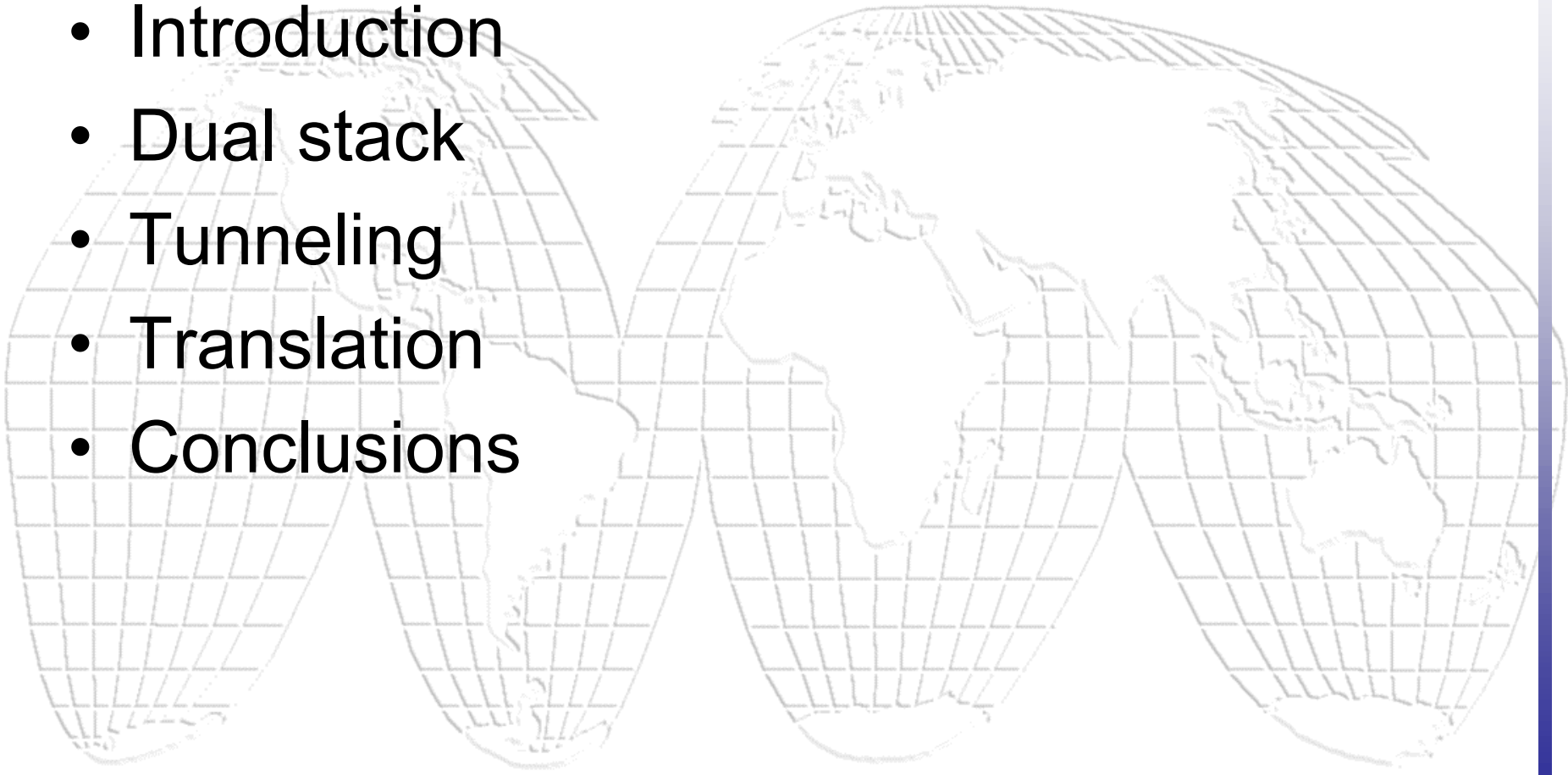
Transition Mechanisms

Javier Sedano (javier.sedano@agora-2000.com)
David Fernández (david@dit.upm.es)

Index



- Introduction
- Dual stack
- Tunneling
- Translation
- Conclusions

Introduction. Motivation

- IPv6 is not “backwards-compatible” ➔ need to change v4 to v6 (not “upgrade”)

- Needs:
  - Internet can not stop → no “flag day”
  - Internet is a large heterogeneous network → no centralization
  - Interoperation v4-v6 is a must → IPv6 is evolution, not revolution

- IPv6 designed thinking in transition

Introd

- IP

to

not

- IPv6 desig

Transition

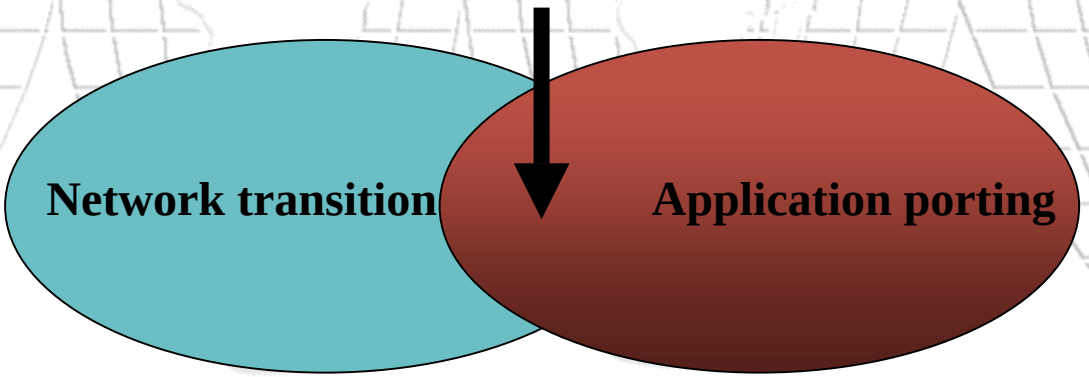
Introduction. This speech

- Tens of mechanisms:
  - A few “survivors”, widely used for general situations
  - Many solutions for very specific situations
- <1 hour → only overview :-(

  Many slides, for further study

Introduction. Other speeches

- Transition scenarios (Alberto García): “Ok, I understand what XYZ is… how can it solve my transition problem?”
- Application porting (Eva Castro, Tomás de Miguel): transition to avoid the porting ;-)



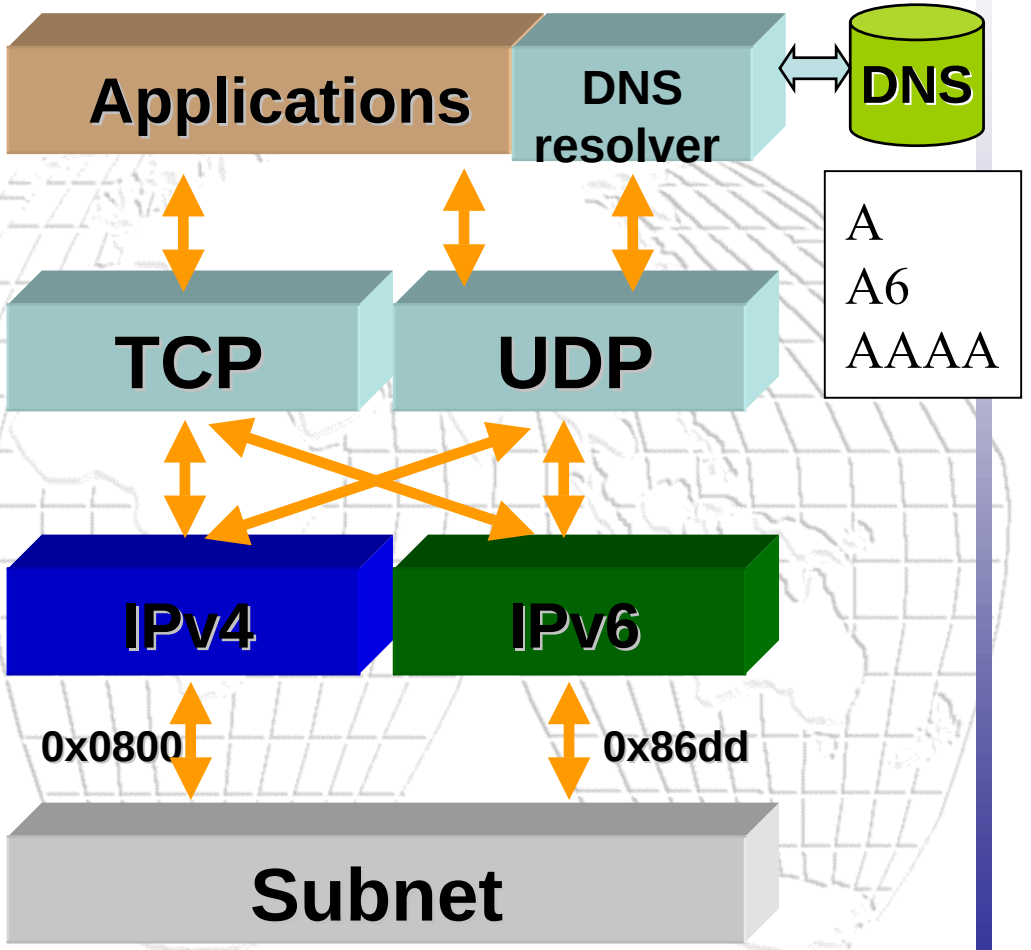
Index

- Introduction
- Dual stack
- Tunneling
- Translation
- Conclusions

Dual stack. Concepts

- Actually "dual layer 3"
- Both IPv4 and IPv6 addresses; resolver knowing of v4 (A) and v6 (A6/AAAA) addresses
- Ideal solution:
  - no transition at all!!
- The application must also know of dual-stack:
  - Clients: use A or A6/AAAA record? `connect()` to v4 or v6?
  - Server: `bind()` to both "`0.0.0.0`" and "`::`"

"Application porting"
(E. Castro, T. de Miguel)



Dual stack. Example


```
jsedano@taran:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:08:74:E6:80:89
          inet addr:192.168.0.55  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::208:74ff:fee6:8089/10 Scope:Link
          inet6 addr: 2001:3:2:1::a/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:6
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:444 (444.0 b)
          Interrupt:11 Base address:0xec80

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:66989 errors:0 dropped:0 overruns:0 frame:0
          TX packets:66989 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:25335759 (24.1 MiB)  TX bytes:25335759 (24.1 MiB)
jsedano@taran:~$
```

Dual stack. Example


```
jsedano@taran:~$ route –n -Ainet
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref  Use Iface
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0      0 eth0
0.0.0.0         192.168.0.1     0.0.0.0         UG    0      0      0 eth0
jsedano@taran:~$
jsedano@taran:~$ route -n -Ainet6
Kernel IPv6 routing table
Destination                    Next           HopFlags Metric Ref  Use Iface
::1/128                        ::             U        0      0      0 lo
2001:3:2:1::a/128              ::             U        0      0      0 lo
2001:3:2:1::/64                ::             UA       256    0      0 eth0
fe80::208:74ff:fee6:8089/128   ::             U        0      0      0 lo
fe80::/10                      ::             UA       256    0      0 eth0
ff00::/8                       ::             UA       256    0      0 eth0
::/0                           2001:3:2:1::1  U        256    0      0 eth0
jsedano@taran:~$
```

Dual stack. Example


```
jsedano@taran:~$ ping -c3 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=255 time=0.030 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=255 time=0.028 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=255 time=0.026 ms
--- 192.168.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.026/0.028/0.030/0.001 ms
jsedano@taran:~$
jsedano@taran:~$ ping -c3 2001:3:2:1::1
ping: unknown host 2001:3:2:1::1
jsedano@taran:~$
jsedano@taran:~$ ping6 -c3 2001:3:2:1::1
PING 2001:3:2:1::1(2001:3:2:1::1) 56 data bytes
64 bytes from 2001:3:2:1::1: icmp_seq=1 ttl=64 time=0.031 ms
64 bytes from 2001:3:2:1::1: icmp_seq=2 ttl=64 time=0.025 ms
64 bytes from 2001:3:2:1::1: icmp_seq=3 ttl=64 time=0.027 ms
--- 2001:3:2:1::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.025/0.027/0.031/0.006 ms
jsedano@taran:~$
```

Index

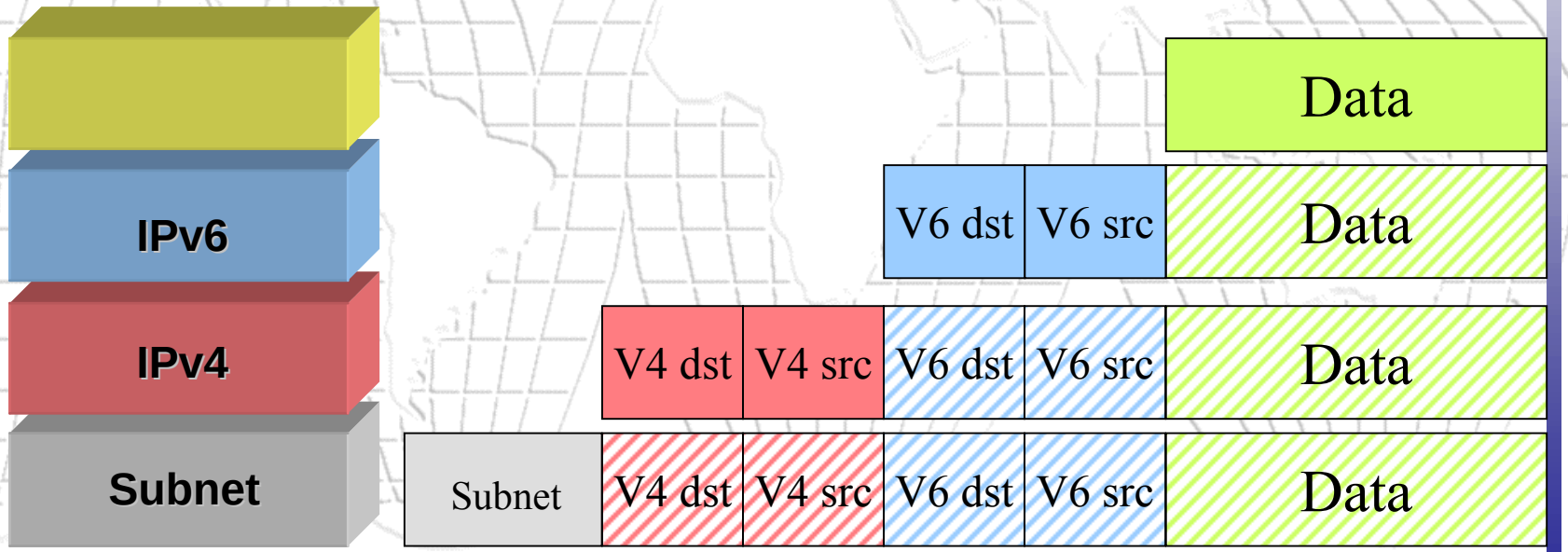
- Introduction
- Dual stack
- Tunneling
- Translation
- Conclusions

