Network Address Translation

All you want to know about

Reasons for NAT

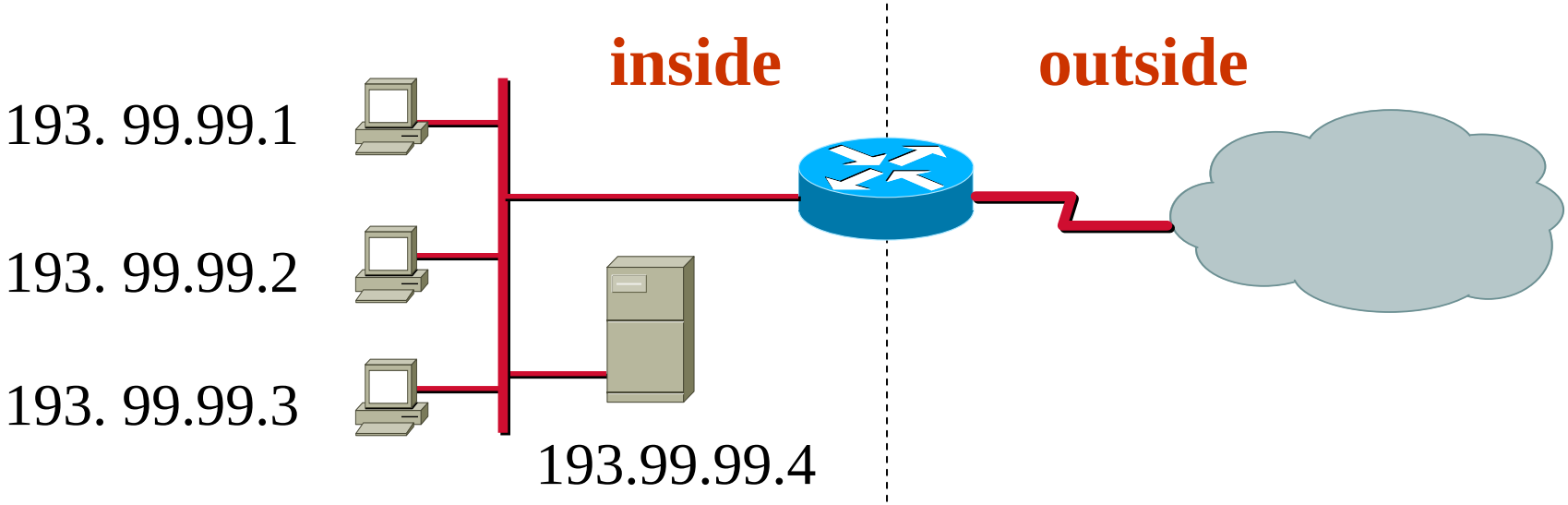
- Mitigate Internet address depletion
- Save global addresses (and money)
- Conserve internal address plan
- TCP load sharing
- Hide internal topology

Credits: The Creators of NAT

Paul Francis

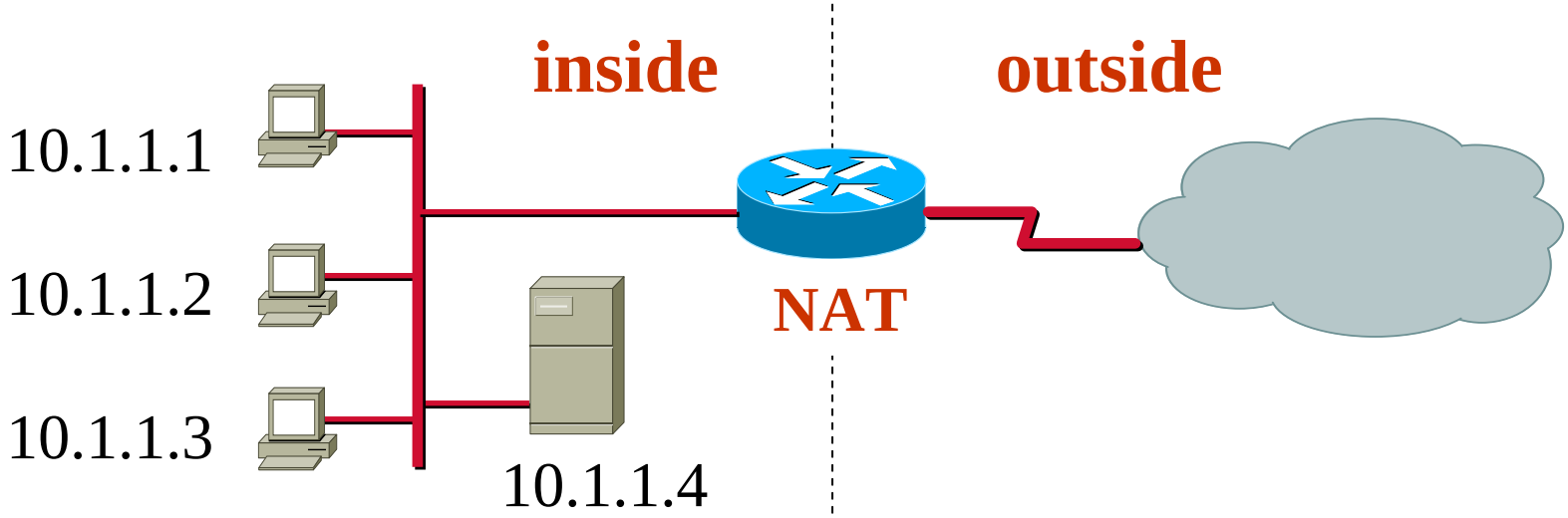
Kjeld Borch Egevang

Terms (1)



Global addresses
(NAT not necessary)

Terms (2)



Local addresses

Terms (3)

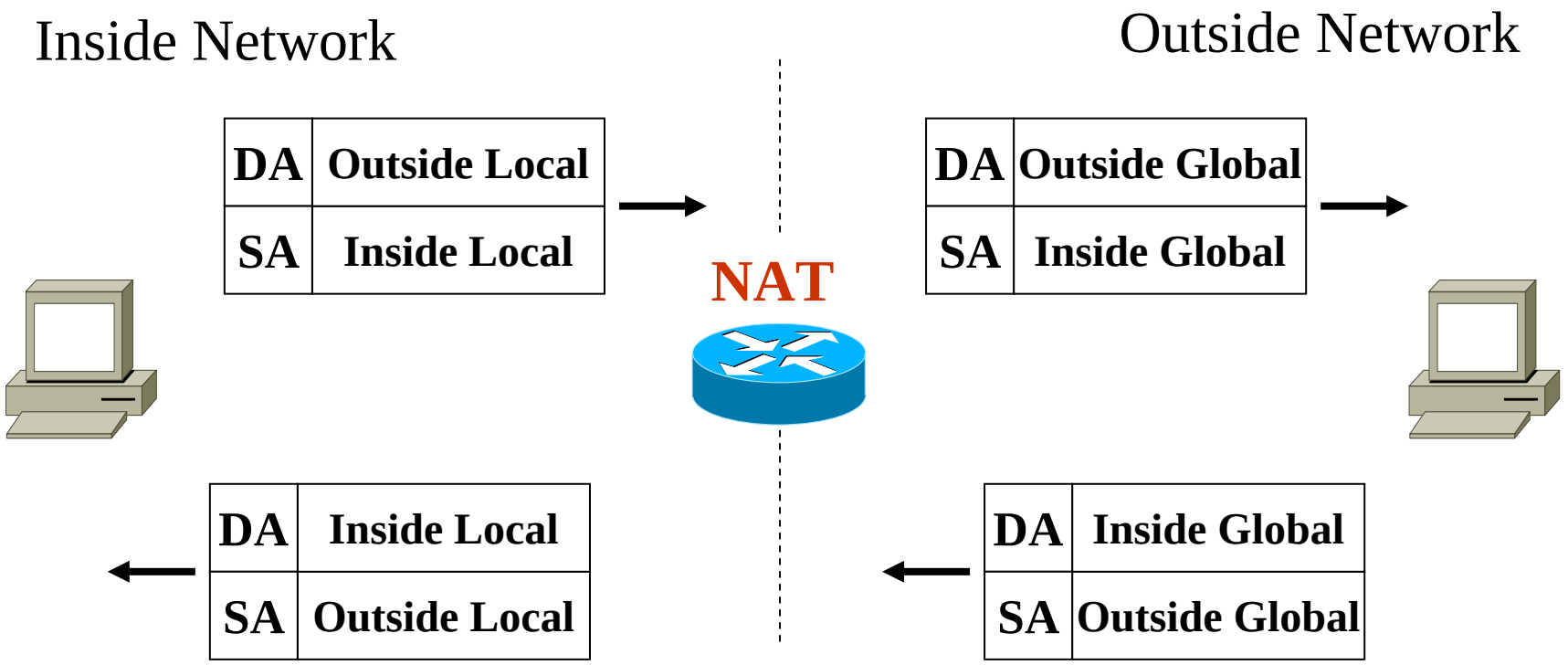
This NAT–Table is maintained inside the router

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

Terms (4)

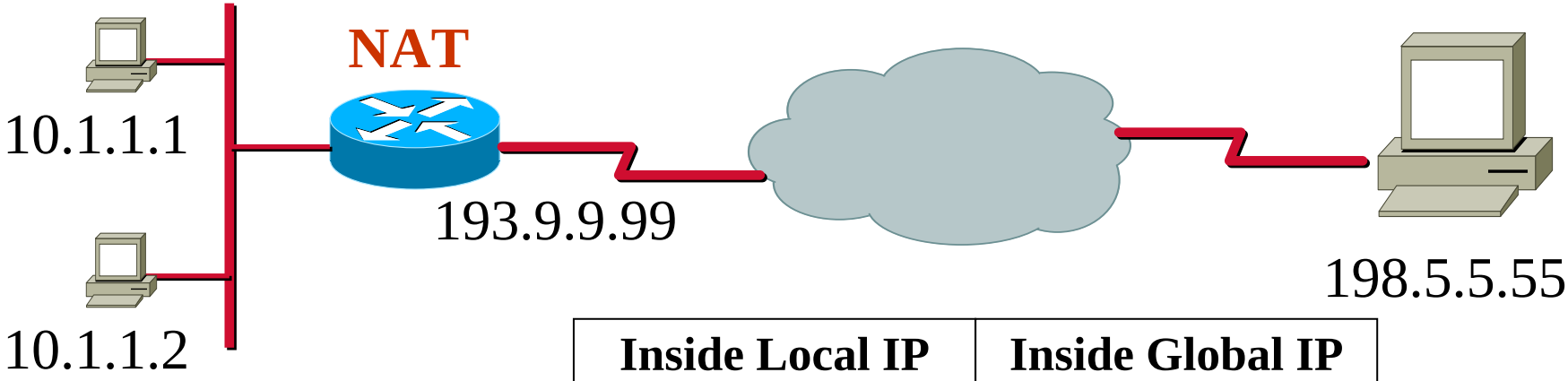
- *Local* versus *global* address
  - Reflects realm of usage (inside or outside)
- *Inside* versus *outside* world
  - Reflects origin

Terms Summary



Basic Principle (1a)

