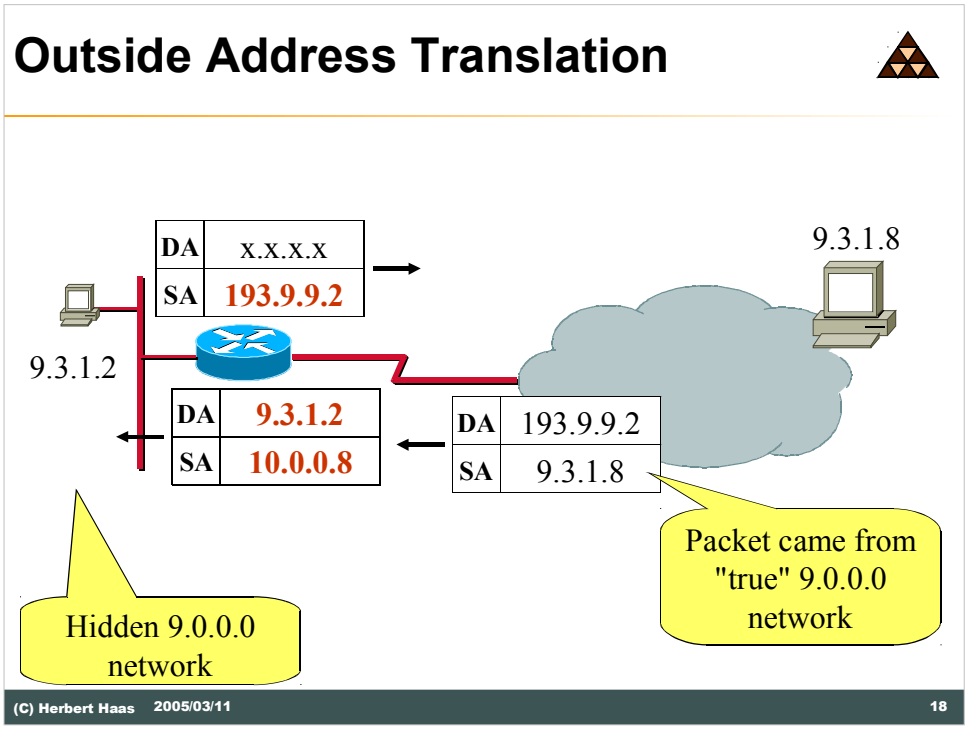
Overlapping Networks

= Same addresses **are used**

locally and ***globally***



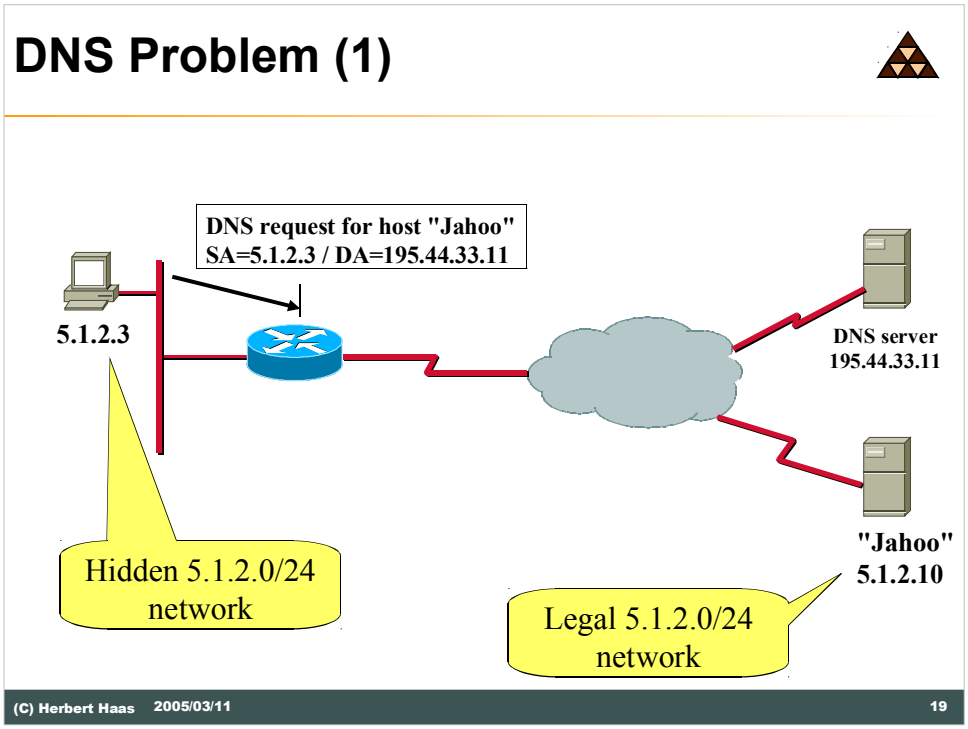
Overlapping networks occur if we use non-legal (not officially assigned) IP addresses that officially belong to another network. We can do that if we use NAT to translate our internal addresses into global ones. However, if we want to communicate with the other network (that use our inside-local addresses as global ones) we must consider some special issues...