  - Configured tunnels
    - 6in4
    - 4in6
    - Tunnel broker
  - Automatic tunnels
  - 6to4 tunnels
  - Others
  - 6bone/m6bone

Tunneling. Configured tunnel

- Tunnels extensively used : MBONE, multiprotocol (IPX, Appletalk,…) over IP backbones, MIP,…
- RFC 2893: IPv6 in IPv4 tunnels

  IPv6 datagrams are encapsulated in IPv4 datagrams



Tunneling. Configured tunnel


```
jsedano@taran:~$ ip addr
[…]
7: myTunnel@NONE: <POINTOPOINT,NOARP,UP> mtu 1480 qdisc noqueue
    link/ipip 10.1.2.3 peer 192.168.6.5
    inet6 2001:1:b:c::5/126 scope global
[…]
jsedano@taran:~$
```

Tunneling. Configured tunnel


```
jsedano@taran:~$ ip addr
[…]
7: myTunnel@NONE: <POINTOPOINT,NOARP,UP> mtu 1480 qdisc noqueue
    link/ipip 10.1.2.3 peer 192.168.6.5
    inet6 2001:1:b:c::5/126 scope global
[…]
jsedano@taran:~$
```

Src. IPv4 address

Dst. IPv4 address

Tunneling concept

} pseudo point to point link

IPv6 network over such link

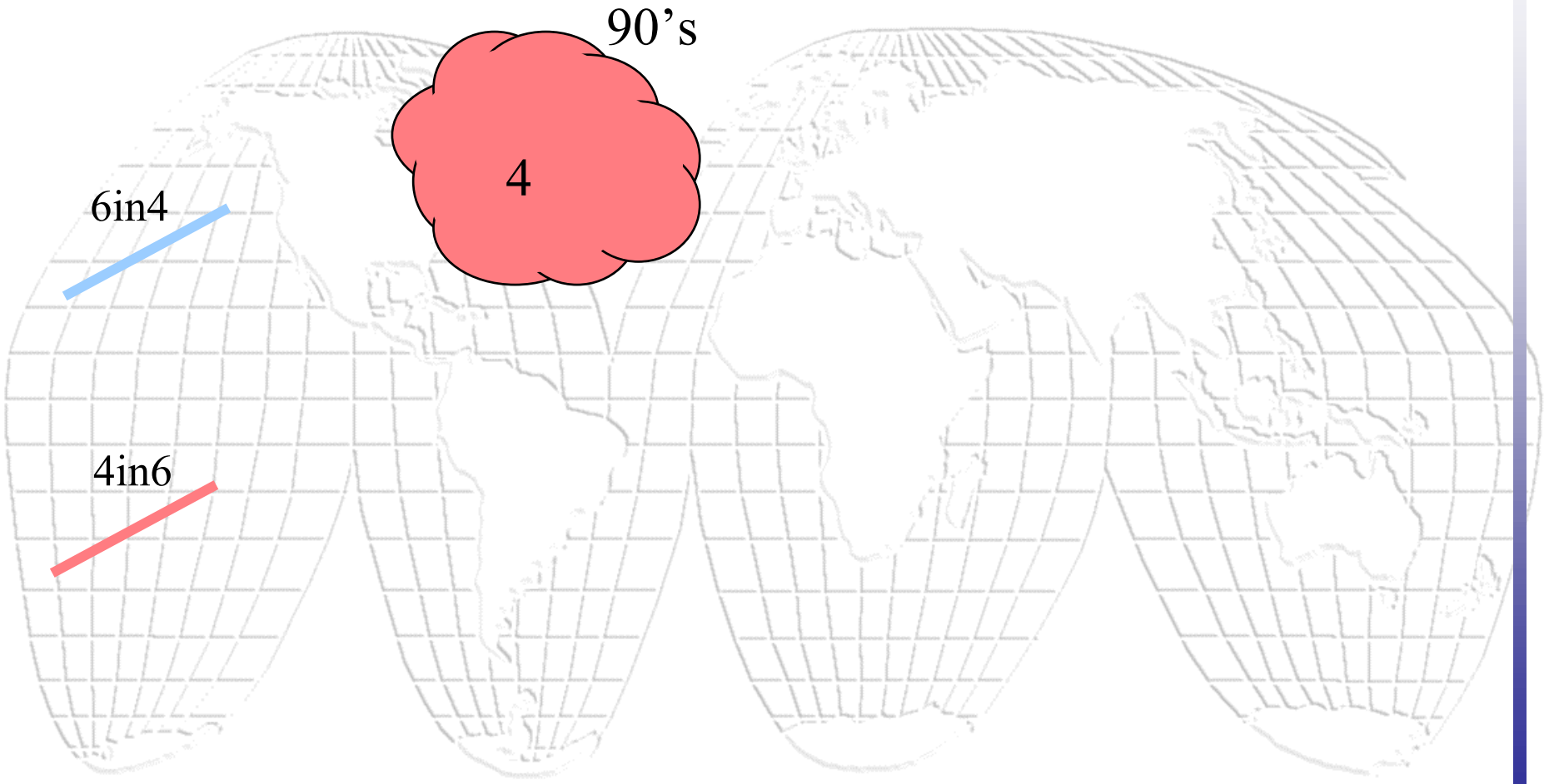
Tunneling. Configured tunnel

- Tunnel→pseudo p-to-p link→one IPv6 hop
  - Errors in the IPv4 path: ICMPv4 to the tunnel origin endpoint. If enough information in ICMPv4, an ICMPv6 error can be sent back to the source
- No fragmentation in IPv6 ➔ IPv4 path MTU discovery inside the tunnel

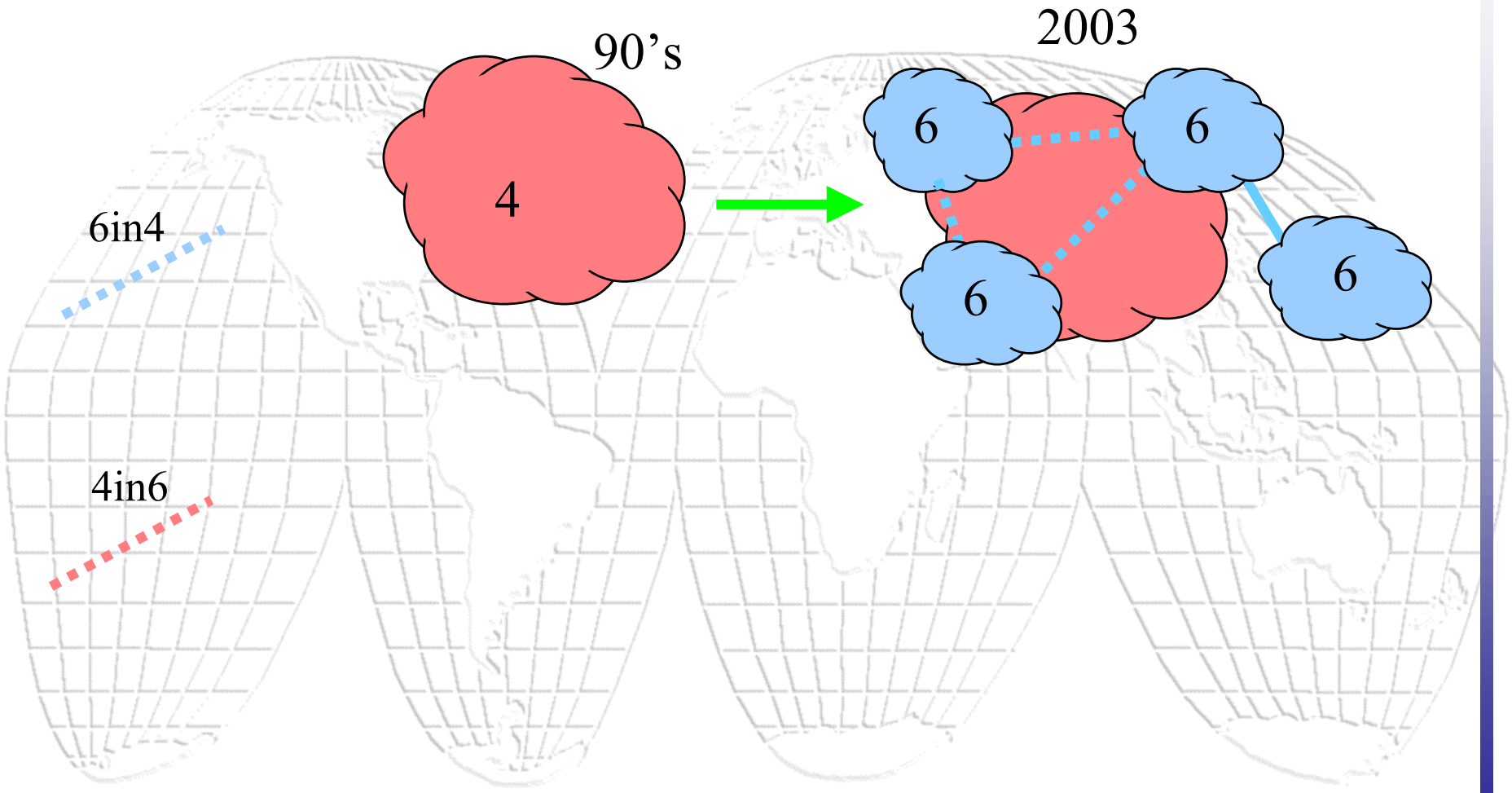
  Warning: IPv6 MTU >= 1280 bytes
- Host2host, host2router, router2router