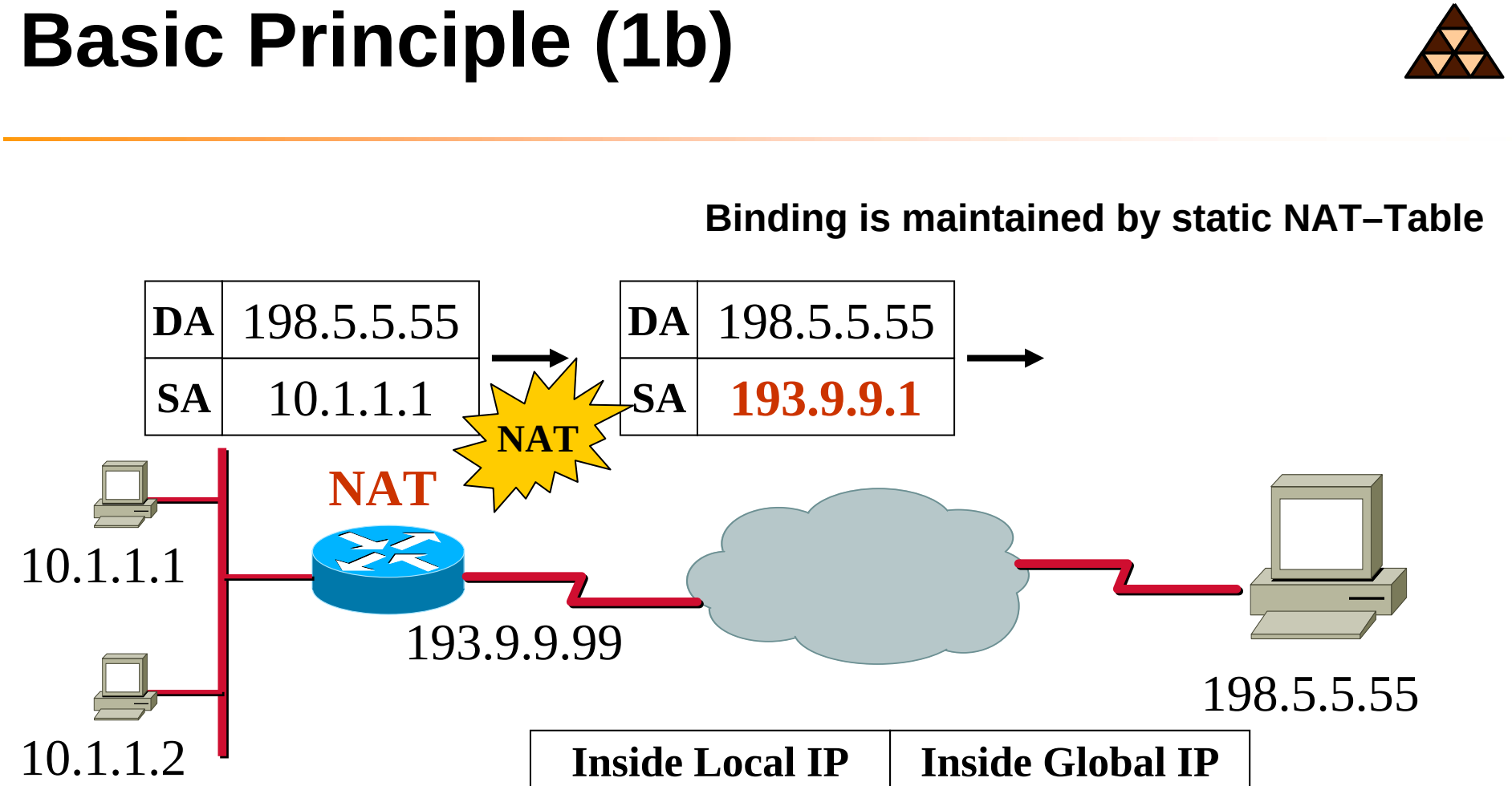
Binding is maintained by static NAT–Table



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
....	....

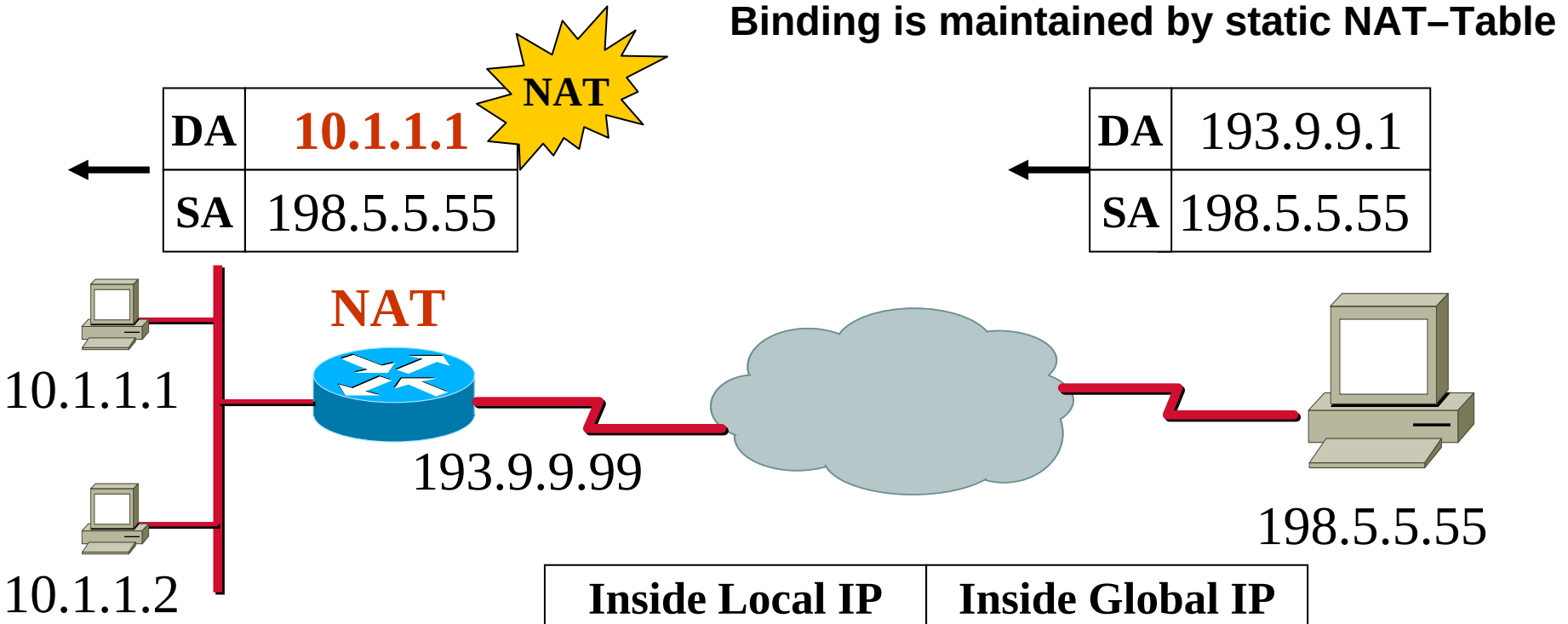
Basic Principle (1b)



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
....	....

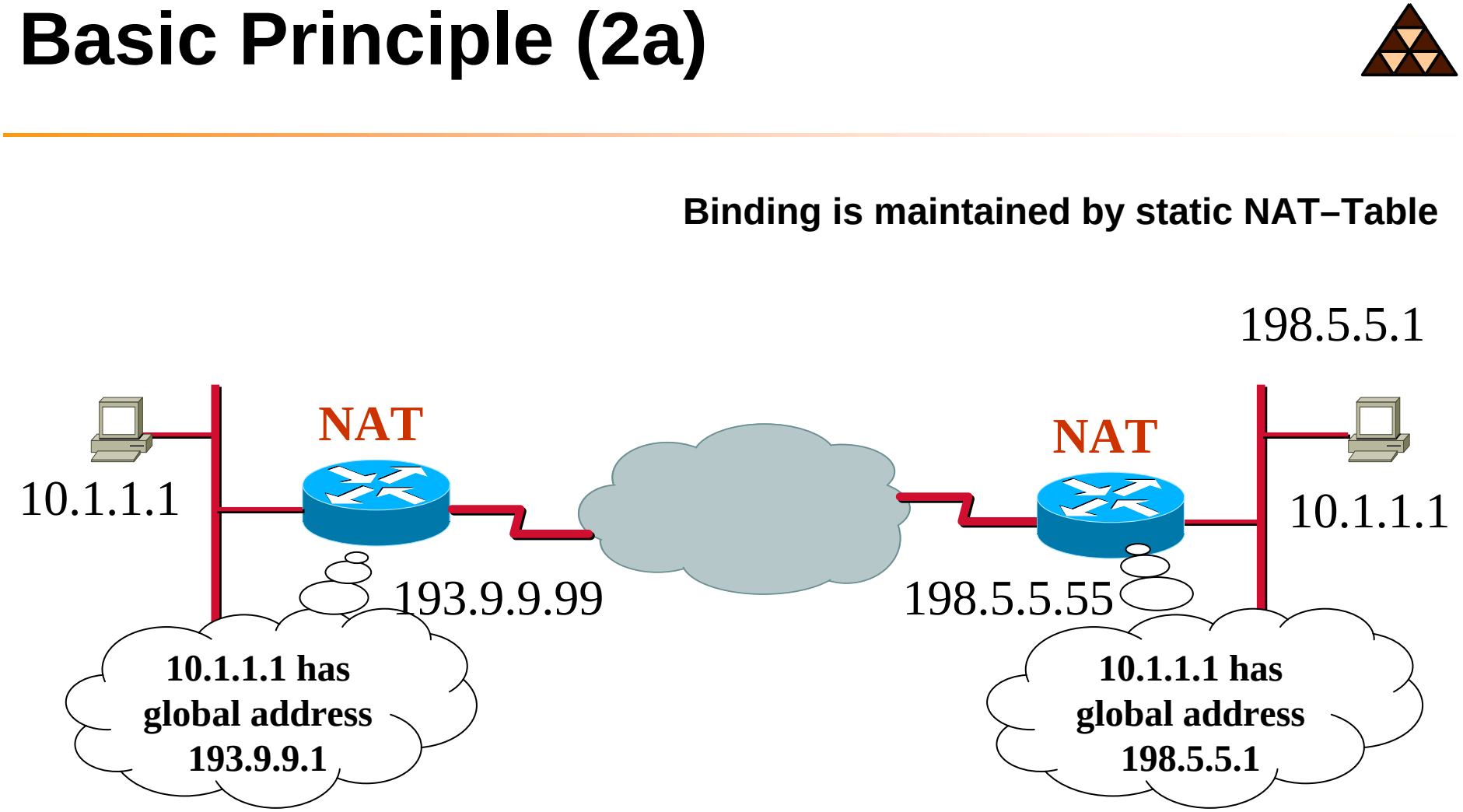
Basic Principle (1c)