First we examine the simple case. Suppose we used a class A network 9.0.0.0 for several years and now we want to give it back to the world (thereby earning a lot of money from our ISP).

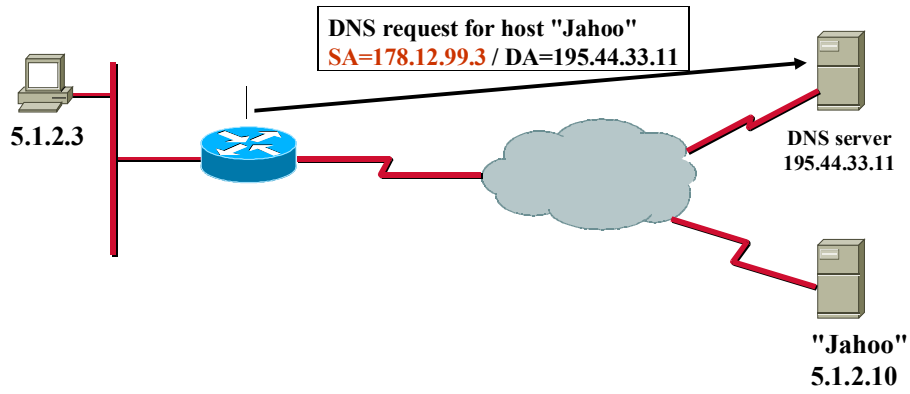
Now we will present our network through NAT to the outside world. Obviously the class A range we had given away will be used by other customers, so incoming packets might have the same source addresses as we still use for our devices. Clearly we should renumber our hosts with RFC1918 private addresses.

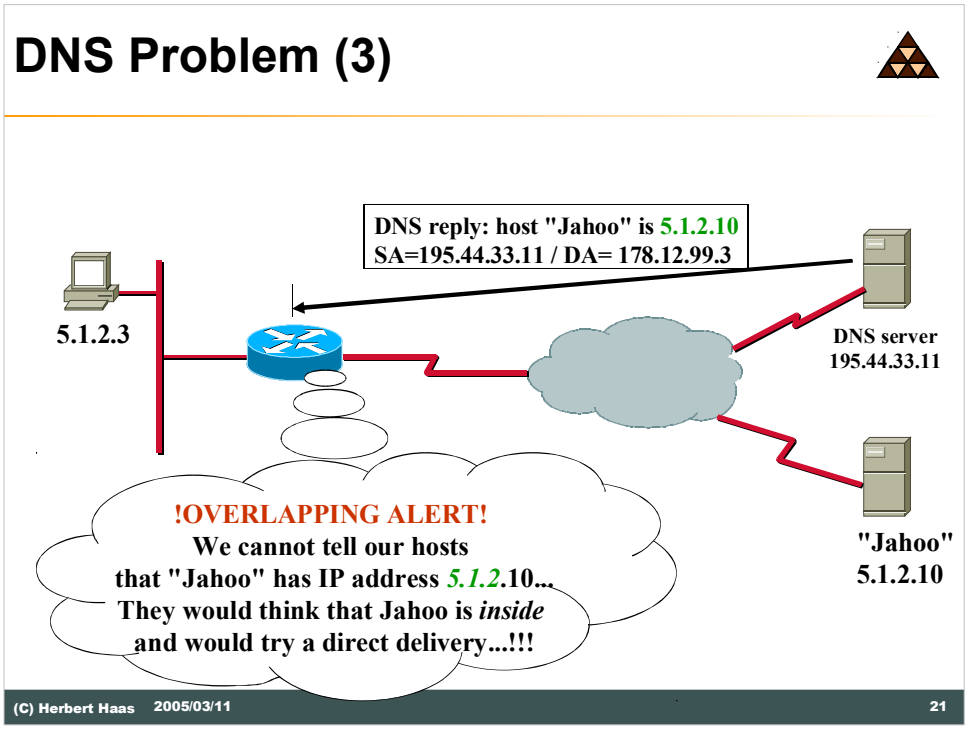
But if we had a big number of hosts we might not want to renumber all devices, instead we will translate the source addresses of incoming packets if they come from the *true* class-A network 9.0.0.0. By changing to an outside-local address, these packets can be routed outside.