Tunneling. 6in4, 4in6

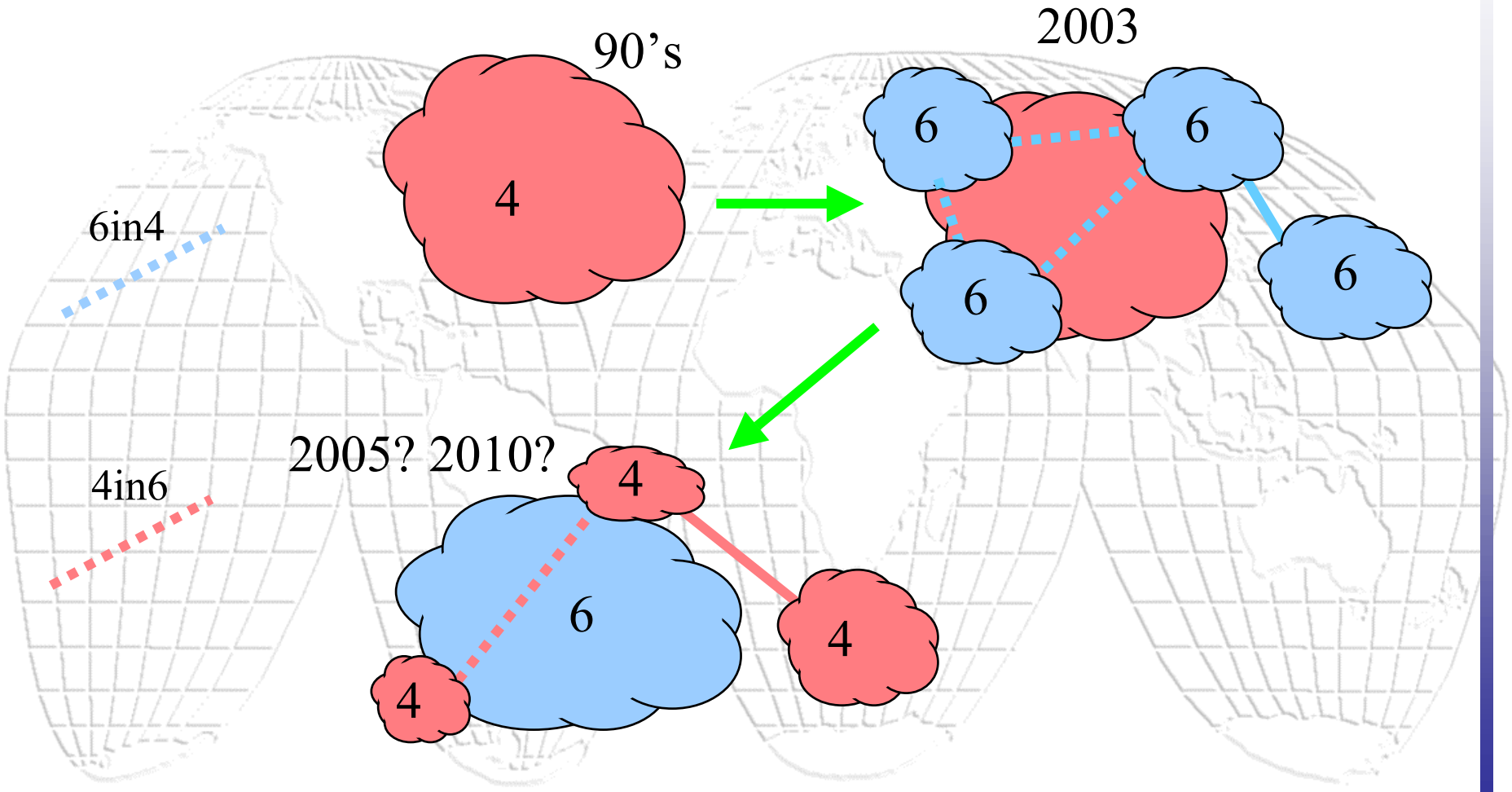
Tunneling. 6in4, 4in6



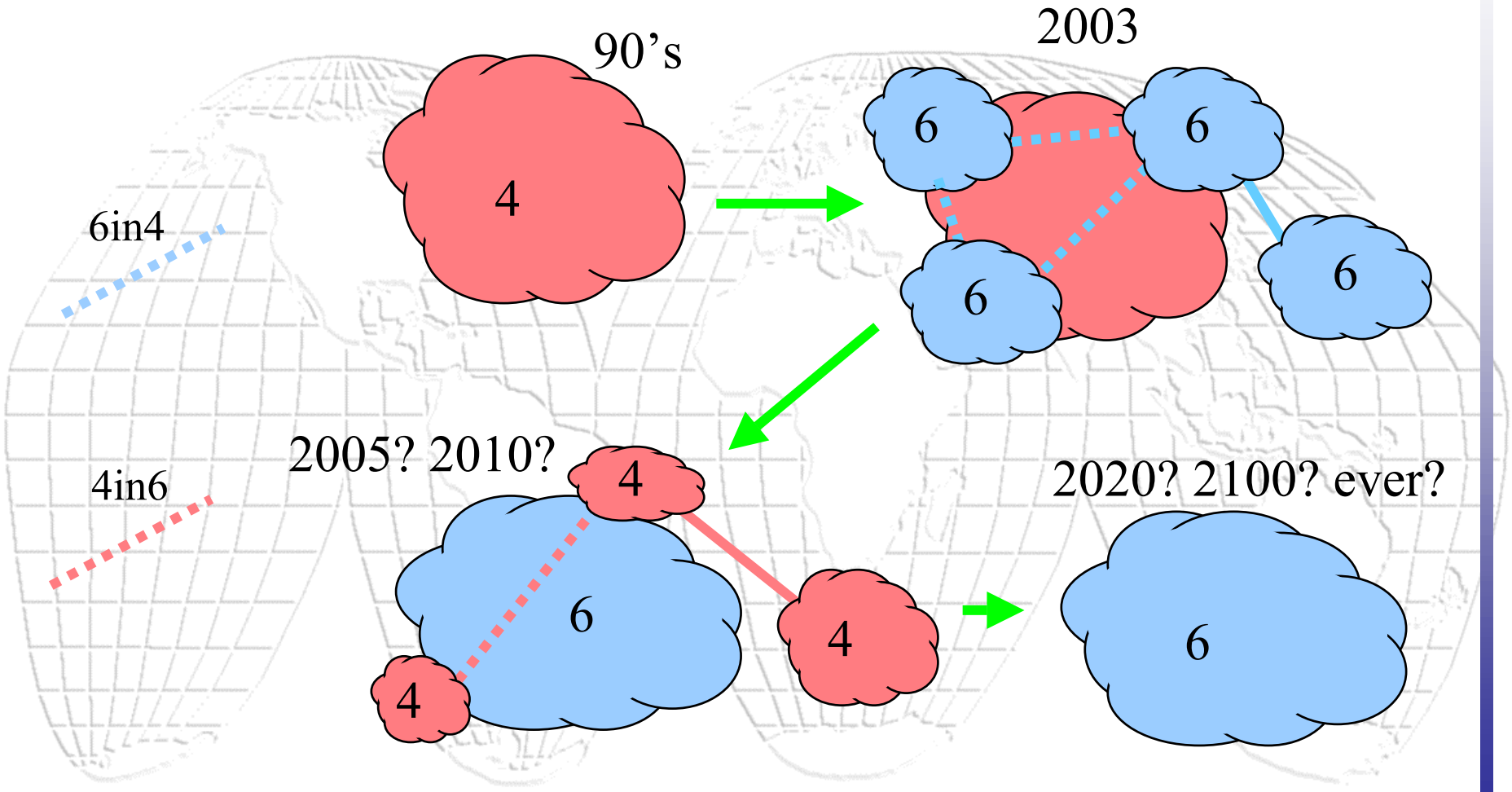
Tunneling. 6in4, 4in6



Tunneling. 6in4, 4in6



Tunneling. 6in4, 4in6

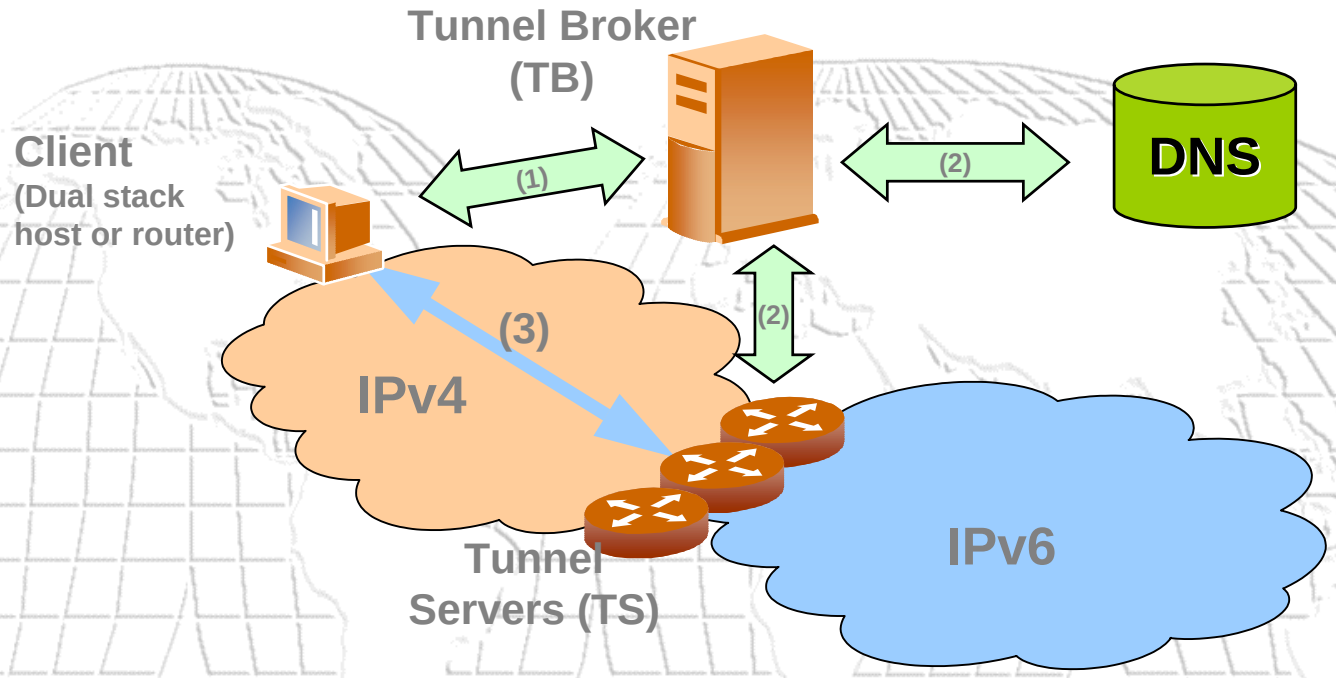


Tunneling. Tunnel broker

- RFC 3053
- Helps automating the setting up of configured tunnels

  The tunnel broker:
  - Creates and configures the “server” side of the tunnel
  - Provides the “client” with information to configure its side
- Examples:
  - Freenet6. http://www.freenet6.net
  - CSELT. https://carmen.cselt.it/ipv6tb
  - BT. http://tb.ipv6.bt.com/v6broker/

Tunneling. Tunnel broker



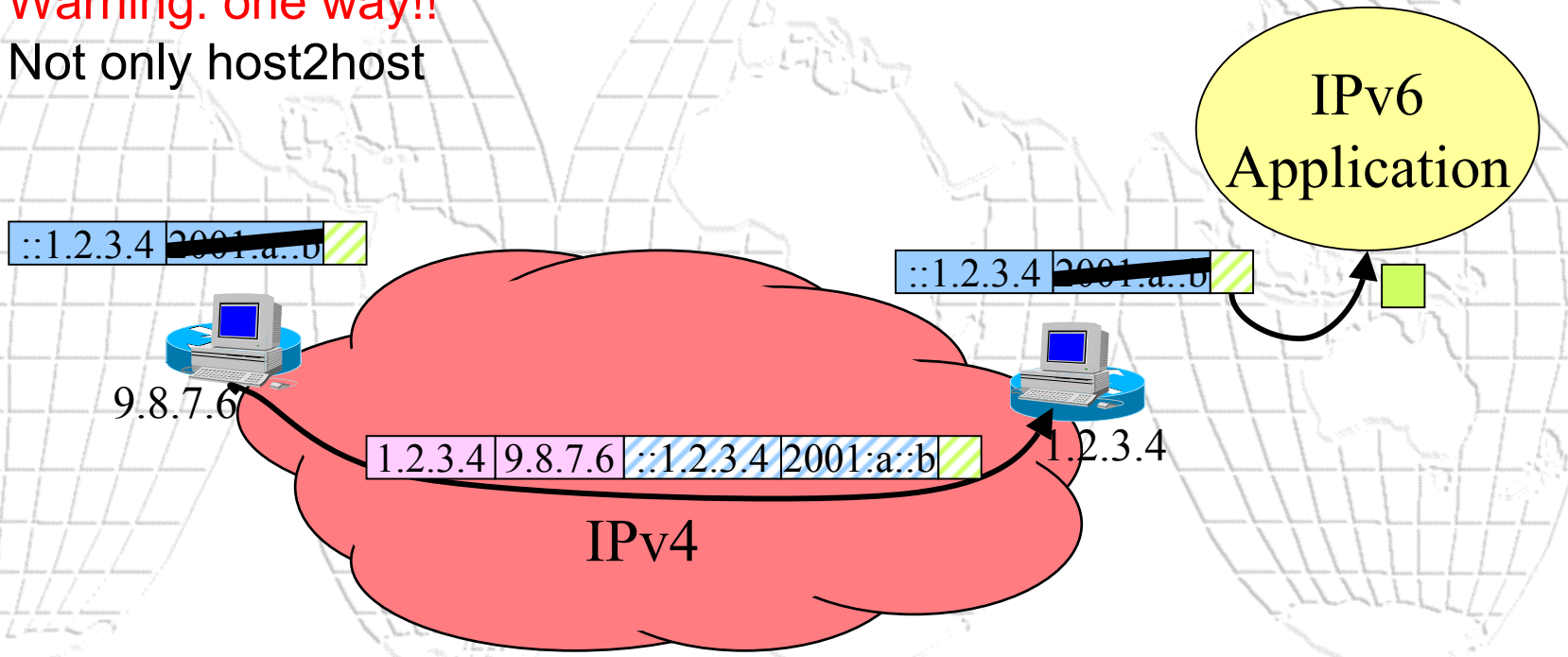
1.- Client registers (e.g. through https) in TB and gets IPv6 address/es

2.- TB configures the server part of the tunnel in a TS and registers the addresses in DNS

3.- Client configures (manually or by means of scripts provided by TB) the client side of the tunnel