Simple NAT Table

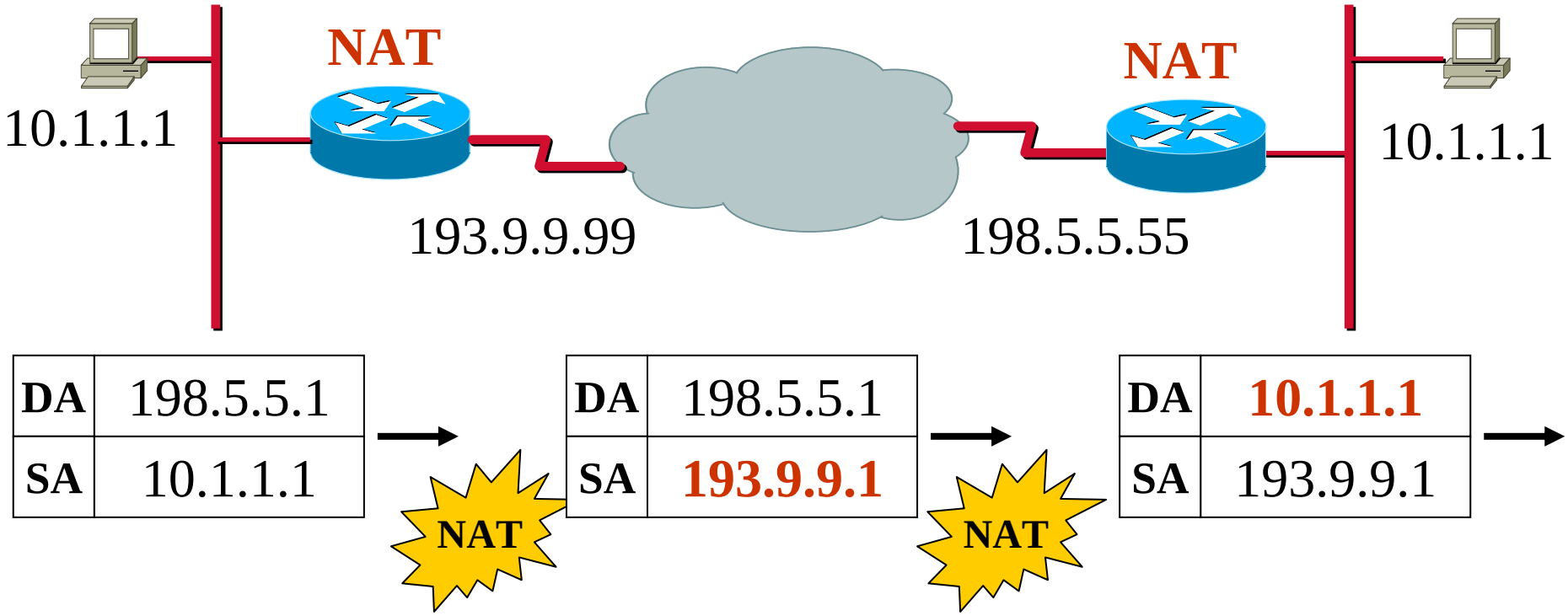
Inside Local IP	Inside Global IP
10.1.1.1	193.9.9.1
10.1.1.2	193.9.9.2
....	....

Basic Principle (2a)



Basic Principle (2b)

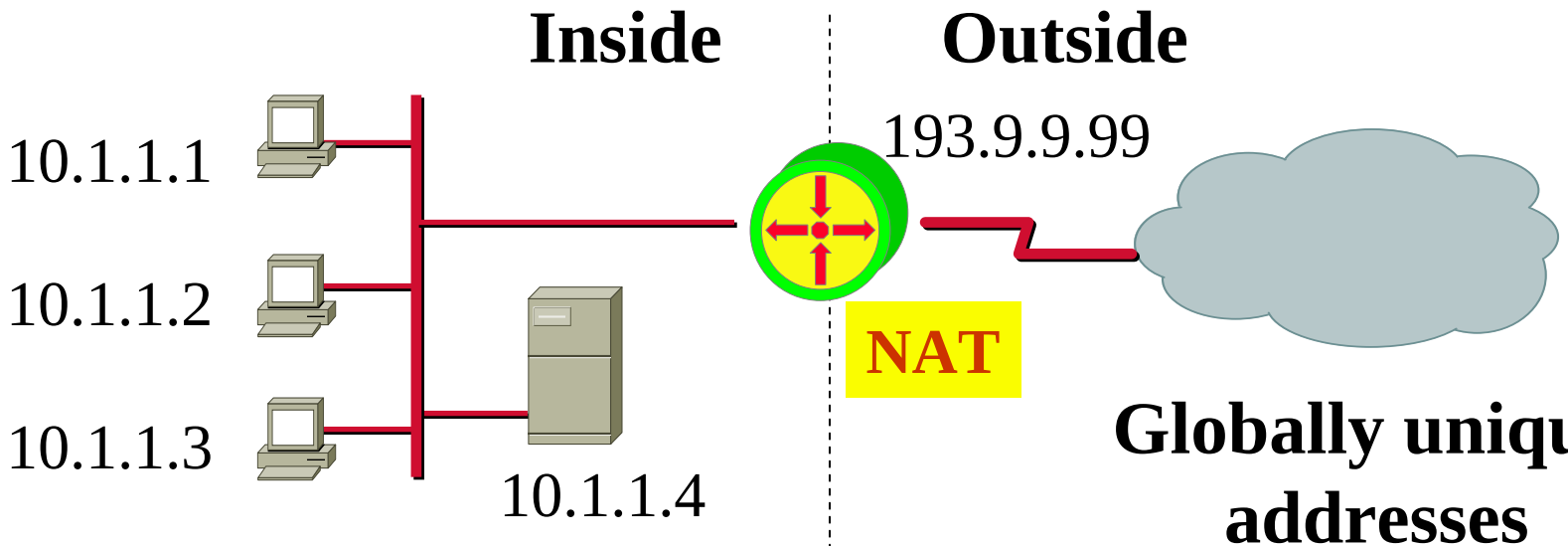
Binding is maintained by static NAT–Table



NAT Binding Possibilities

- **Static ("Fixed Binding")**
  - in case of one-to-one mapping of local to global addresses
- **Dynamic ("Binding on the fly")**
  - in case of sharing a pool of global addresses
  - connections initiated by private hosts are assigned a global address from the pool
  - as long as the private host has an outgoing connection, it can be reached by incoming packets sent to this global address
  - after the connection is terminated (or a timeout is reached), the binding expires, and the address is returned to the pool for reuse
  - is more complex because state must be maintained, and connections must be rejected when the pool is exhausted
  - unlike static binding, dynamic binding enables address reuse, reducing the demand for globally unique addresses.

Scenario Dynamic Binding



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	Currently not possible	
10.1.1.4	Currently not possible	

Binding is maintained by dynamic NAT–Table

Note: a connection state or timer must be maintained per mapping

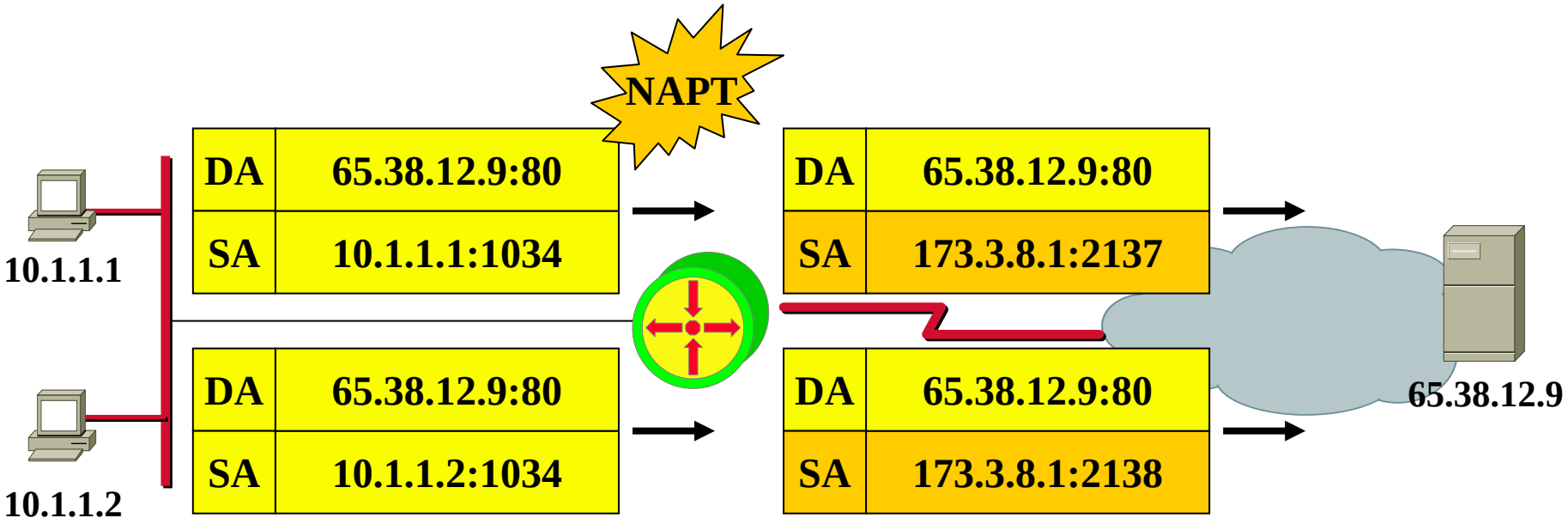
NAT Tasks and Behaviour

- modify IP addresses according to NAT table
- but also must modify the IP checksum and the TCP checksum
  - note: TCP's checksum also covers a pseudo IP header which contains the source and destination address.
- must also look out for ICMP and modify the places where the IP address appears
- there may be other places, where modifications must be done (FTP, NetBIOS over TCP/IP, SNMP, DNS, Kerberos, X-Windows, SIP, H.323, IPsec, IKE…)
- the packet sender and receiver (should) remain unaware that NAT is taking place
- NAT devices were intended to be unmanaged devices that are transparent to end-to-end protocol interaction
- hence no specific interaction is required between the end systems and the NAT device

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"*

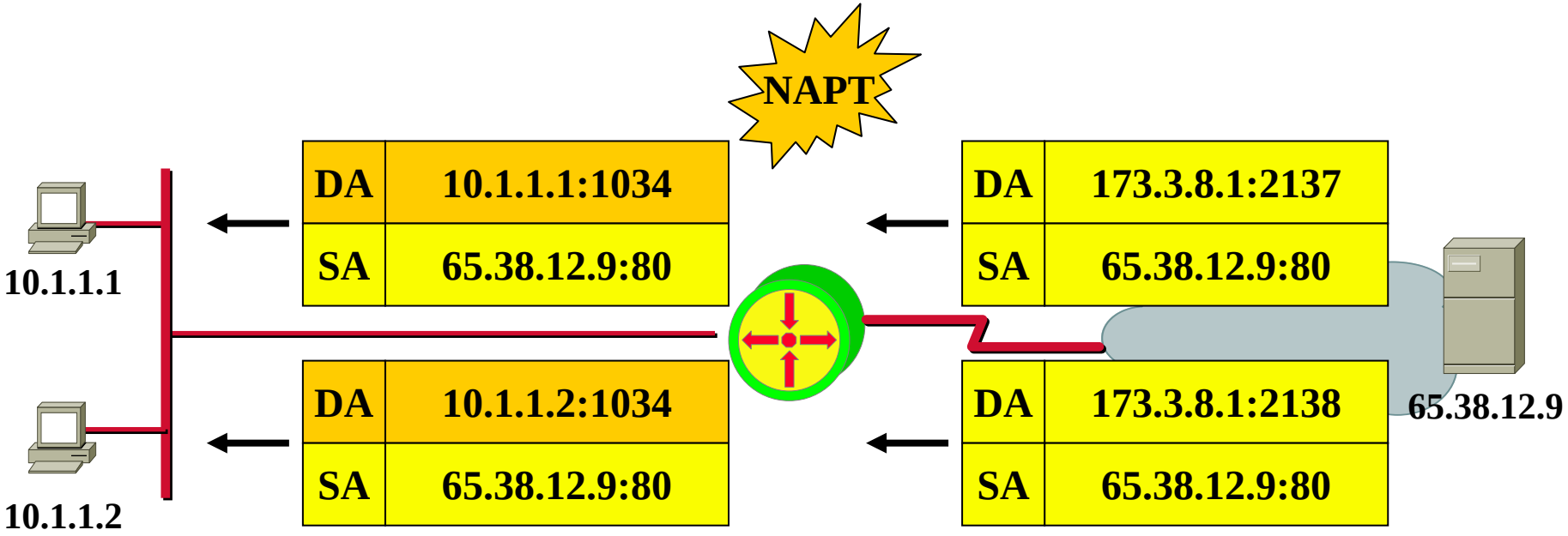
NAPT Example (1)