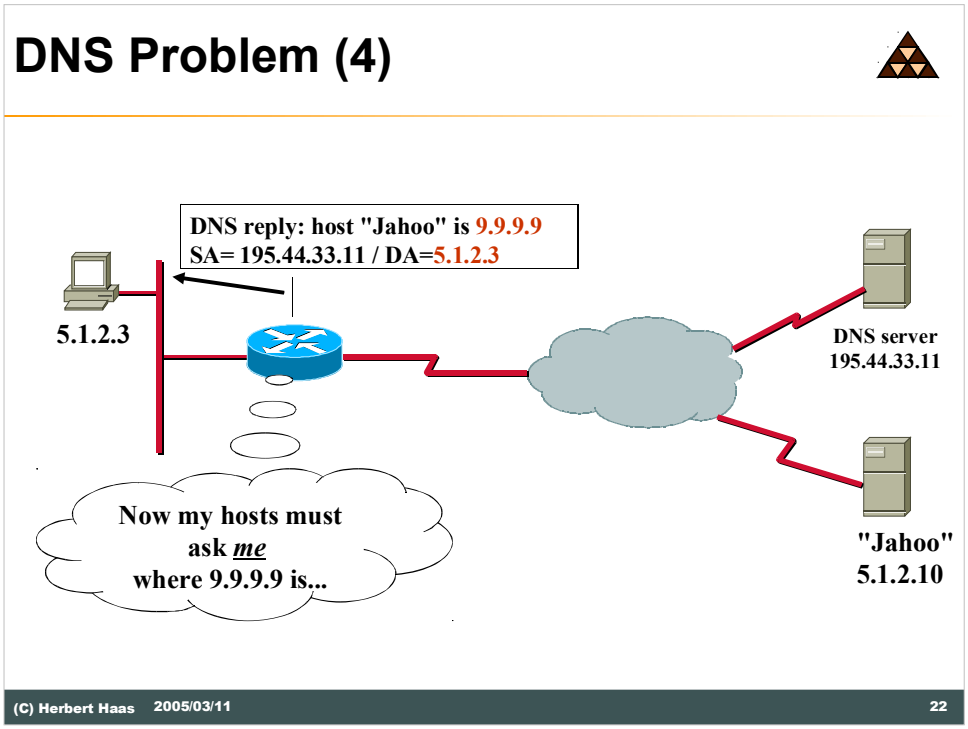
This is a more tricky issue. Usually we do not know IP addresses of outside hosts, rather we ask a DNS server for name resolution.

DNS Problem (2)





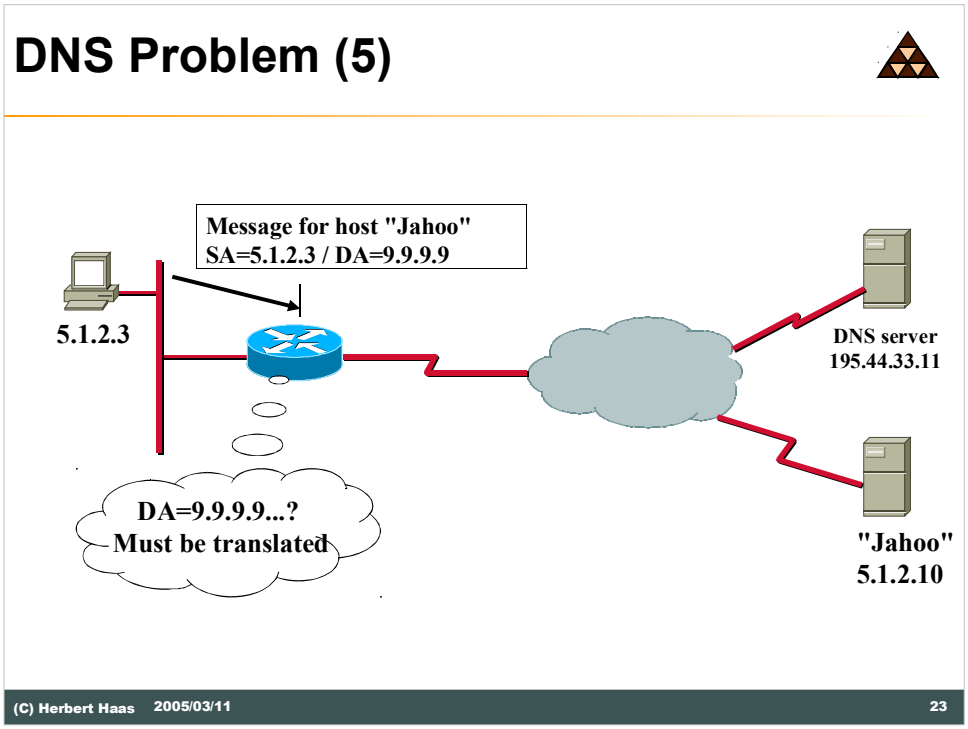
But what, if the DNS server replies an IP address which is supposed to be inside our own network? In this case the NAT router must manipulate the layer-7 DNS information and translate the global-outside addresses.