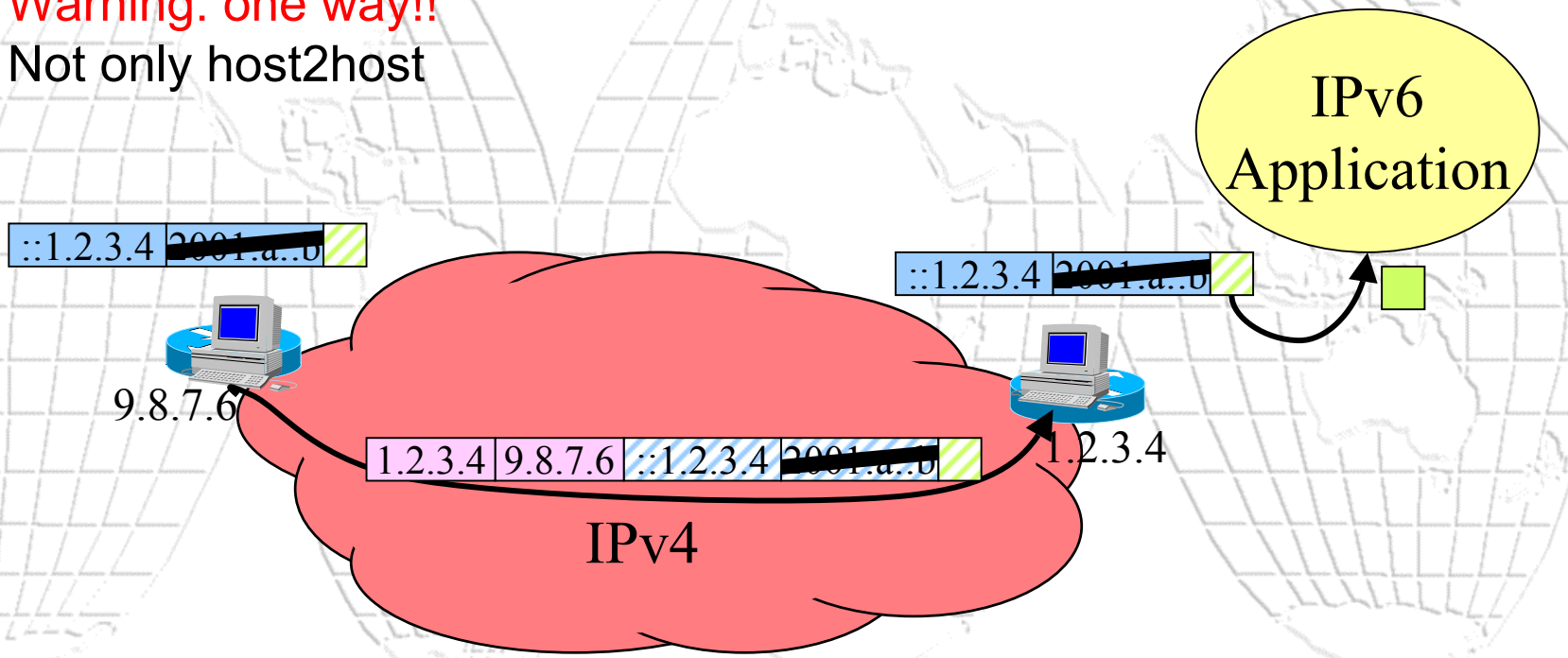
Tunneling. Automatic tunnel

- ::x.y.z.t addresses (x.y.z.t being an IPv4 address)
- When a host with automatic tunnels enabled sees a datagram to ::x.y.z.t, it is encapsulated and sent to x.y.z.t over IPv4
- Warning: one way!!
- Not only host2host



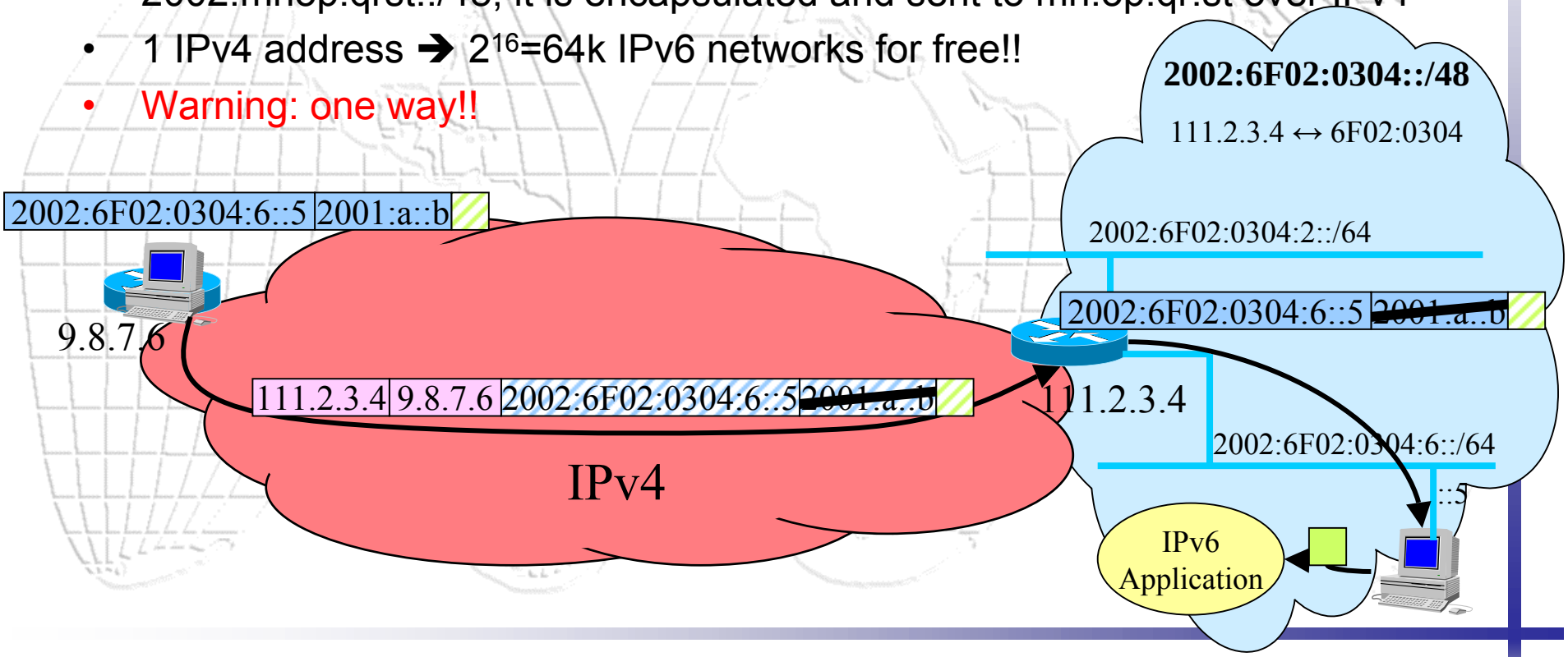
Tunneling. Automatic tunnel

- ::x.y.z.t addresses (x.y.z.t being an IPv4 address)
- When a host with automatic tunnels enabled sees a datagram to ::x.y.z.t, it is encapsulated and sent to x.y.z.t over IPv4
- Warning: one way!!
- Not only host2host



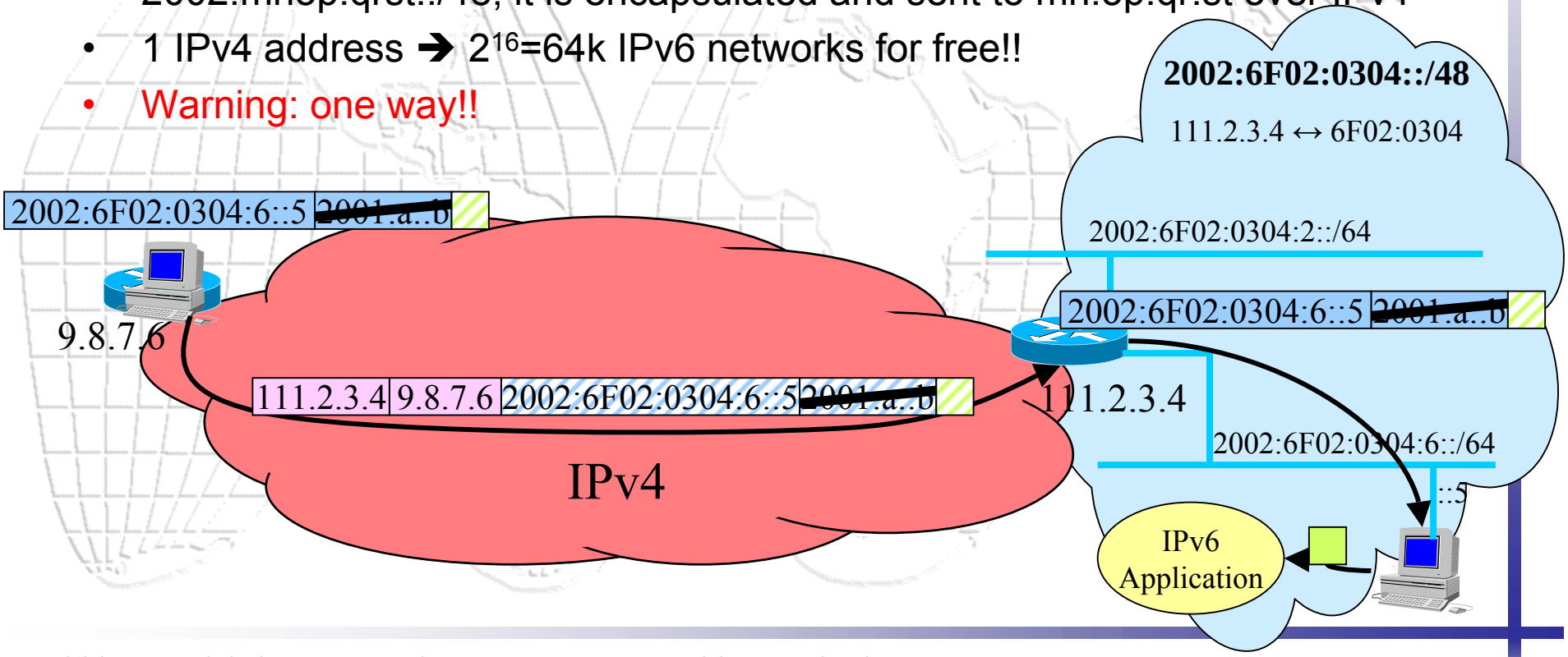
Tunneling. 6to4

- 6to4: 2002:mnop:qrst::/48 (mnop:qrst is an IPv4 address in hexadecimal notation)
- When a host with 6to4 tunnels enabled (aka 6to4 relay) sees a datagram to 2002:mnop:qrst::/48, it is encapsulated and sent to mn.op.qr.st over IPv4
- 1 IPv4 address ➔ 2^{16}=64k IPv6 networks for free!!
- Warning: one way!!

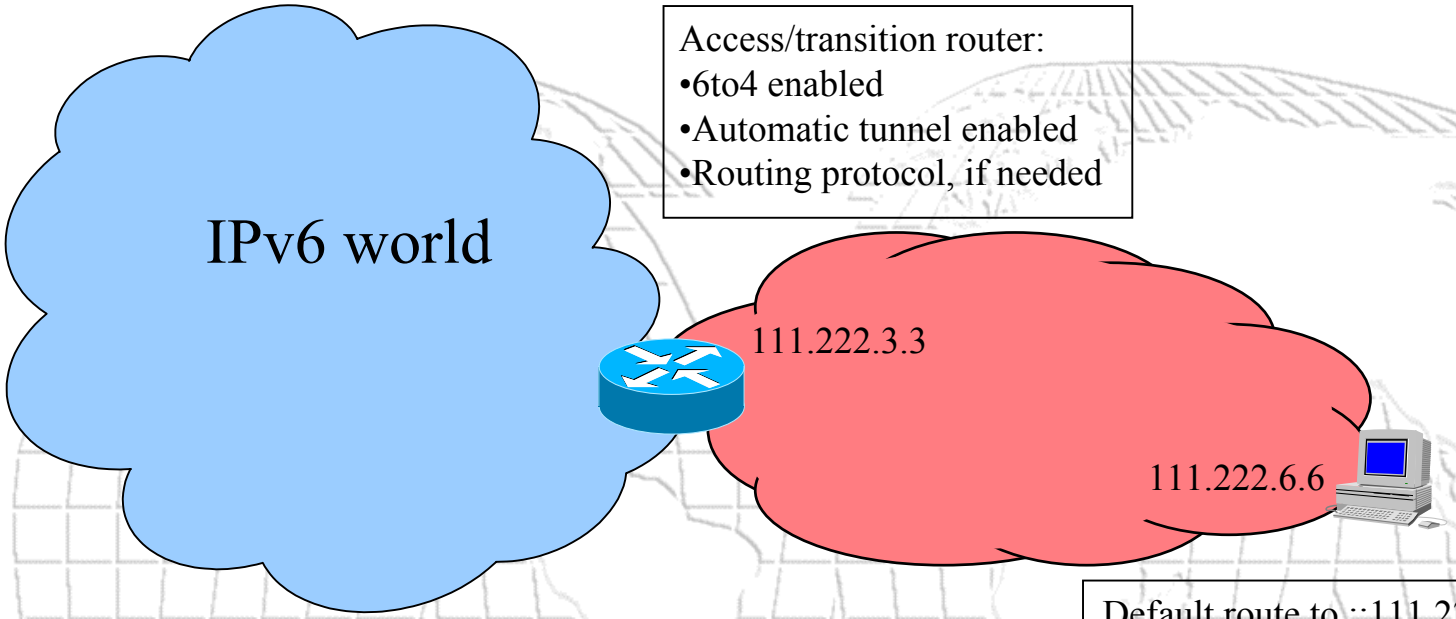


Tunneling. 6to4

- 6to4: 2002:mnop:qrst::/48 (mnop:qrst is an IPv4 address in hexadecimal notation)
- When a host with 6to4 tunnels enabled (aka 6to4 relay) sees a datagram to 2002:mnop:qrst::/48, it is encapsulated and sent to mn.op.qr.st over IPv4
- 1 IPv4 address ➔ 2^{16}=64k IPv6 networks for free!!
- Warning: one way!!