Prot.	Local	Global
TCP	10.1.1.1:1034	173.3.8.1:2137
TCP	10.1.1.2:1034	173.3.8.1:2138

Extended Translation Table

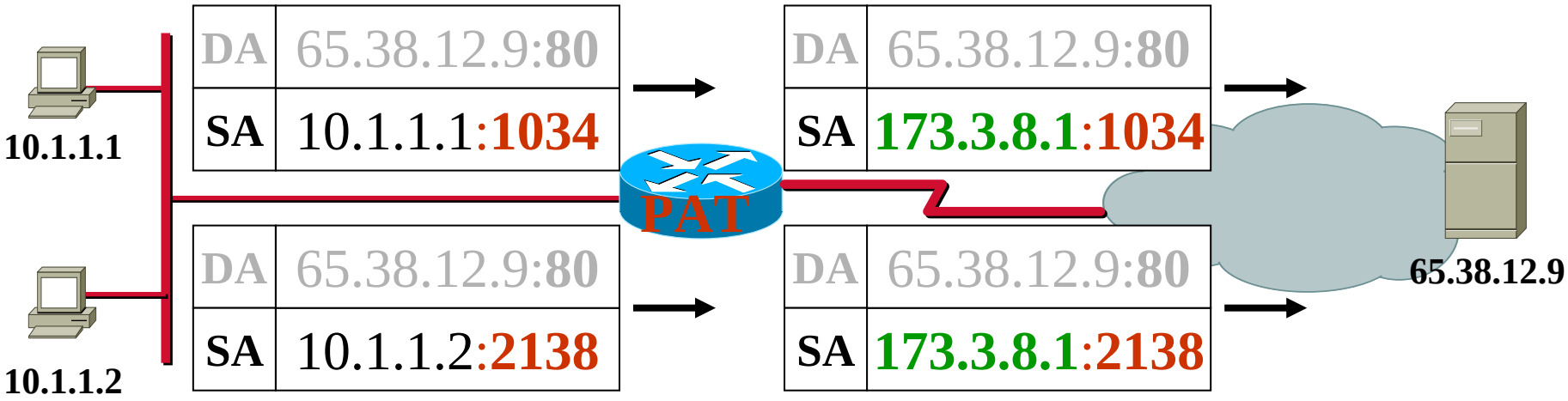
NAPT Example (2)



Prot.	Local	Global
TCP	10.1.1.1:1034	173.3.8.1:2137
TCP	10.1.1.2:1034	173.3.8.1:2138

Extended Translation Table

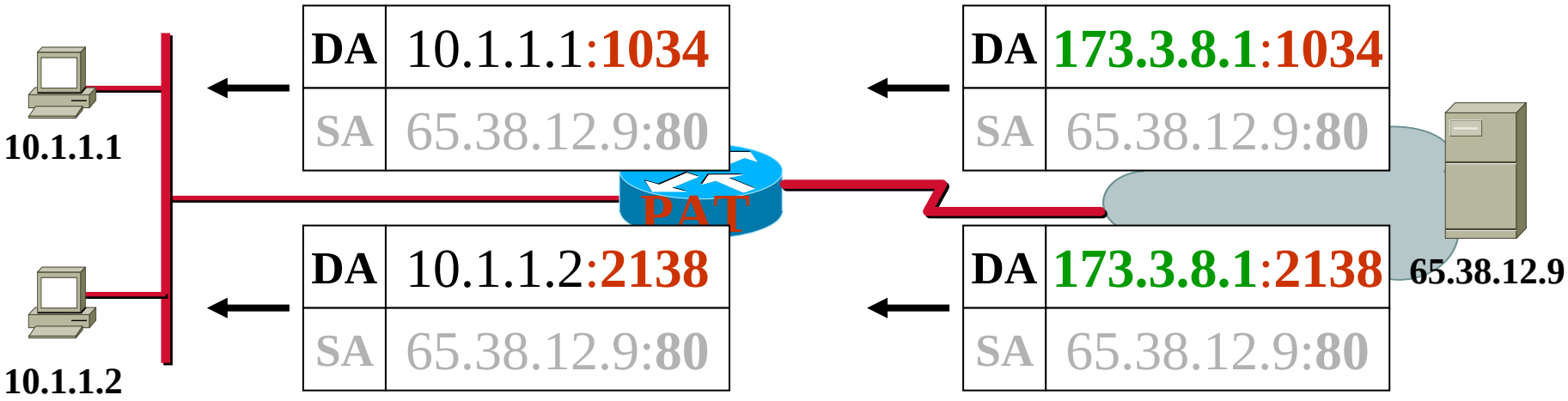
Overloading Example (1)



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

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**

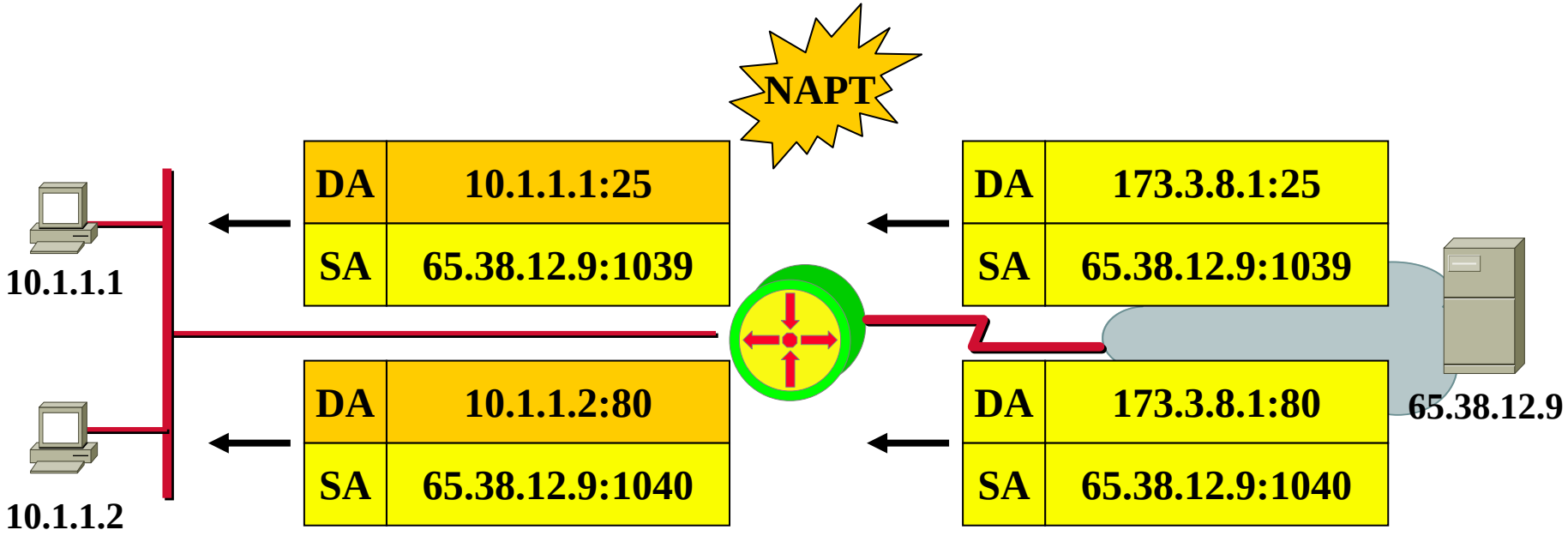
Extended Translation Table

Virtual Server Table

- Problem:
  - **How to reach an inside server from the outside**
  - **NAPT/NAT let IP datagram's (with UDP or TCP segments as payload) from to outside only in if a binding is found**
  - **But server waits for connections from the outside hence cannot install binding in the NAPT/NAT device**

- Solution:
  - **Virtual Server Table**
  - **Creating manually a static binding in the NAPT/NAT device to forward IP datagram's to the real inside server**

Virtual Server Table Example



Prot.	Local	Global
TCP	10.1.1.1:25	173.3.8.1:25
TCP	10.1.1.2:80	173.3.8.1:80

Extended Translation Table

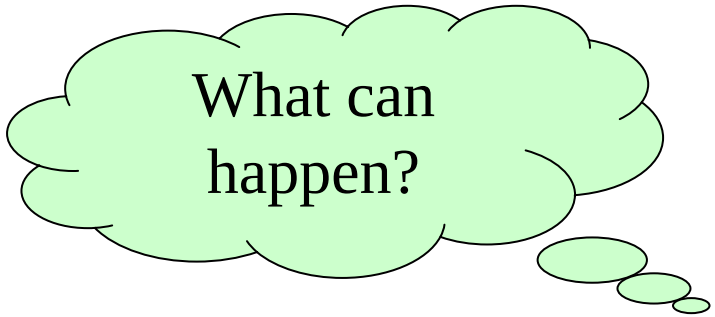
Further Information

- **Internet Protocol Journal**
  - www.cisco.com/ipj
    - Issue Volume 3, Number 4 (December 2000)
    - „The Trouble with NAT“
    - Issue Volume 7, Number 3 (September 2004)
    - „Anatomy (of NAT)“