The router examines every DNS reply, ensuring that the resolved address is not used inside. In such overlapping situations the router will translate the address.

Note:

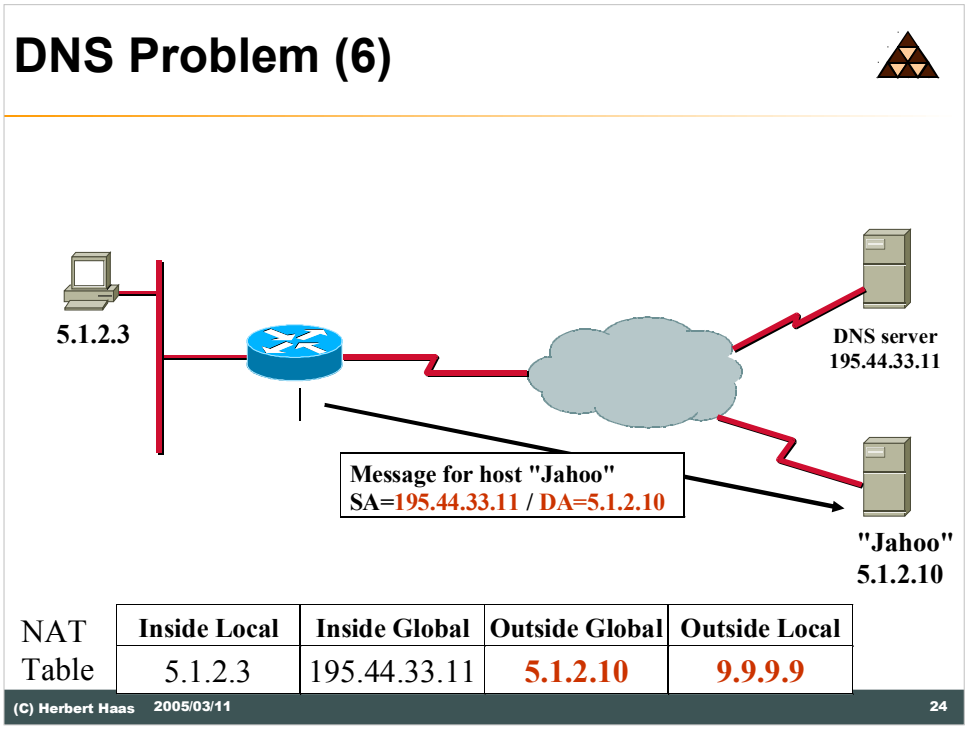
Cisco NAT is able to inspect and perform address translation on *A* (Address) and *PTR* (Pointer) DNS Resource Records.



Of course if the destination address of outgoing packets match a previously introduced outside-local address, it must be translated into a outside-global address.

The same performance is done in a converse situation where the DNS server is inside and a DNS request is sent by an outside host. If the name resolution result in an inside local address the NAT router has to translate this address.

NOTE: Cisco IOS does *not* translate addresses inside DNS zone transfers.



To prepare our router for overlapping addresses we use either a static or a dynamic configuration.