Tunneling. 6to4 + automatic



•Host to IPv6 world

Src=2002:6FDE:0303:1::1

Dst=a:b:c:d::f

Gw=::111.222.3.3

•IPv6 world to host

Src=a:b:c:d::f

Dst=2002:6FDE:0303:1::1

The first 6to4 relay able to route it will encapsulate and send it to 111.222.6.6

Default route to ::111.222.3.3
One interface with 2002:6FDE:0303:1::1

6FDE:0303 ↔ 111.222.3.3

Tunneling. Others

- **6over4**

  If the underlying IPv4 network supports mcast, it is very similar to Ethernet ➔ Use IPv4 as underlying layer 2

- **DSTM** (Dual Stack Transition Mechanism):

  4in6 inside an IPv6-only network; a known TEP (Tunnel End Point) deencapsulates to the IPv4 world

- **TEREDO**:

  6in4 does not traverse NAPTs ➔ 6inUDP

- **ISATAP** (Intra-Site Automatic Tunnel Addressing Protocol):

  IPv6 address = Prefix/64 + ID

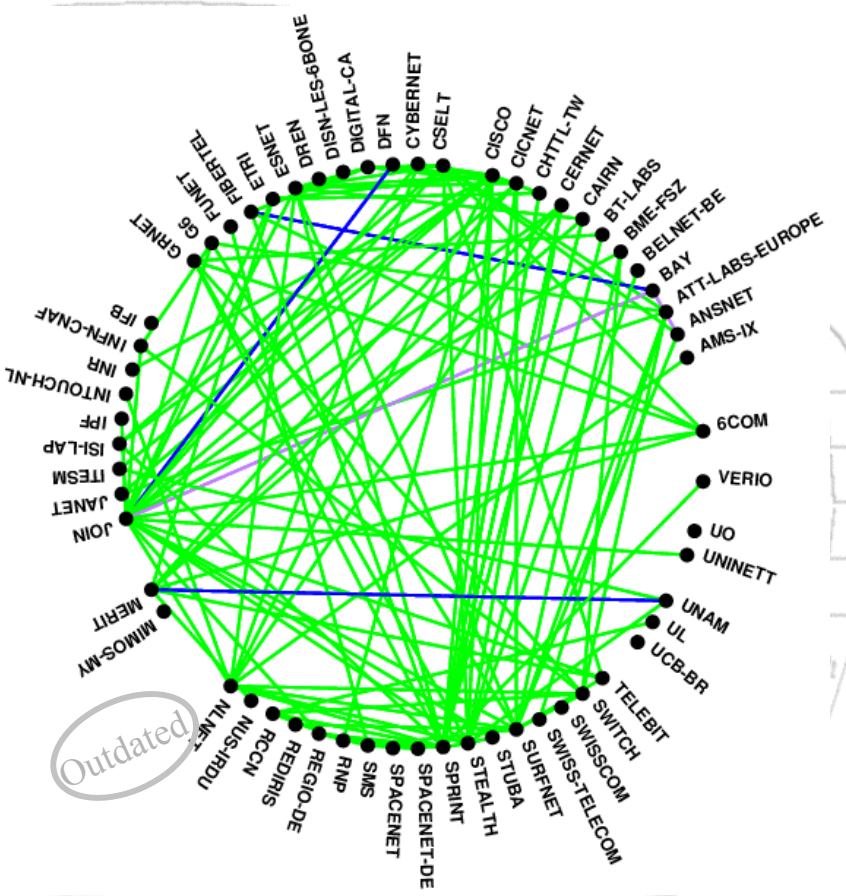
  Prefix ➔ allocated by the administrator

  ID ➔ with the IPv4 address embedded

  Tunneled 6in4 inside the site

Tunneling. 6bone/m6bone

- 6bone
  - Research network
  - IPv6 in IPv4 tunnels
  - Validation of IPv6 itself, addressing, routing, transition mechanism, applications,…
  - http:://www.6bone.net
- M6bone
  - 6in4 and 6in6 tunnels. Some native links
  - IPv6 multicast research



Index

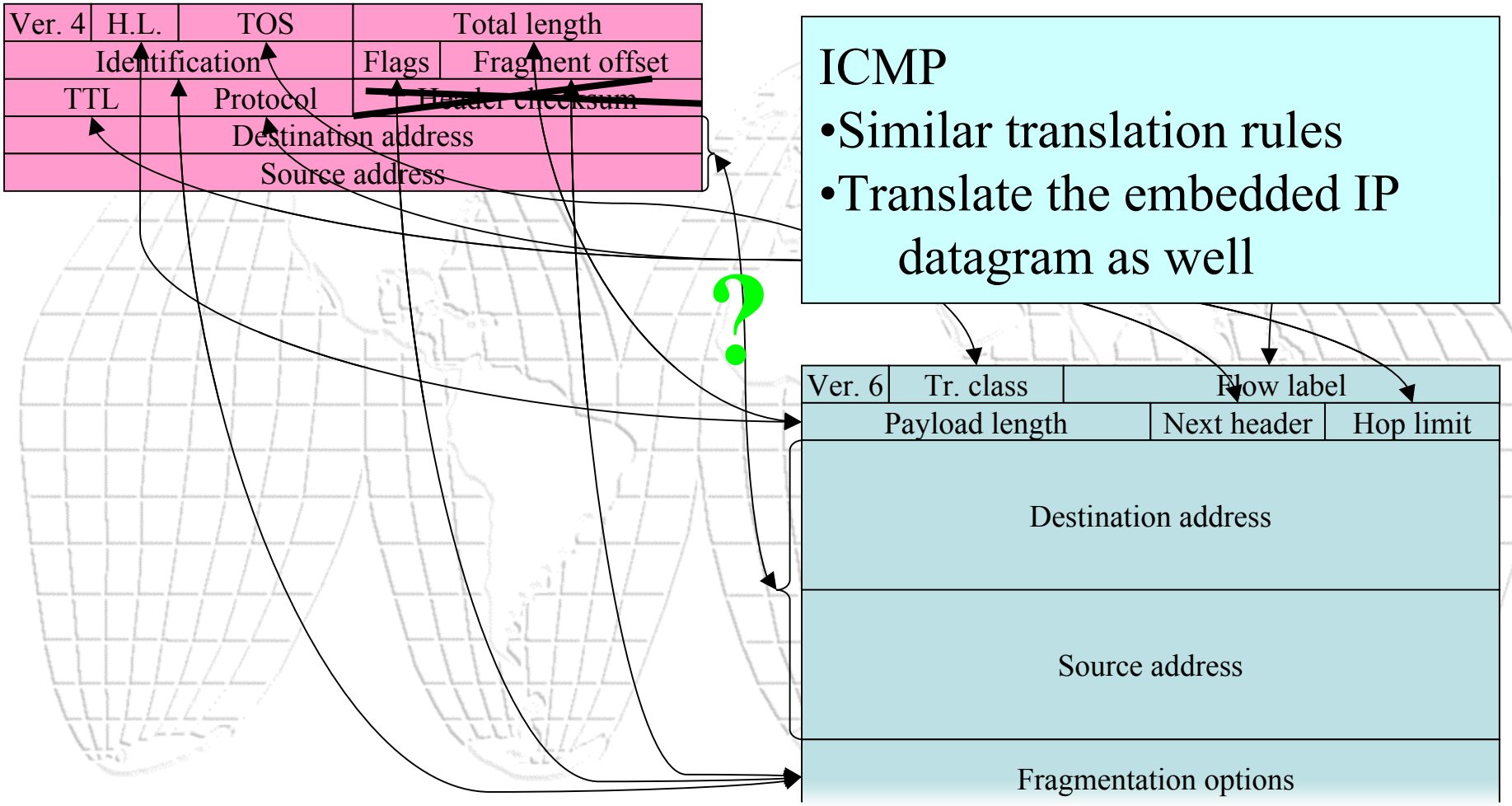
- Introduction
- Dual stack
- Tunneling
- Translation
  - SIIT
  - Application layer gateway
  - TRT
  - NAT-PT
  - Others
- Conclusions

Translation. SIIT

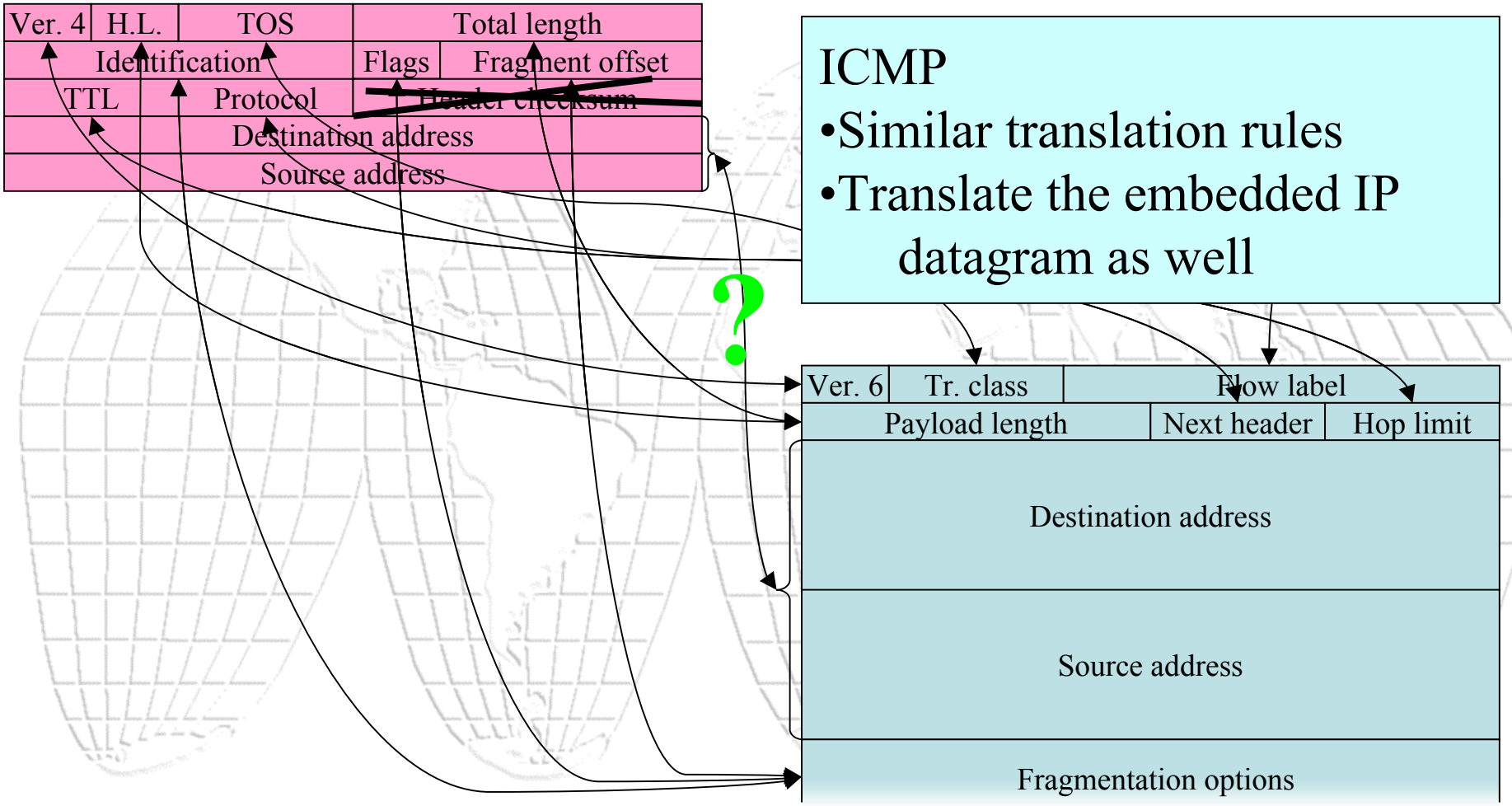
- IPv6 is evolution, not revolution:
  - Same philosophy
  - Many similar fields (TOS ↔ Tclass, TTL ↔ HopLimit,…)
- Translation is possible and makes sense
- SIIT (Stateless IP/ICMP Translator):