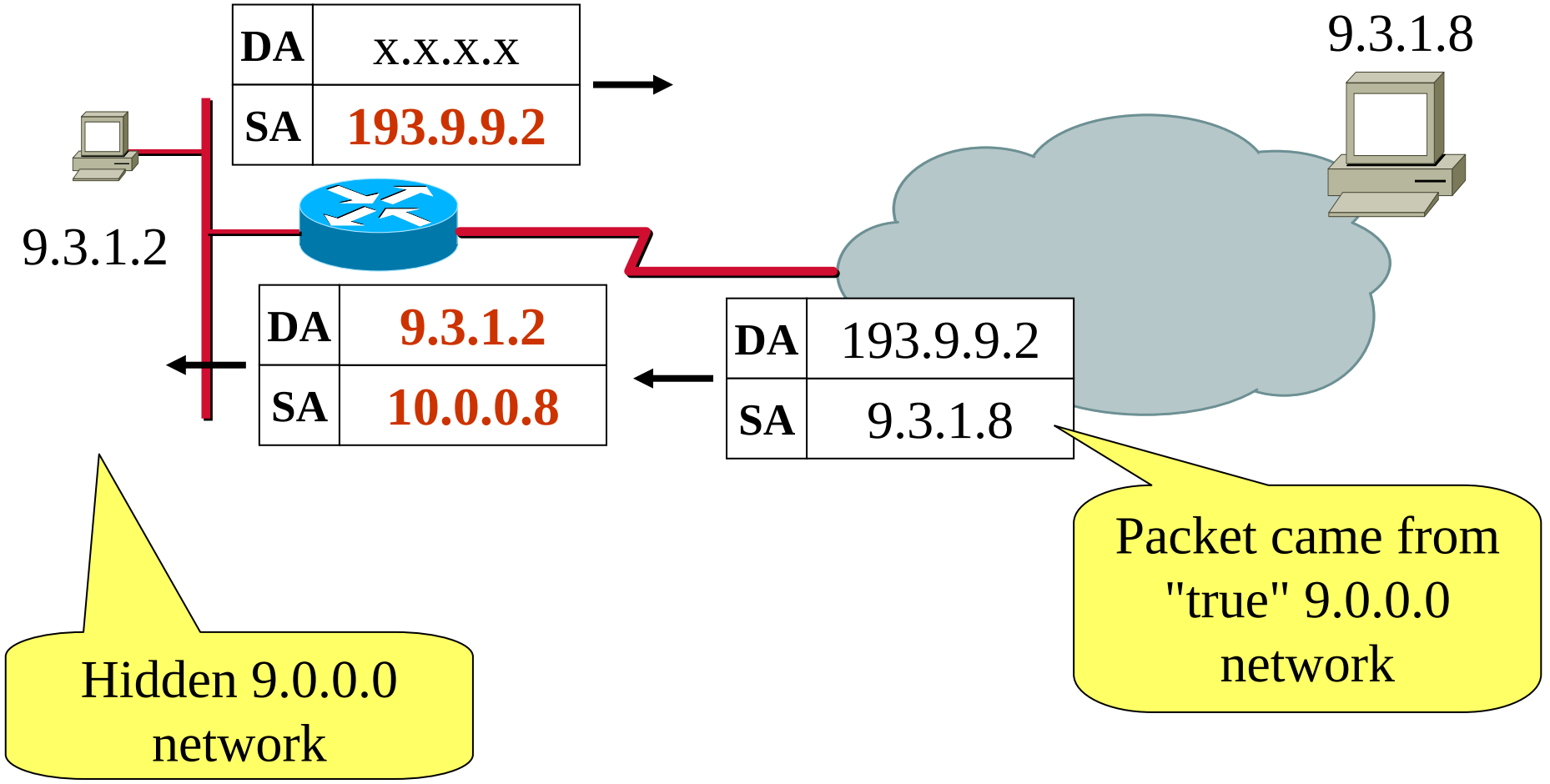
Overlapping Networks

= Same addresses **are used**

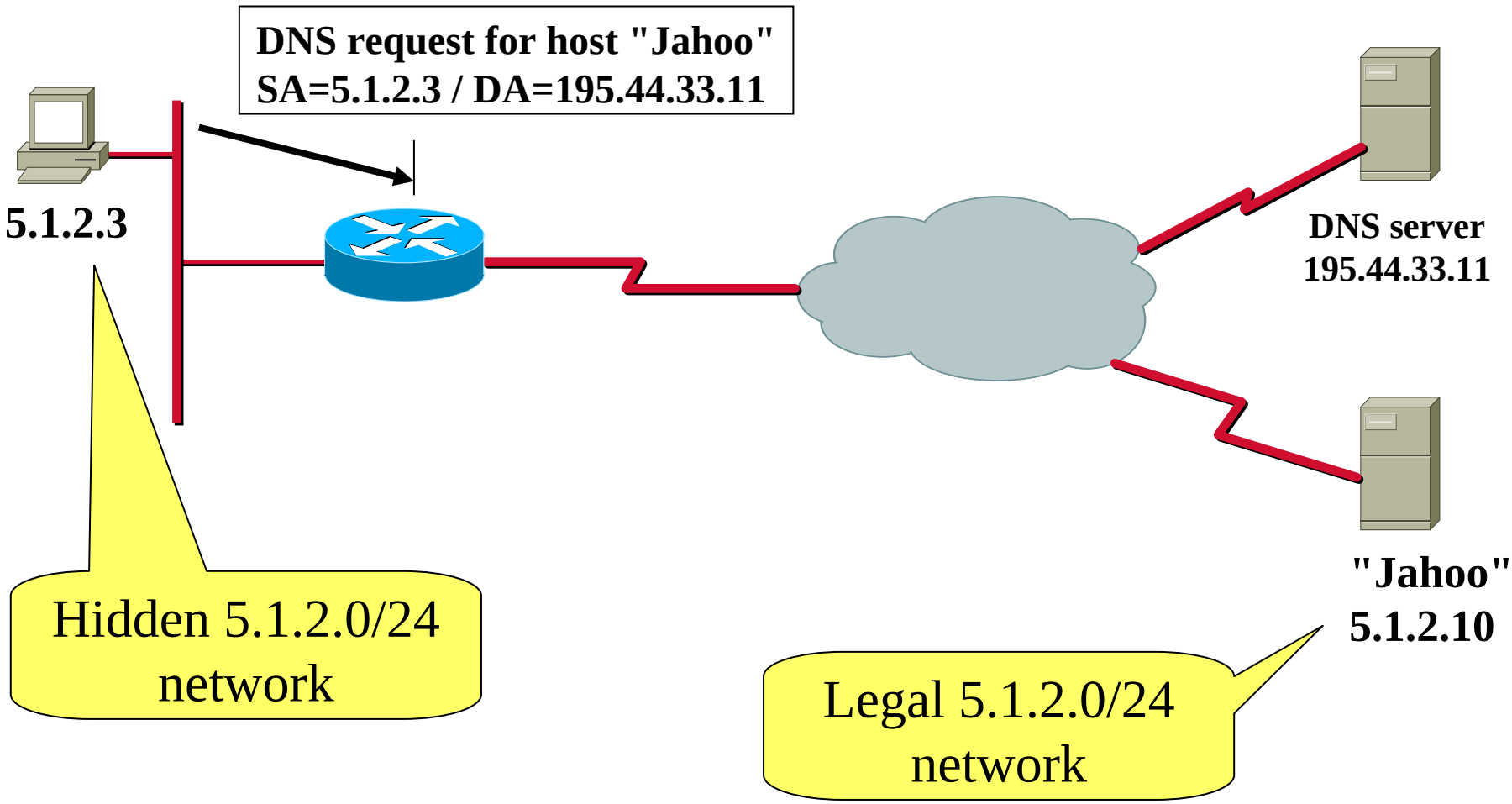
locally and ***globally***



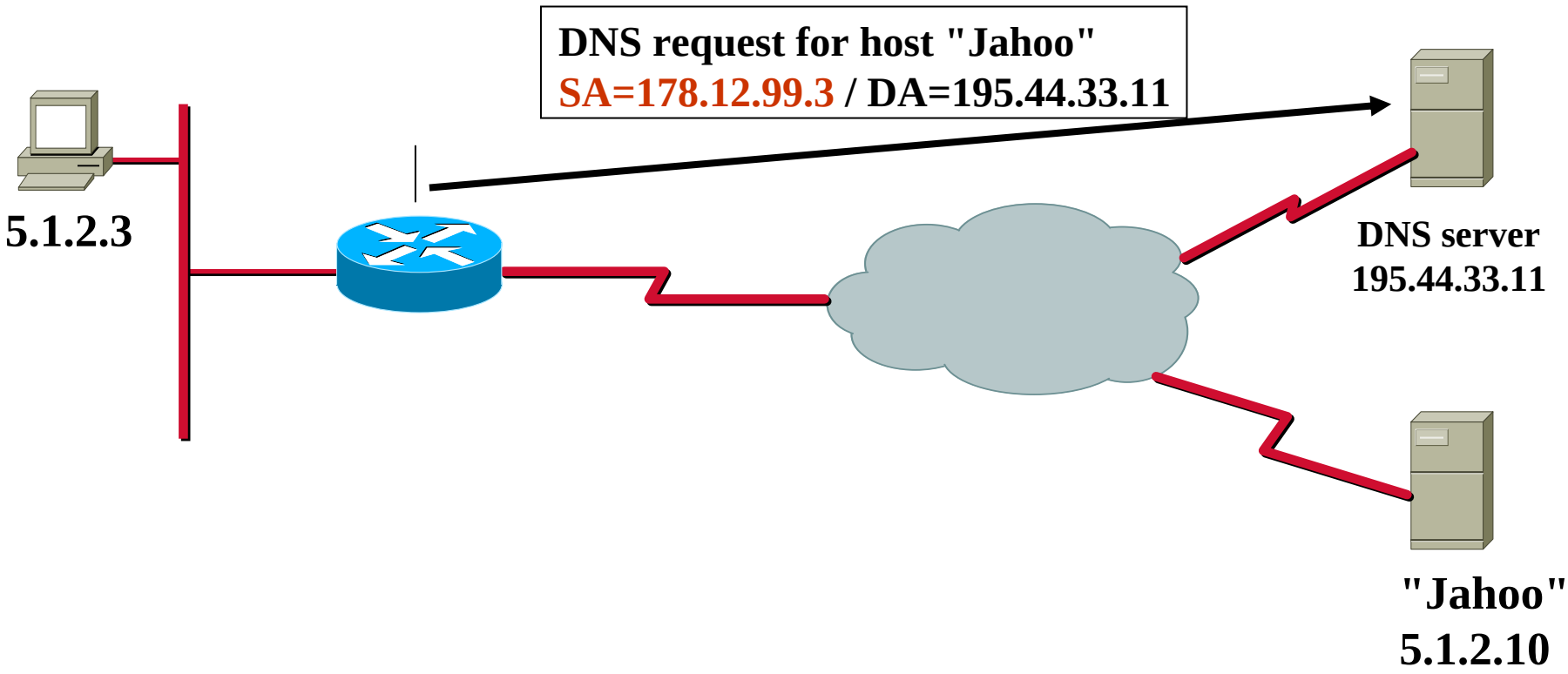
Outside Address Translation



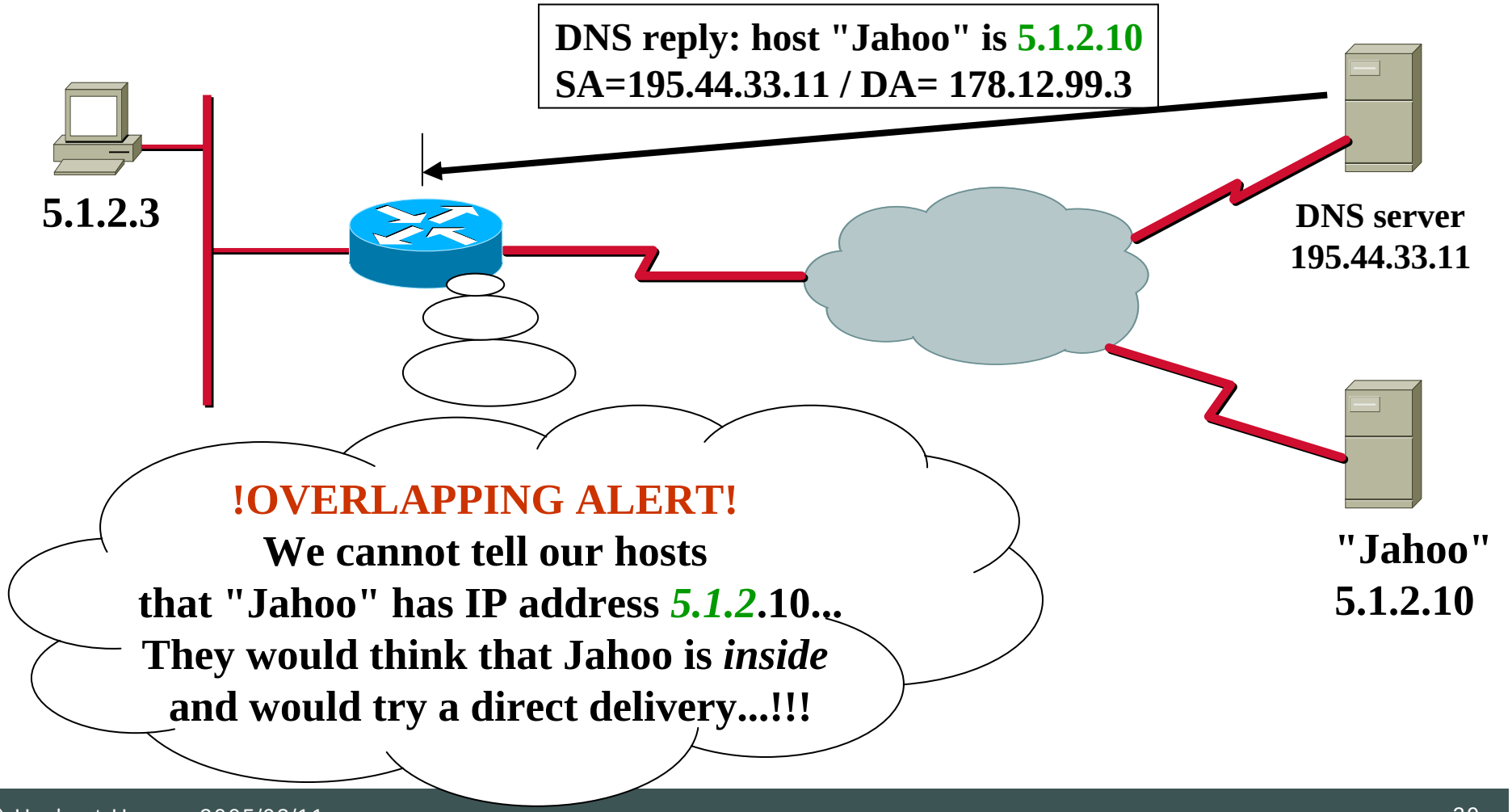
DNS Problem (1)