Static: (rest is similar as in previous examples)

```
ip nat outside source static 5.1.2.10  9.9.9.9
```

Dynamic:

```
ip nat pool insidepool 195.44.33.11 195.44.33.13 netmask 255.255.255.0
ip nat pool outsidepool 9.9.9.1  9.9.9.255 prefix-length 24
ip nat inside source list 1 pool insidepool
ip nat outside source list 1 pool outsidepool
!
interface ethernet0
              ip address 5.1.2.99 255.0.0.0
              ip nat inside
!
interface serial0
              ip address 195.44.33.99 255.255.255.0
              ip nat outside
!
access-list 1 permit 5.1.2.0  0.0.0.255
```

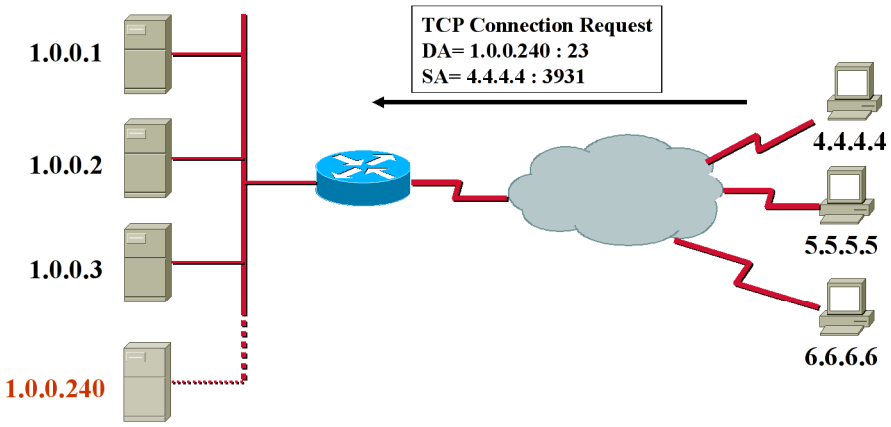
TCP Load Sharing (1)

- **Multiple servers represented by a single inside-global IP address**
  - ***Virtual host* address**
- **New TCP session requests to the Virtual Host are forwarded to one of a group of real hosts**
  - ***Rotary group***

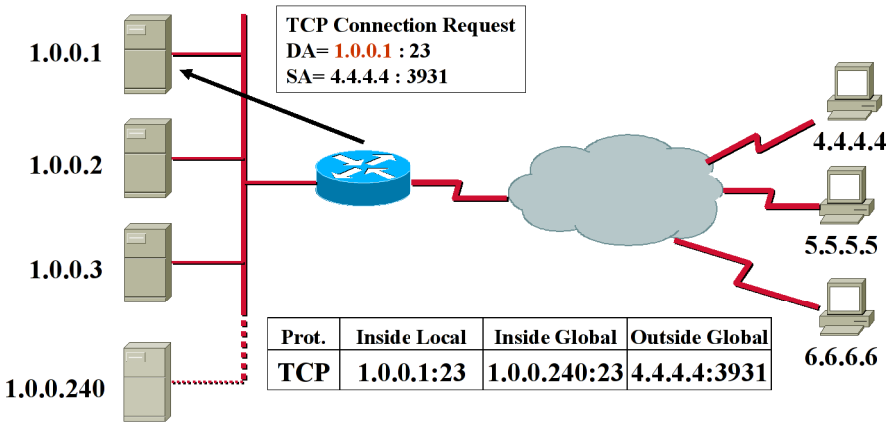
TCP load sharing is an enhanced NAT feature and is used inside the Intranet because this has nothing to do with private address translation. If we want to offer a highly loaded specific service to users, we can employ a NAT router to map a single inside-global address (the virtual host address which is known to the users) to multiple inside-local addresses, each assigned to a real host. Everytime a user connects to the virtual host and wants to establish a session, this session is mapped to one of the real hosts in a round-robin manner. That is why the group of real hosts is called "rotary group".

Note that the NAT router has no idea of the load distribution. Neither the service availability is known to the router!

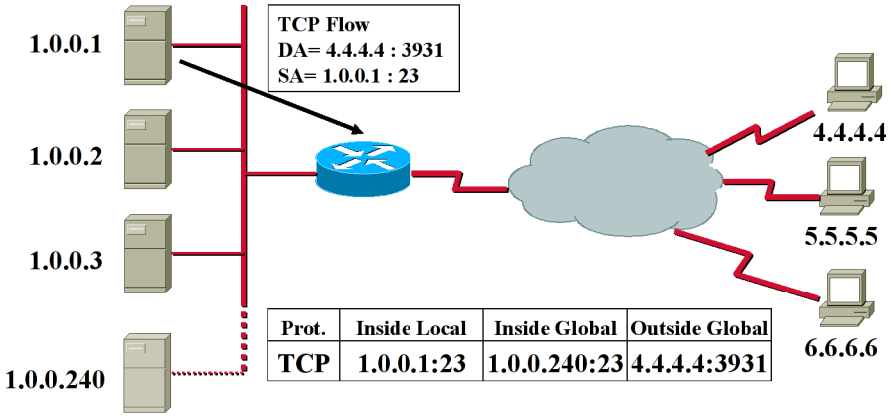
TCP Load Sharing (2)