  A framework for other translators

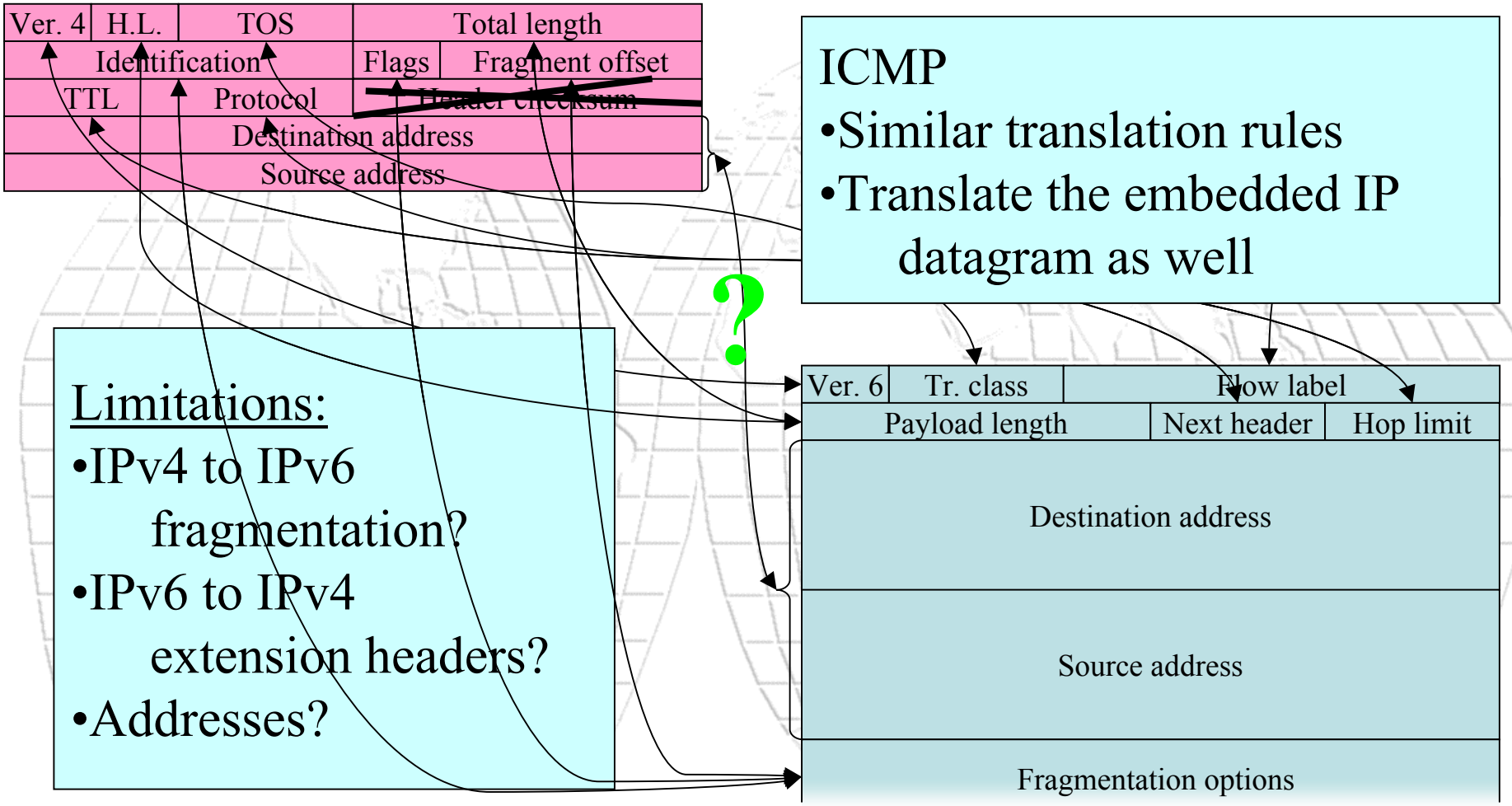
Translation. SIIT



Translation. SIIT

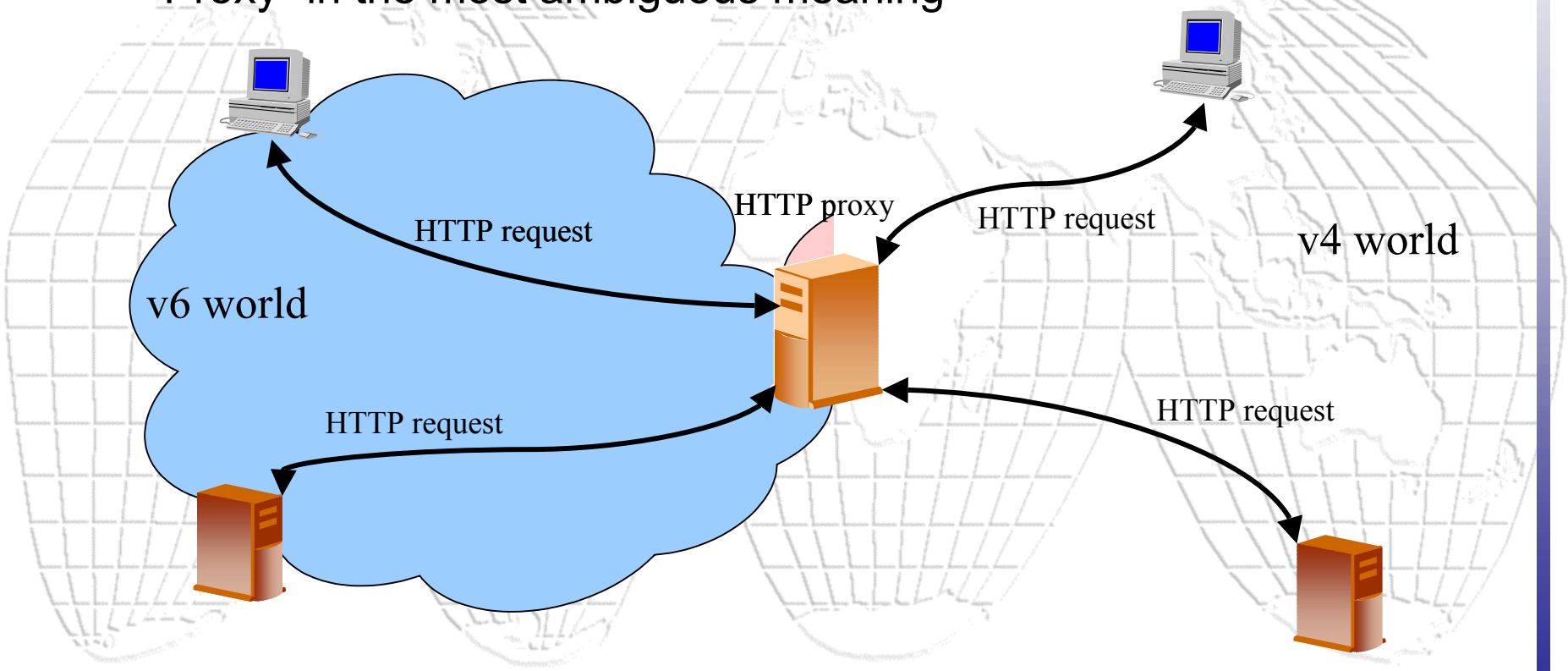


Translation. SIIT



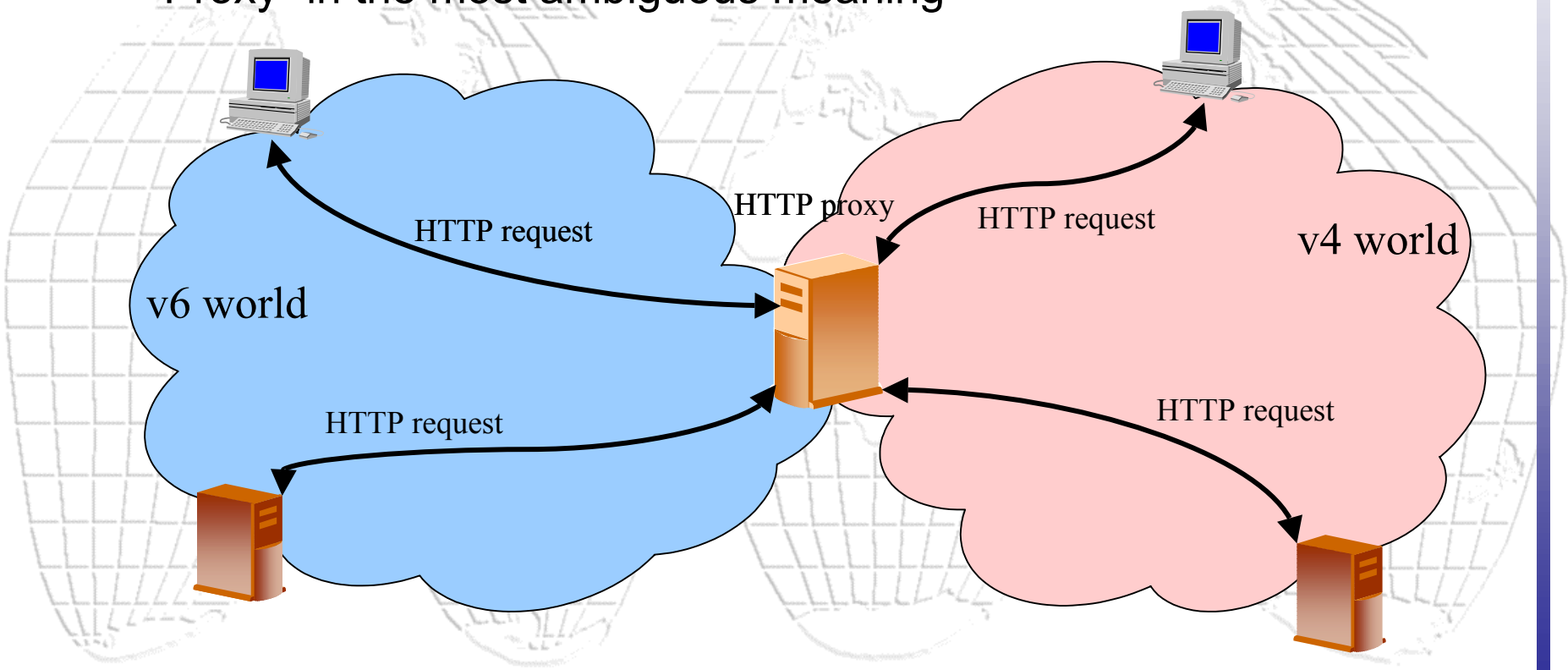
Translation. App. layer gateway

- What if the HTTP proxy is dual-stacked?
  - "Proxy" in the most ambiguous meaning



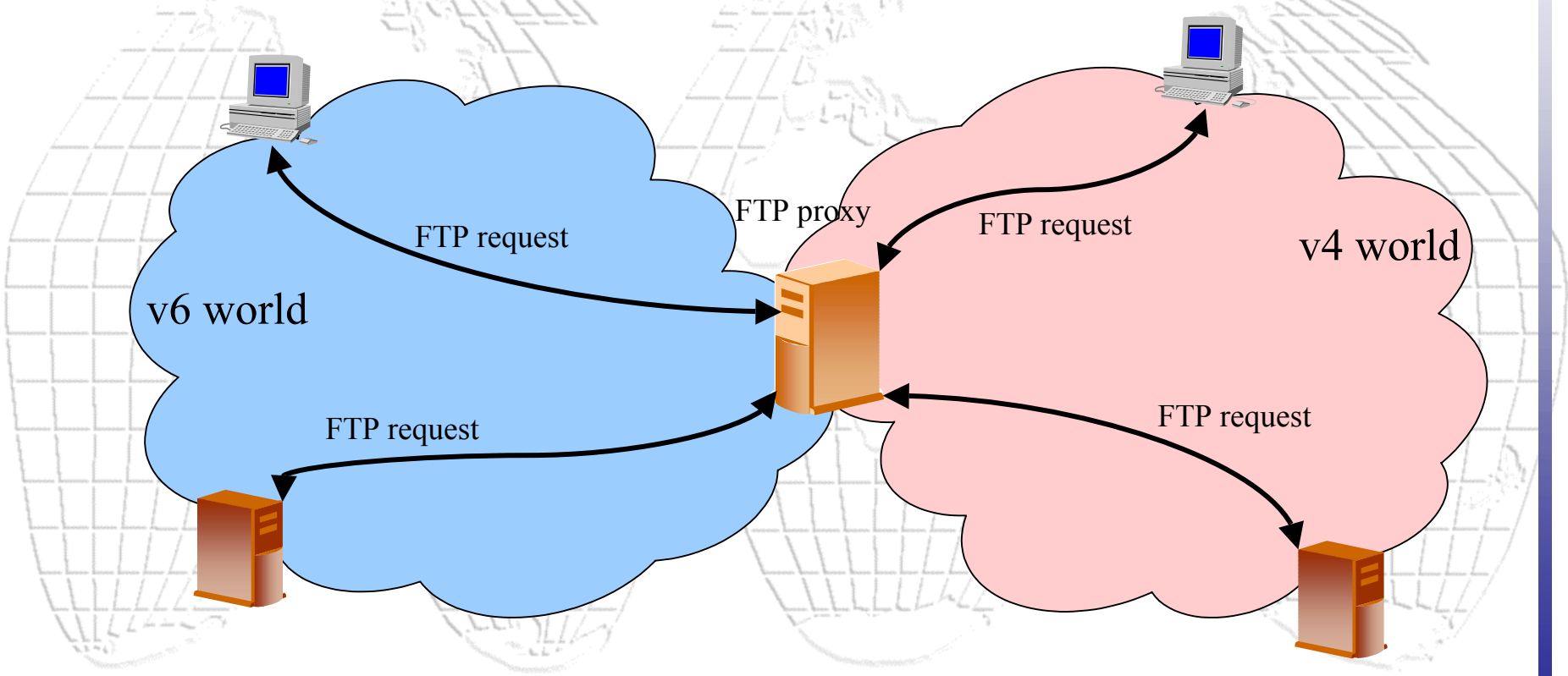
Translation. App. layer gateway

- What if the HTTP proxy is dual-stacked?
  - "Proxy" in the most ambiguous meaning



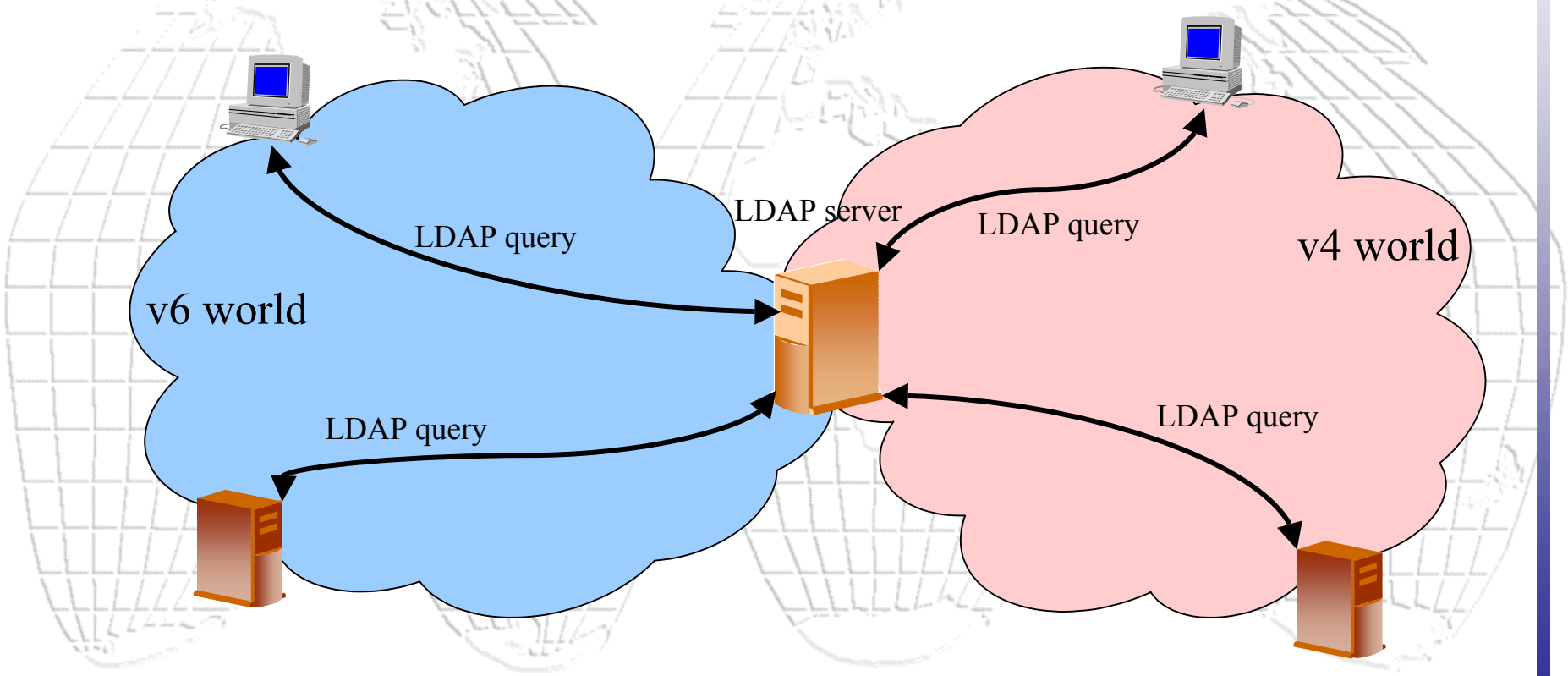
Translation. App. layer gateway

- What if the FTP proxy is dual-stacked?