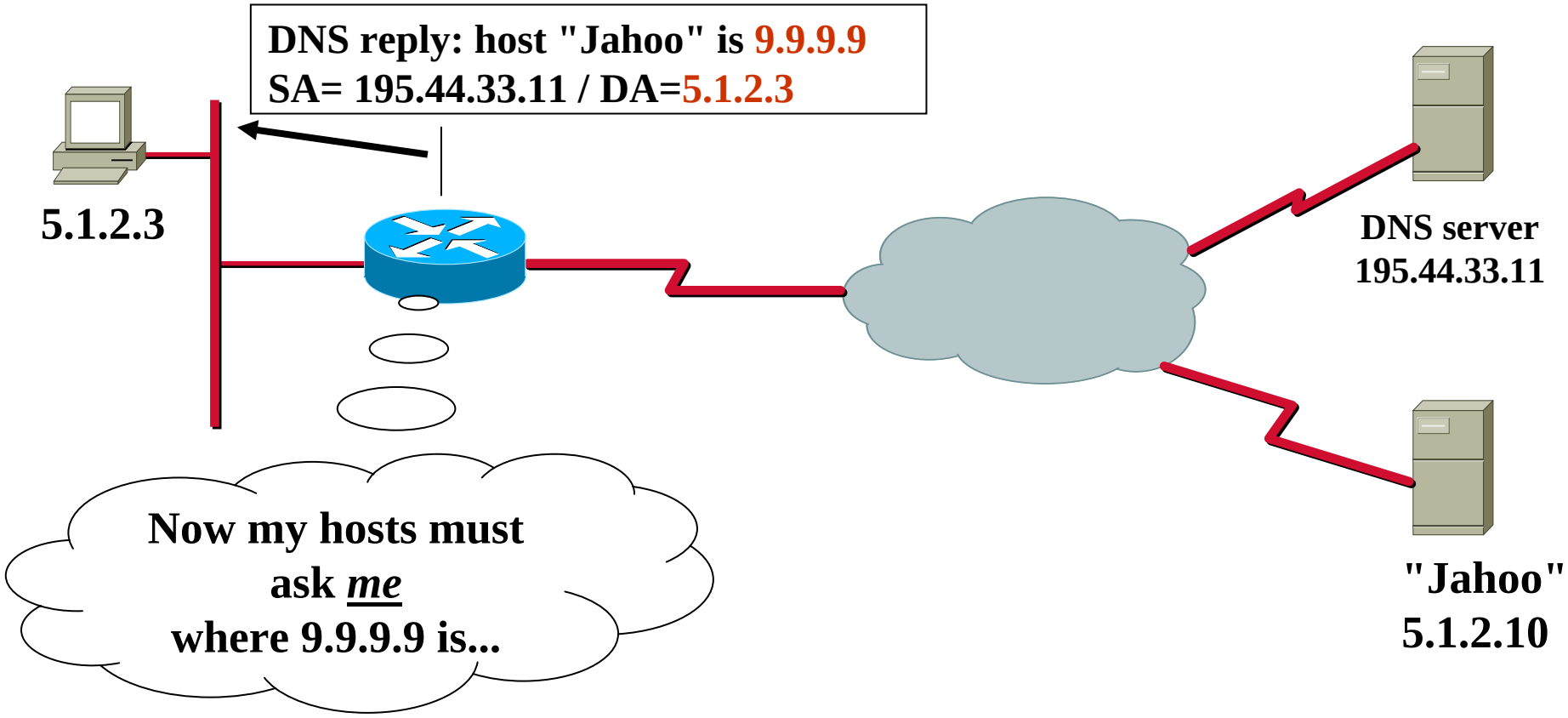
DNS Problem (2)



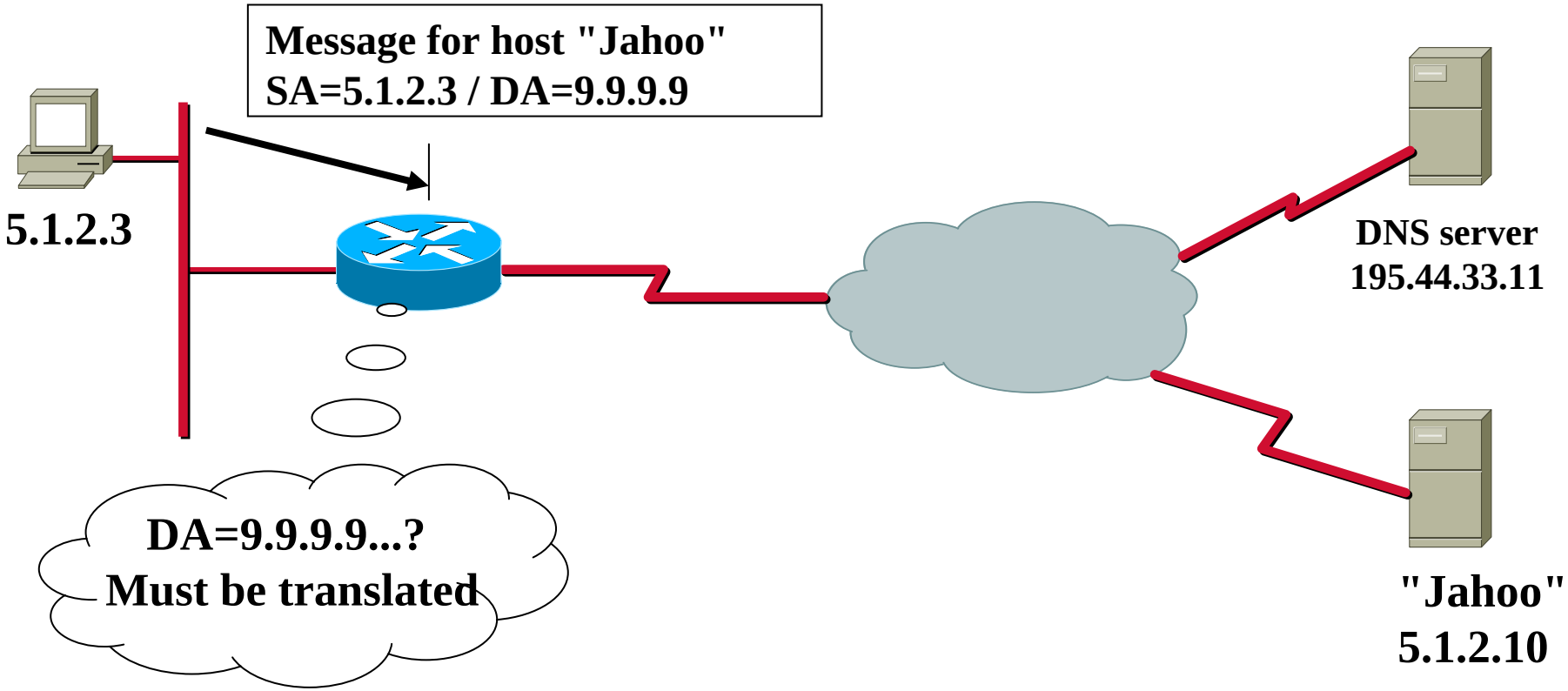
DNS Problem (3)



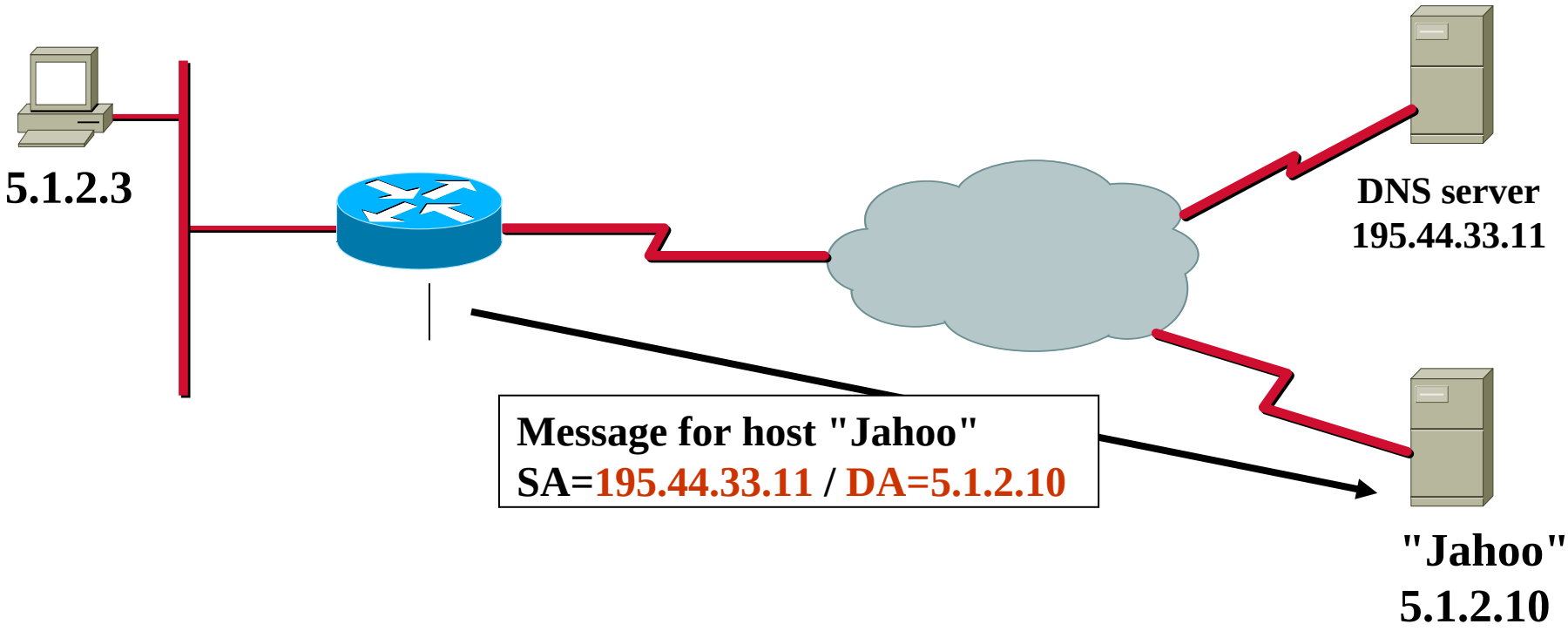
DNS Problem (4)



DNS Problem (5)



DNS Problem (6)