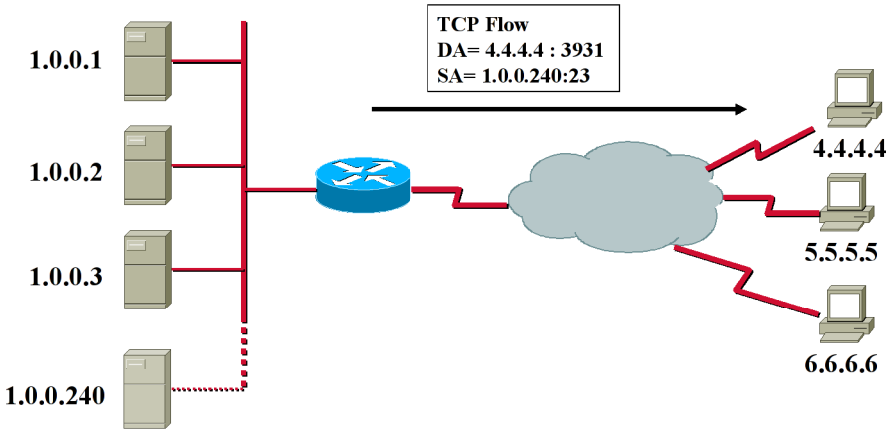
TCP Load Sharing (3)



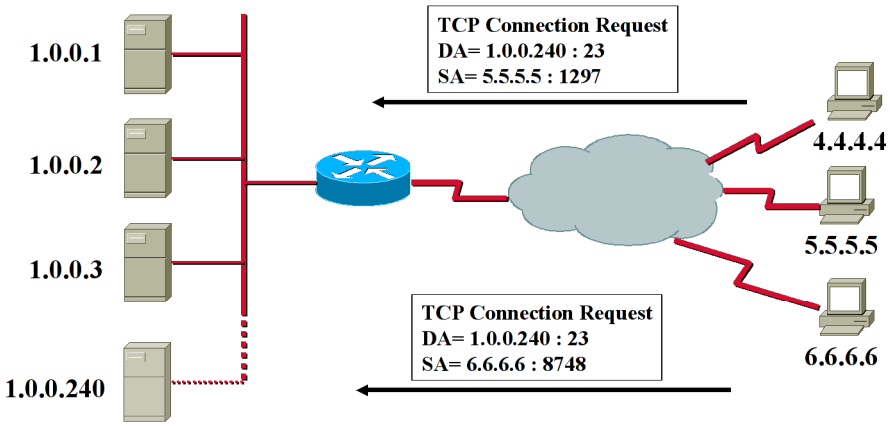
TCP Load Sharing (4)