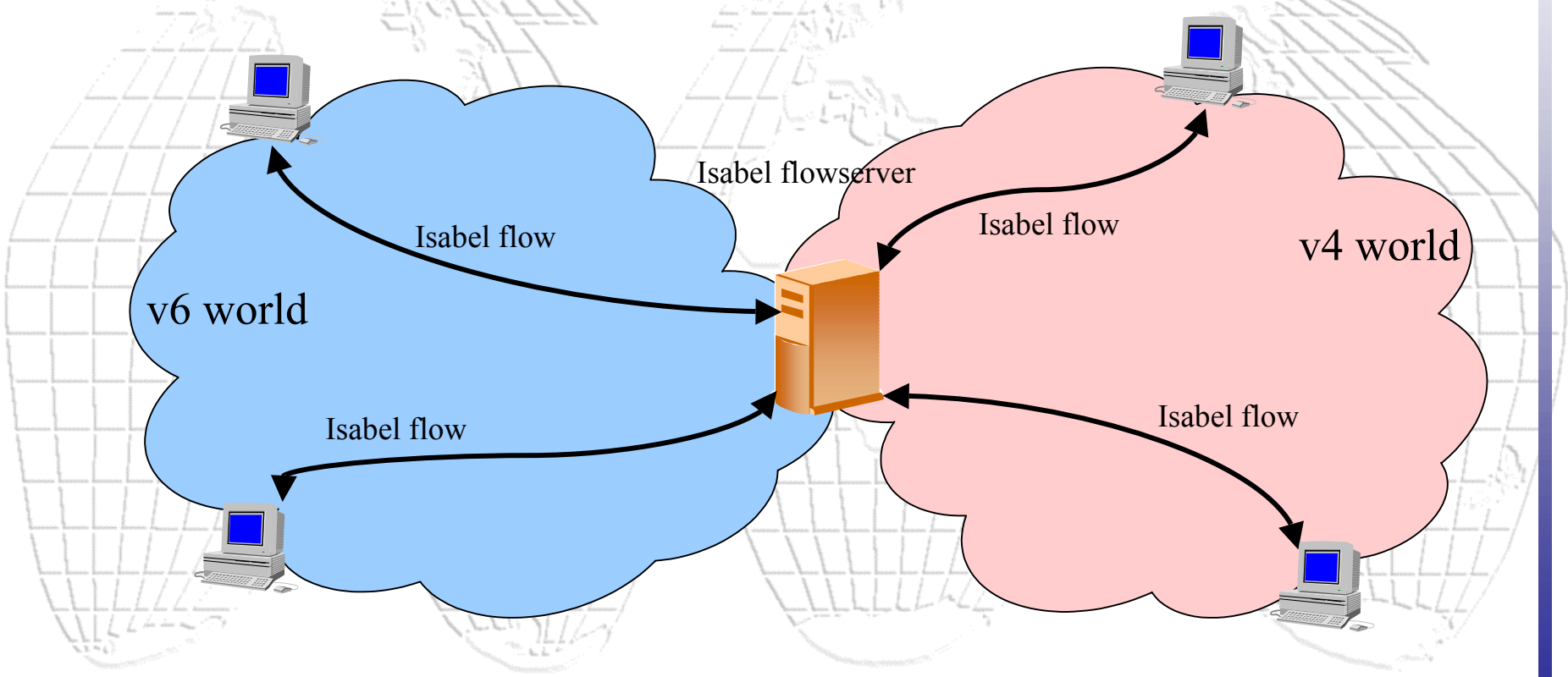
Translation. App. layer gateway

- What if the LDAP server is dual-stacked?



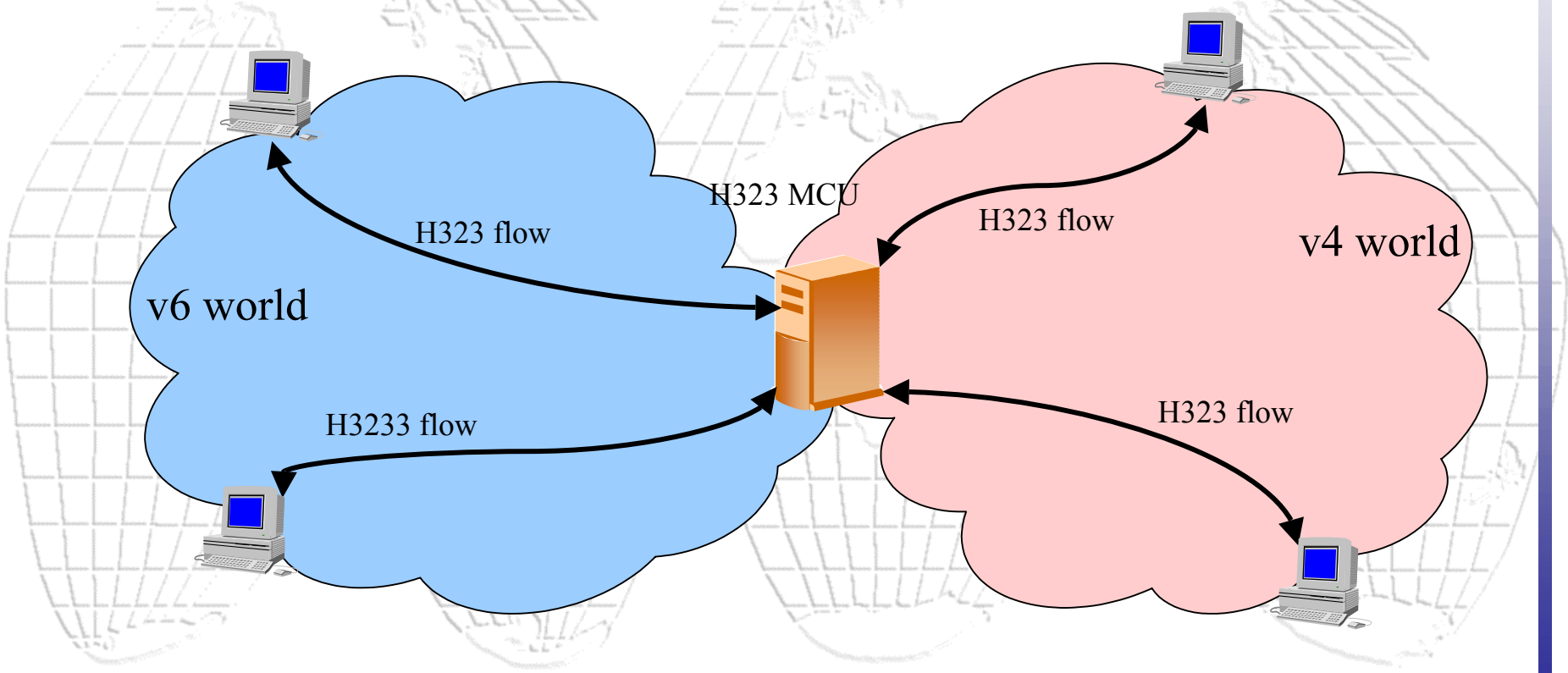
Translation. App. layer gateway

- What if the Isabel flowserver is dual-stacked?



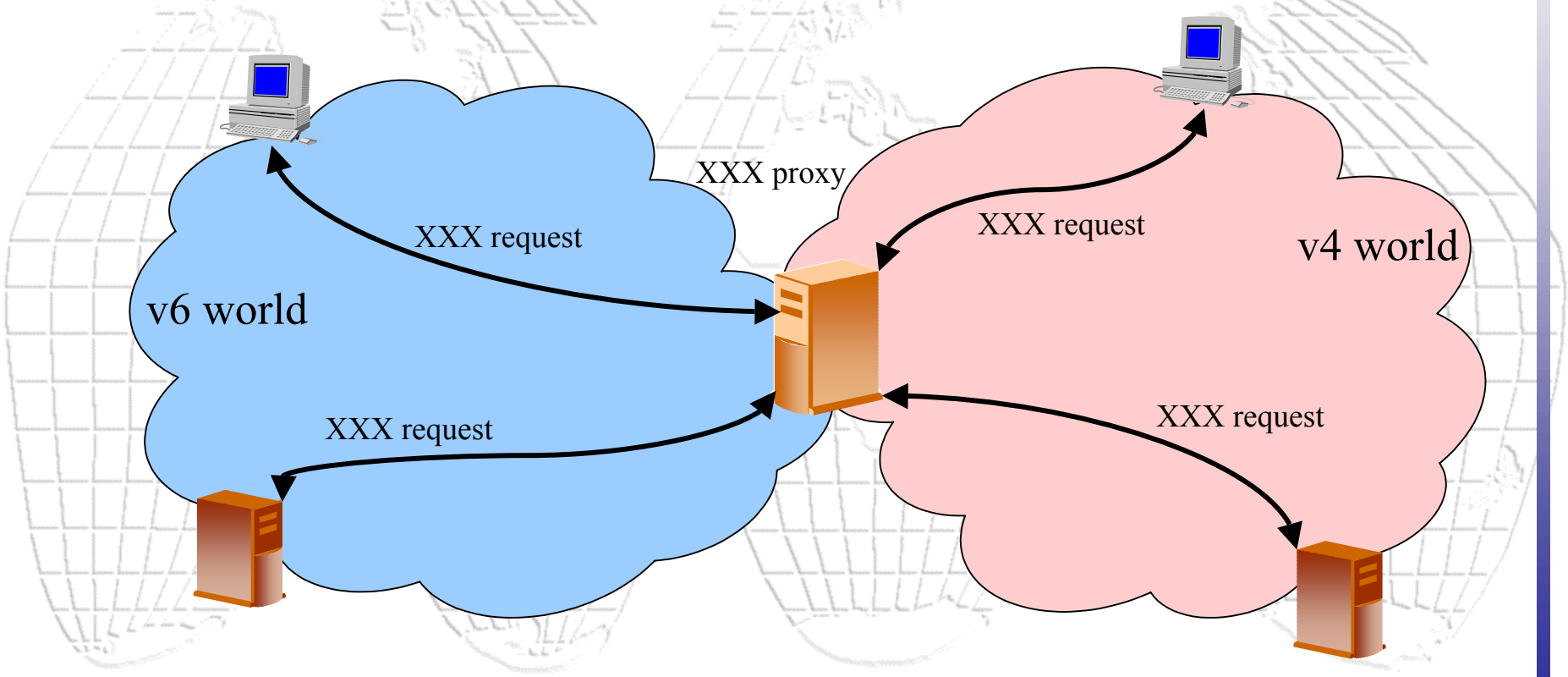
Translation. App. layer gateway

- What if the H323 MCU is dual-stacked?



Translation. App. layer gateway

- What if the XXX proxy is dual-stacked?