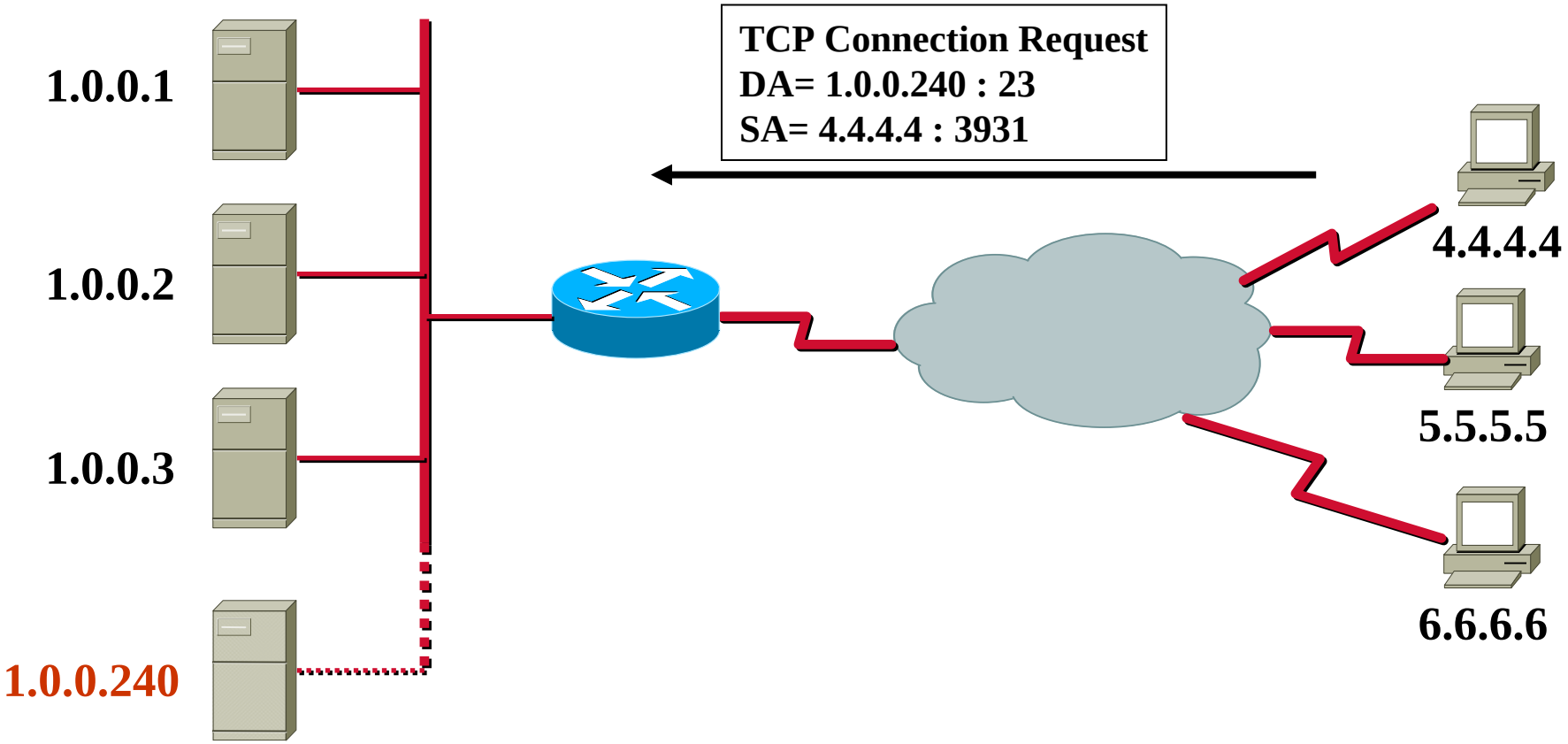
NAT Table

Inside Local	Inside Global	Outside Global	Outside Local
5.1.2.3	195.44.33.11	5.1.2.10	9.9.9.9

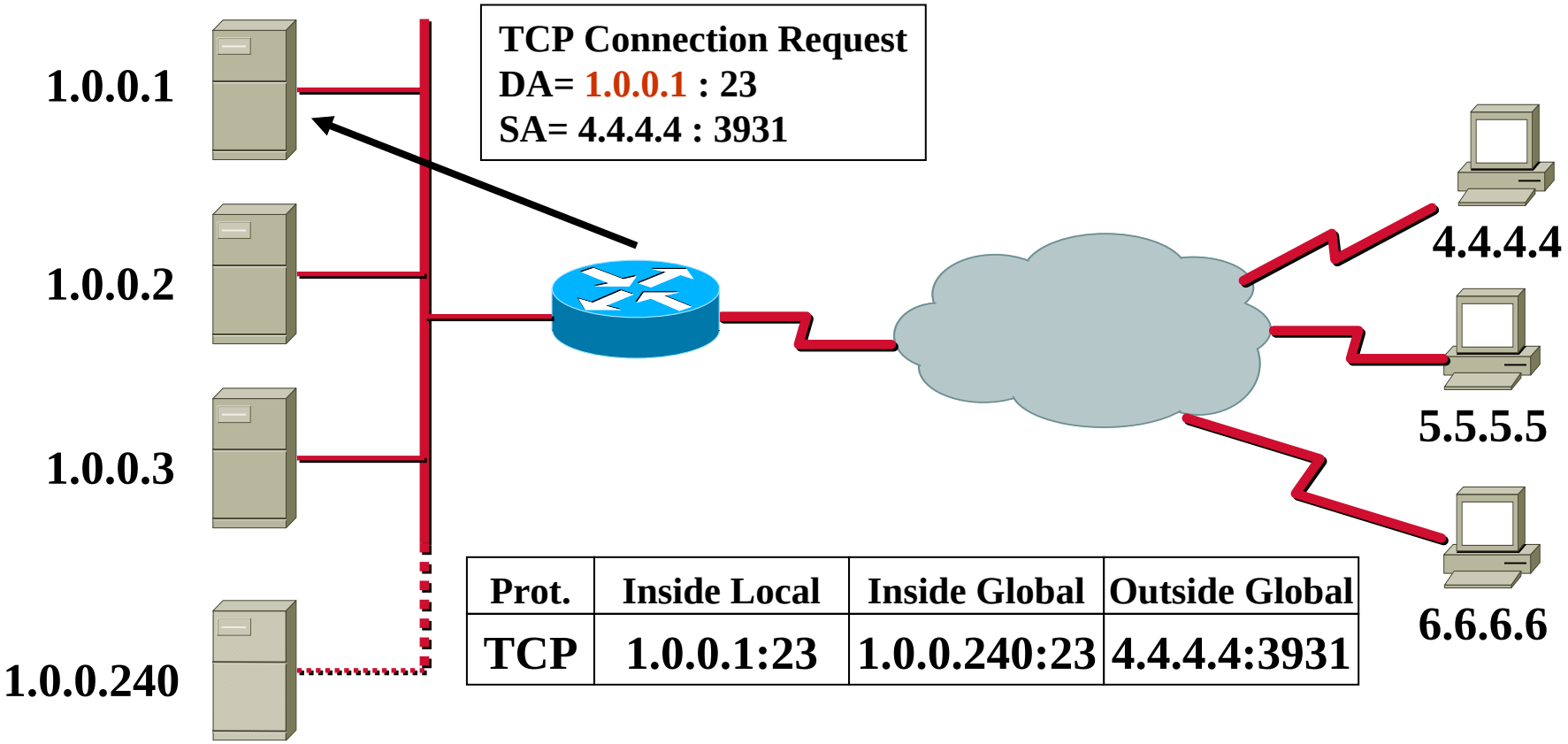
TCP Load Sharing (1)

- **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***

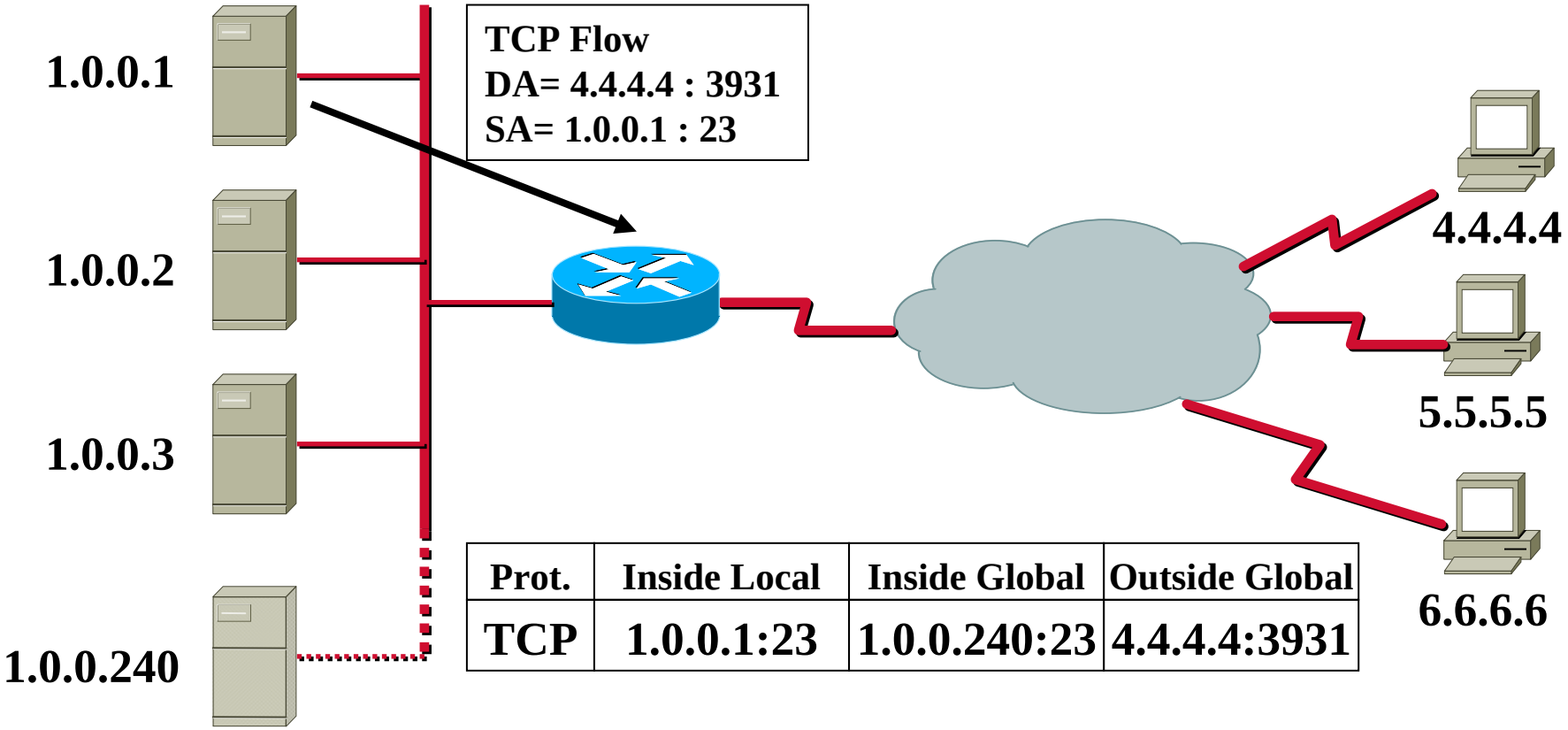
TCP Load Sharing (2)