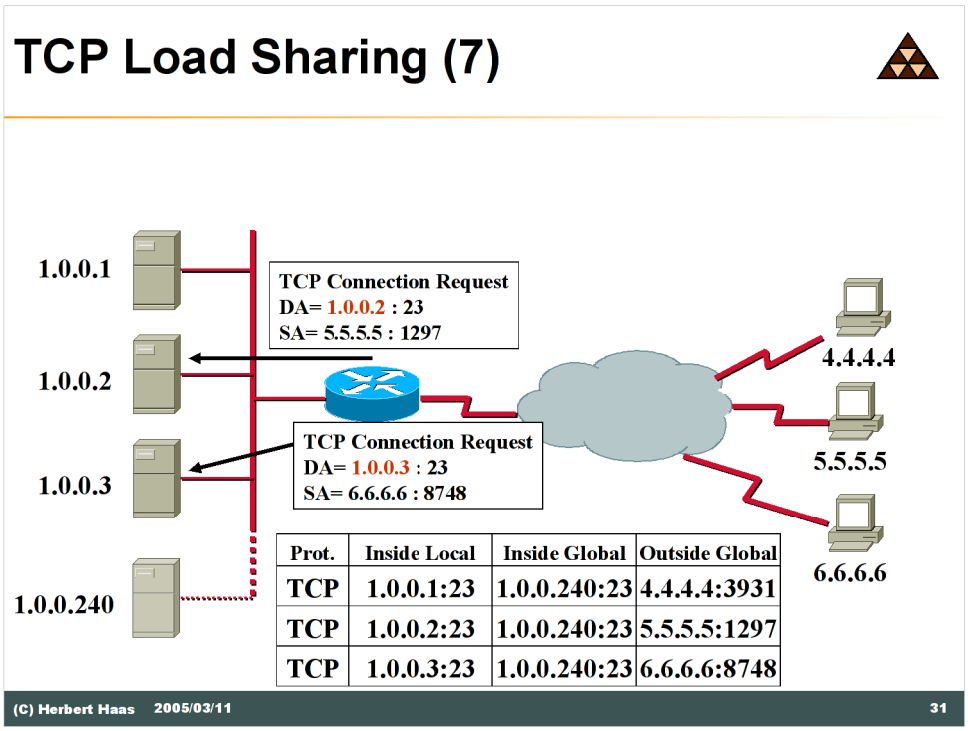


TCP Load Sharing (5)



TCP Load Sharing (6)





Configuration for TCP Load Sharing:

```
ip nat pool myrealhosts 1.0.0.1  1.0.0.3 prefix-length 16 type rotary
ip nat inside destination list 1 pool myrealhosts
!
interface ethernet0
             ip address 1.0.0.99 255.255.0.0
             ip nat inside
!
interface serial0
             ip address 1.1.0.99 255.255.0.0
             ip nat outside
!
access-list 1 permit 1.0.0.240 0.0.0.0
```

NAT and FTP

- **FTP control session negotiates port numbers**
  - **PORT and PASV parameters must be processed by NAT router when doing overloading (ASCII coded!!!)**
- **Non-standard FTP port numbers are mostly supported today**
  - **Cisco:** `ip nat service` **command**

(C) Herbert Haas 2005/03/11 32

IP addresses and port numbers are carried by the FTP PORT and PASV parameters. The problem here is that these addresses are in human readable ASCII format – the address length is variable! This affects the TCP segment length and the SEQ and ACK numbers. These parameters must be transformed for the duration of the connection.

Configuration of a non-standard port number on a Cisco IOS NAT router:

```
ip nat service list <acl> ftp tcp port <port-nr>
```

The access list specifies the name for the inside local address of the FTP server. The port number specifies the non-standard FTP *control* port.

NAT and ICMP

- Many ICMP payloads contain IP headers
  - NAT must translate both addresses and checksum
- PING
  - Echo request & Echo are matched by *ICMP-identifier*
  - Used by NAT instead of port numbers (overloading)
  - If fragmented, only fragment 0 contains this identifier
  - NAT tracks IP identifier for following fragments

(C) Herbert Haas 2005/03/11 33

Consider a NAT configuration with overloaded addresses (= using an extended translation table). Overloading requires some sort of identifier in order to distribute incoming packets to the corresponding internal hosts. Since ICMP is carried directly within IP, NAT cannot utilize port numbers, but the ICMP identifier is used instead. This is only important for query messages such as PING which uses echo request and echo ICMP messages. Both ICMP message types contain a 16 bit identifier field and a 16 bit sequence number field (according RFC 792 both are only optional but indeed commonly used). Only fragment 0 *creates* the translation entry. If a fragment N with N>0 arrives first at the router, it gets dropped.

NAT and ...

- H.323: **TCP/UDP session bundles, ASN.1 encoded IP addresses in payload**
- NetBIOS over TCP/IP (NBT): **packet header information at inconsistent offsets**
- SNMP: **dynamic NAT makes it impossible to track hosts (traps) over longer periods of time**

Security (1)

- **Usually PAT can be detected**
  - **Typical translation signatures**
- **Local topology cannot be seen outside**
  - **Typically SYN-ACKS from outside are blocked**

Some ISPs do not allow customers to use PAT. The employment of PAT can be detected by looking for translation hints. For example a Linux box typically performs Port Address Translation (PAT) using ports between 61000 and 64000, others use ranges starting above 32000, while a TCP/UDP end-system would start at 1024 for each socket. Additionally, most NAT routers decrement the TTL as packets are routed.

Thus an ISP will see that the source ports are at very high values and the TTL is one less than expected. The TTL problem can be solved with most operating systems but requires some administration skills. In windows the default TTL can be adjusted by modifying the registry while Linux needs a recompilation.

Most *hardware* NAT routers (such as Cisco routers) will *not* change the source port where possible. If it does have to change the source port, it chooses the next free port and so on. In such cases no ISP would suspect NAT...

Security (2)

- Typically prevents attacks like SMURF and WinNuke
  - NAT cannot protect all DoS attacks
- Security requires additional software
  - Mailfilters etc.
- Encrypted L3 payload must not contain address/port information

(C) Herbert Haas 2005/03/11 36

Some NAT routers perform stateful packet inspection (SPI), which allows NAT devices to filter harmful packets such as SYN-floods. SPI is merely a marketing term meaning enhanced firewalling features.

NAT cannot translate payload address information if the payload is encrypted. Secure Socket Layer (SSL) and Secure Shell (SSH) are implemented as encrypted TCP payload but the TCP head is not encrypted. Thus, NAT can deal with SSL and SSH without problems. On the other hand, problems may occur with Kerberos, X-Windows, Session Initiation Protocol (SIP), remote shell (RSH), and others NAT-sensitive protocols.

Drawbacks of NAT

- Translation is ressource intensive (delays)
- Encrypted protocols cannot be translated
- Increased probability of mis-addressing
- Might not support all applications
- Hiding hosts might be a negative effect
- Problems with SNMP, DNS, ...

Ressource demand means, the traffic matrix requires lots of RAM while augmented protocol handling requires CPU power.

Each NAT session consumes about 160 bytes in DRAM (using Cisco IOS). From this we conclude that 10,000 translations would consume 1.6 MB.

Mis-addressing occurs as the administrator is responsible for a proper NAT configuration.

SNMP traffic is not supported by Cisco IOS NAT because of the MIB-dependent style of SNMP packets.

Configuration Commands (1)

- **Declare interfaces to be inside/outside**

```
ip nat { inside | outside }
```

- **Define a pool of addresses (global)**

```
ip nat pool <name> <start-ip>
 <end-ip> { netmask <netmask>
 | prefix-length <prefix-
 length> } [ type { rotary } ]
```


Note that a pool of addresses must only be defined for dynamic translation.
If you plan to employ static translation only you can skip the second command.

Configuration Commands (2)

- **Enable translation of** inside source **addresses**

```
ip nat inside source { list <acl> pool <name>
 [overload] | static <local-ip> <global-ip> }
```

- **Enable translation of** inside destination **addresses**

```
ip nat inside destination { list <acl> pool
 <name> | static <global-ip> <local-ip> }
```

- **Enable translation of** outside source **addresses**

```
ip nat outside source { list <acl> pool <name>
 | static <global-ip> <local-ip> }
```

(C) Herbert Haas 2005/03/11 39

Packets from addresses that match those on the simple access-list are translated *dynamically* using the previously defined address pool. The keyword [overload] enables PAT. The access list must permit only those addresses that are to be translated.

Inside destination address translation should use addresses from a previously defined *rotary pool*. A destination address (of an incoming packet) matching the access list will be replaced with an address of the rotary pool in a round-robin manner. See the previous section about TCP load sharing.

Outside source address translation is necessary for overlapping networks. See the corresponding previous section.

Clearing Commands

- Clear **all** dynamic NAT table entries

```
clear ip nat translation *
```

- Clear a **simple** dynamic **inside or inside+outside** translation entry

```
clear ip nat translation inside <global-ip>
 <local-ip> [outside <local-ip global-ip>]
```

- Clear a **simple** dynamic **outside** translation entry

```
clear ip nat translation outside <local-ip>
 <global-ip>
```

- Clear an **extended** dynamic translation entry

```
clear ip nat translation <protocol> inside <global-
  ip> <global-port> <local-ip> <local-port>
  [outside <local-ip> <local-port> <global-ip>
  <global-port>]
```

Further Information

- RFC 1631 (NAT)
- RFC 3022 (Traditional NAT)
- RFC 2694 (DNS ALG)
- RFC 2766 (IPv4 to IPv6 Translation)
- NAT Friendly Application Design Guidelines (Draft)

Summary

- NAT hides inside from outside
- Important to know terms inside/outside versus local/global
- NAT devices must also be able to process L4-L7 headers
- Some protocols might bever be supported (SNMP, NBT, ...)
- Simple TCP load sharing possible
- NAT processing is resource intensive

TODO

- RFC 2766 (IPv4-IPv6 NAT-Protocol Translation)
- NAT with ISP multihoming and routing
- Special NAT situations by example, case studies
- DEBUG commands
- IPSec Tunnel and NAT
- IP Multicast and NAT

...will be covered in future releases!