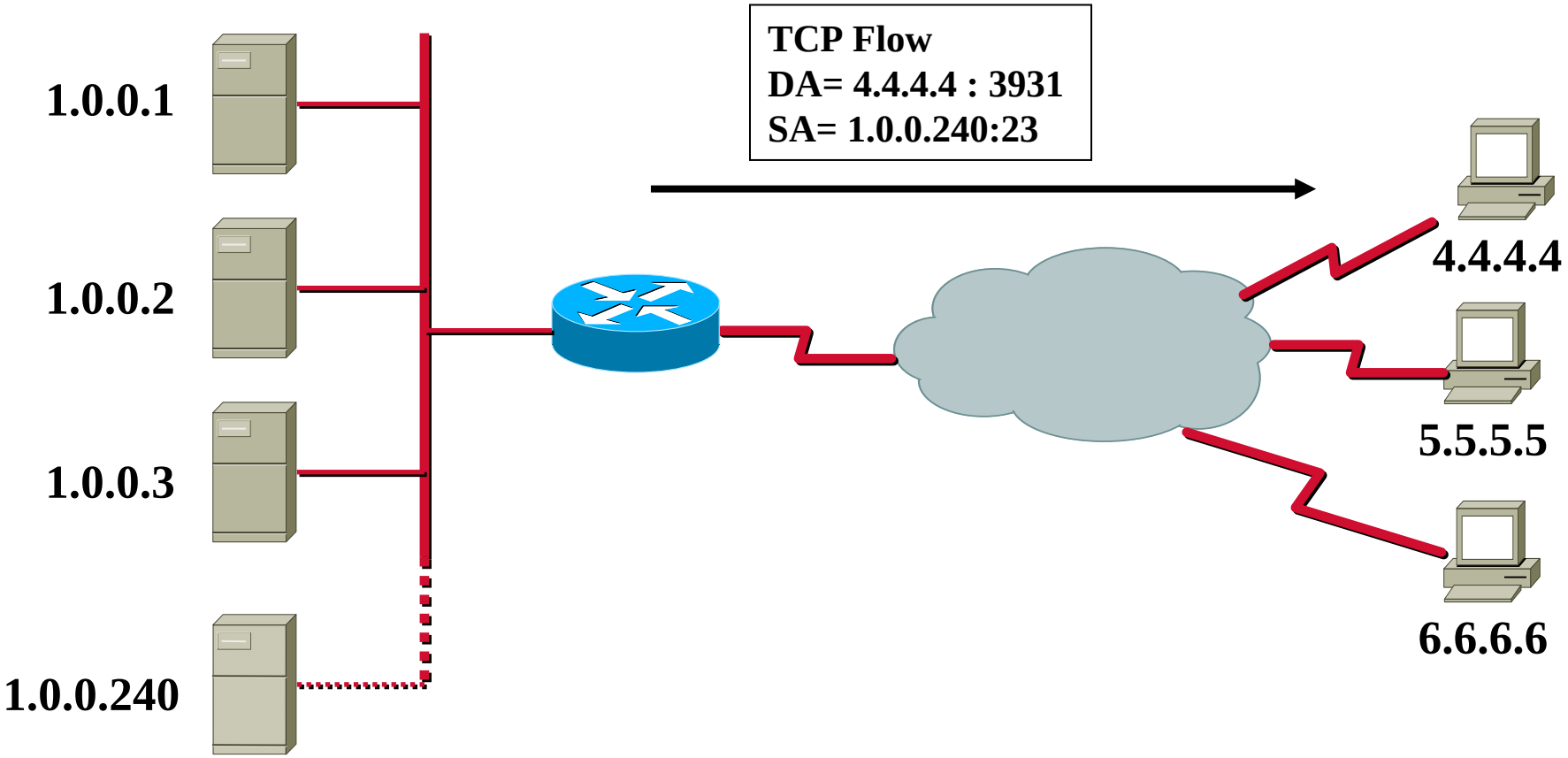
TCP Load Sharing (3)



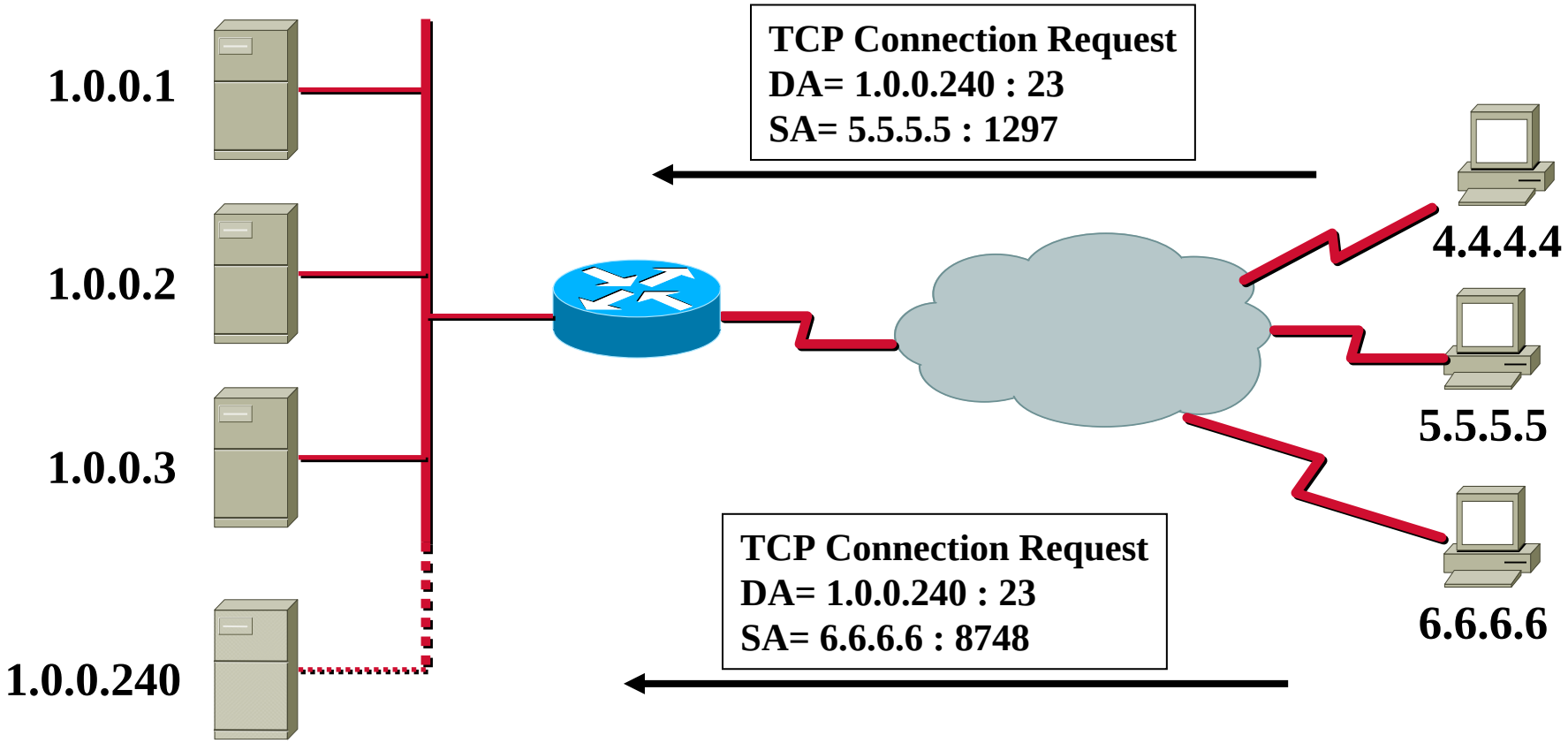
TCP Load Sharing (4)



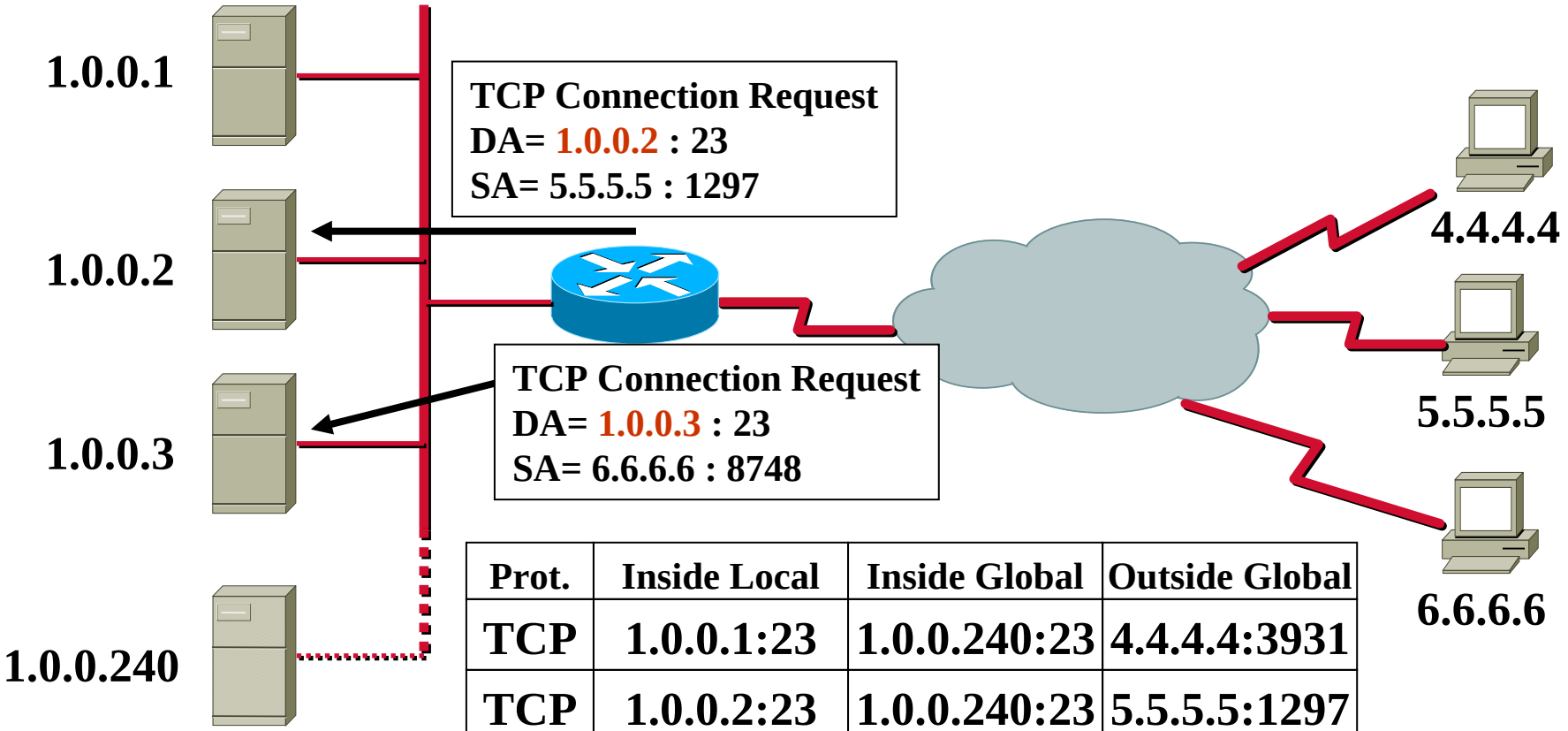
TCP Load Sharing (5)



TCP Load Sharing (6)



TCP Load Sharing (7)



Prot.	Inside Local	Inside Global	Outside Global
TCP	1.0.0.1:23	1.0.0.240:23	4.4.4.4:3931
TCP	1.0.0.2:23	1.0.0.240:23	5.5.5.5:1297
TCP	1.0.0.3:23	1.0.0.240:23	6.6.6.6:8748

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**

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

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

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

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, ...

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 } ]
```

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> }
```

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!