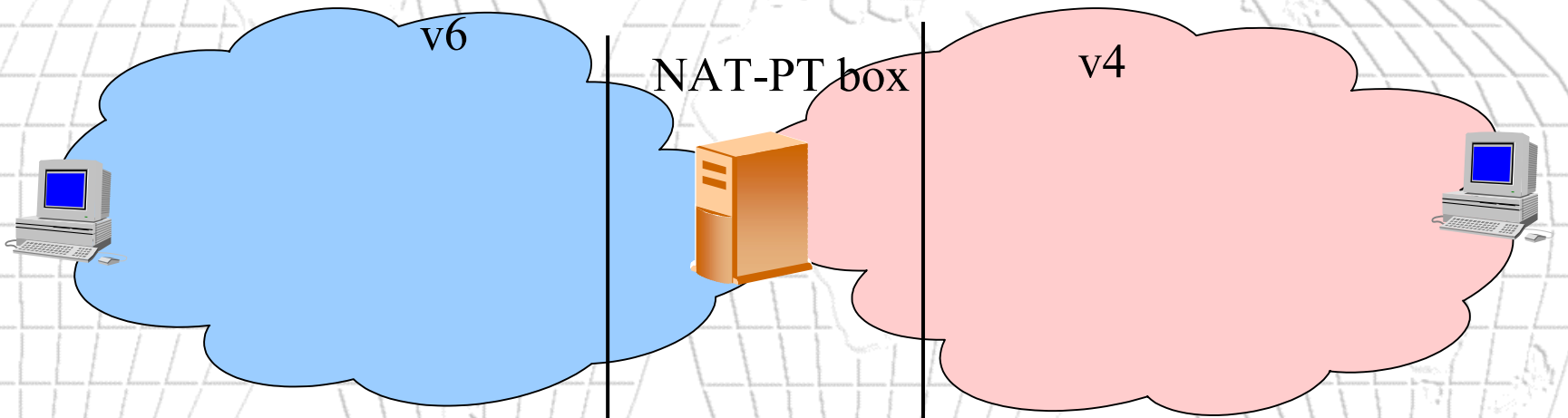


Translation. App. layer gateway

- No change in the clients!
  - Transparent to scared users
  - Useful for brave sysadmins
- Single failure point
- Scalability problems?
- Only applicable to some protocols/apps

Translation. NAT-PT

- Probably the most complex mechanism
- One idea, several situations
- NAT-PT uses SIIT for header translation



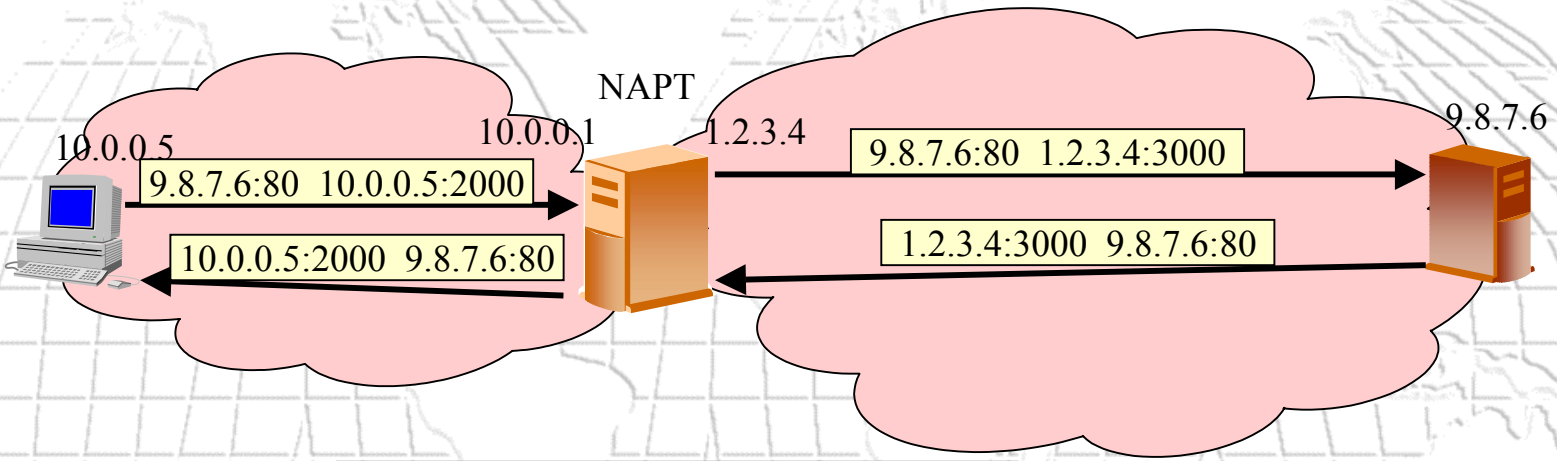
The v4 network is seen as a set of v6 addresses (from 1 to all)

PREFIX::/96 (PREFIX may be ::)

The v6 network is seen as a set of v4 addresses (from 1 to all)

Translation. NAT-PT

Classical IPv4 NAT (NAPT)

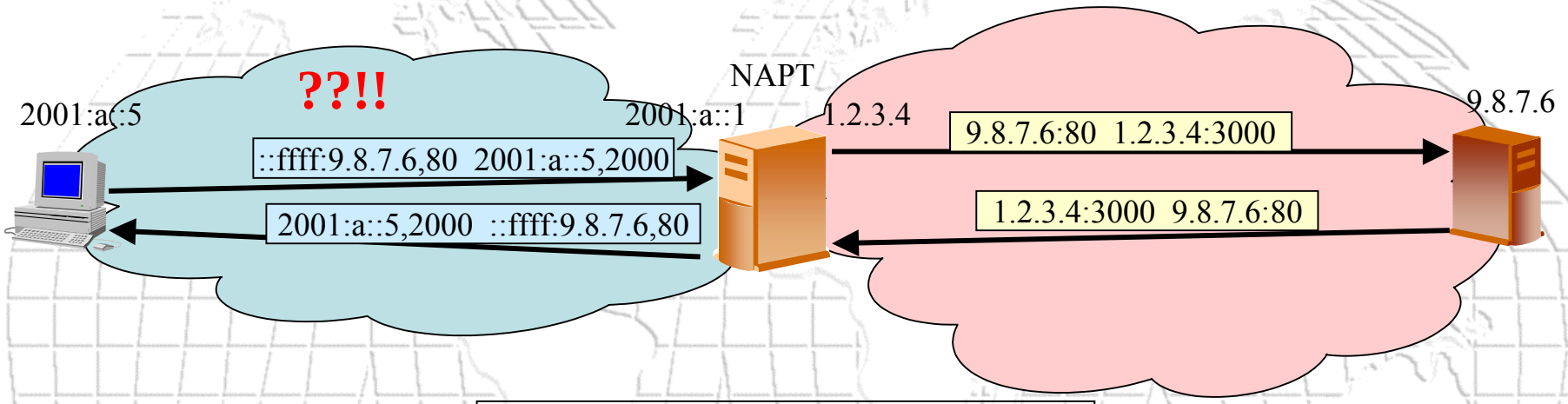


NAPT table

Sa	Sp	Na	Np	Da	Dp
10.0.0.5	2000	1.2.3.4	3000	9.8.7.6	80
10.0.0.5	2001	1.2.3.4	3001	9.8.7.6	6667
10.0.0.6	1500	1.2.3.4	3002	9.9.9.9	80
10.0.0.6	1501	1.2.3.4	3003	8.8.8.8	80

Translation. NAT-PT

Simple IPv6 to IPv4 NAT-PT (NAPT-PT)

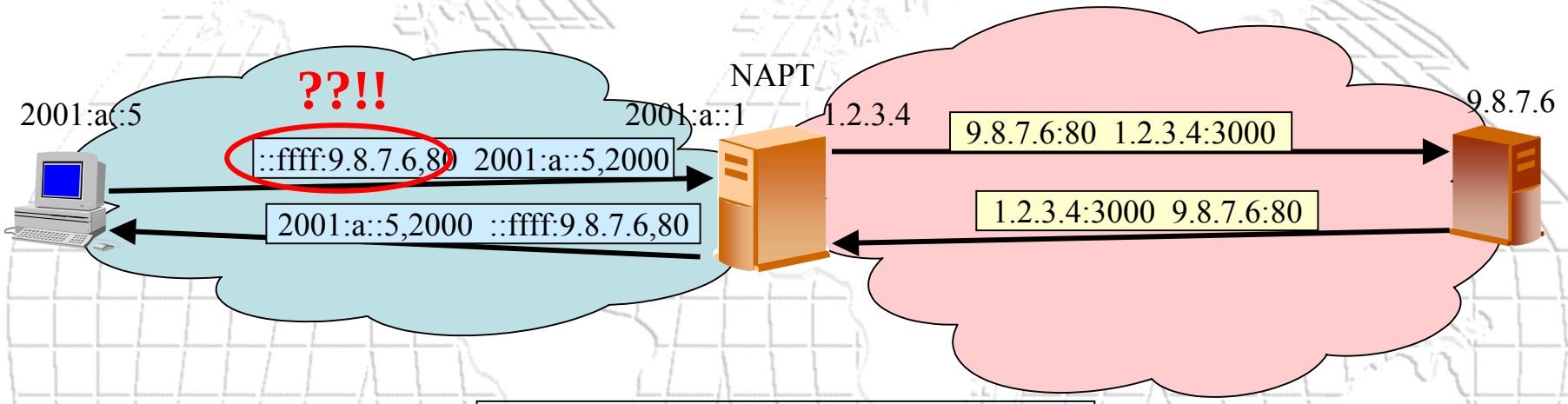


NAPT-PT table

Sa	Sp	Na	Np	Da	Dp
2001:a::5	2000	1.2.3.4	3000	9.8.7.6	80
2001:a::5	2001	1.2.3.4	3001	9.8.7.6	6667
2001:b::6	1500	1.2.3.4	3002	9.9.9.9	80
2001:b::6	1501	1.2.3.4	3003	8.8.8.8	80

Translation. NAT-PT

Simple IPv6 to IPv4 NAT-PT (NAPT-PT)



NAPT-PT table

Sa	Sp	Na	Np	Da	Dp
2001:a::5	2000	1.2.3.4	3000	9.8.7.6	80
2001:a::5	2001	1.2.3.4	3001	9.8.7.6	6667
2001:b::6	1500	1.2.3.4	3002	9.9.9.9	80
2001:b::6	1501	1.2.3.4	3003	8.8.8.8	80

Translation. NAT-PT

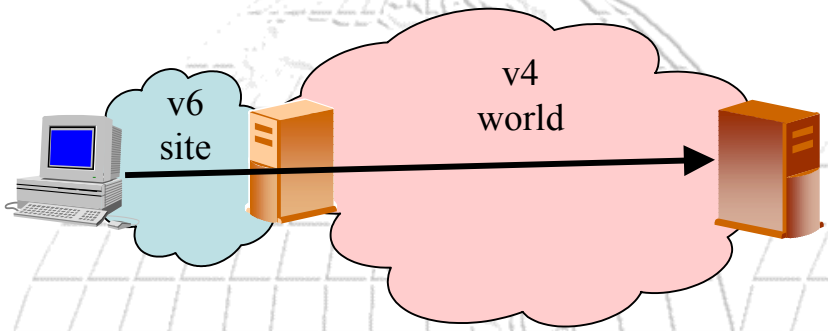
- Why ::ffff:9.8.7.6?
  1. The client requests AAAA for www.xyz.com
  2. The DNS server request A for www.x.y.z.com and generates the AAAA, which is returned to the client
     - Explicit DNS translator server
     - Transparent DNS translator intercepts the request
  - ::ffff: prefix is not a must, just a good example
- Only clients behind the NAPT-PT box
  - Explicit rules for servers behind the NAPT-PT box
  - Explicit rules to avoid the A→AAAA translation (i.e.: service deployment in a v4 site)

Translation. NAT-PT

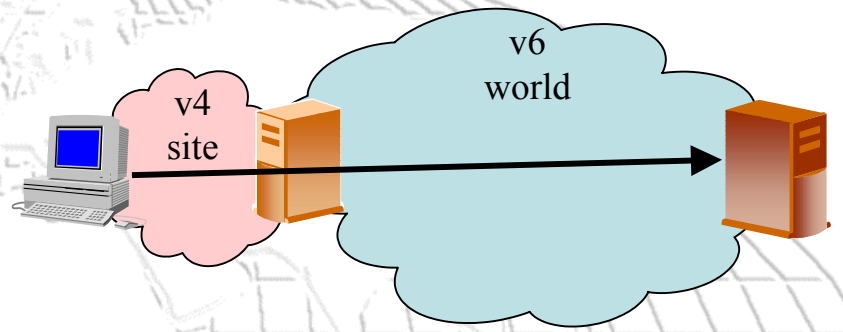
- Avoid the P in NAPT:
  More than one v4 address +
  - Assignation algorithm
    - In coordination with DNS translation
  - Explicit rules
- Servers behind the NAT-PT
  - Explicit rules
  - Coordination with DNS translation
- Not only v6 site to v4 world…

Translation. NAT-PT

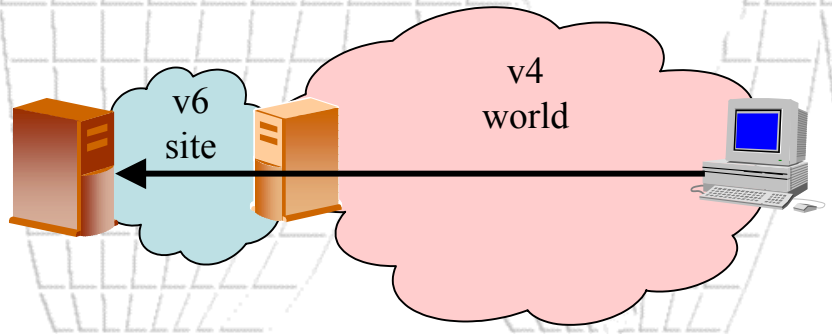
Four situations



➔Simple automatic v6 to v4 NAPT-PT

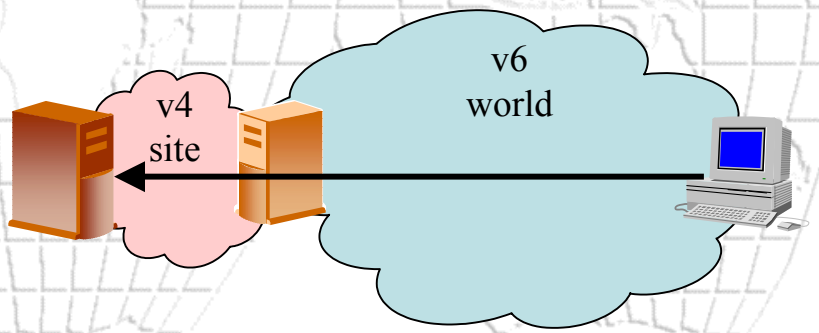


➔Simple automatic v4 to v6 NAPT-PT



➔Explicit rule v4 to v6

or dynamic with DNS translation



➔Explicit rule v6 to v4

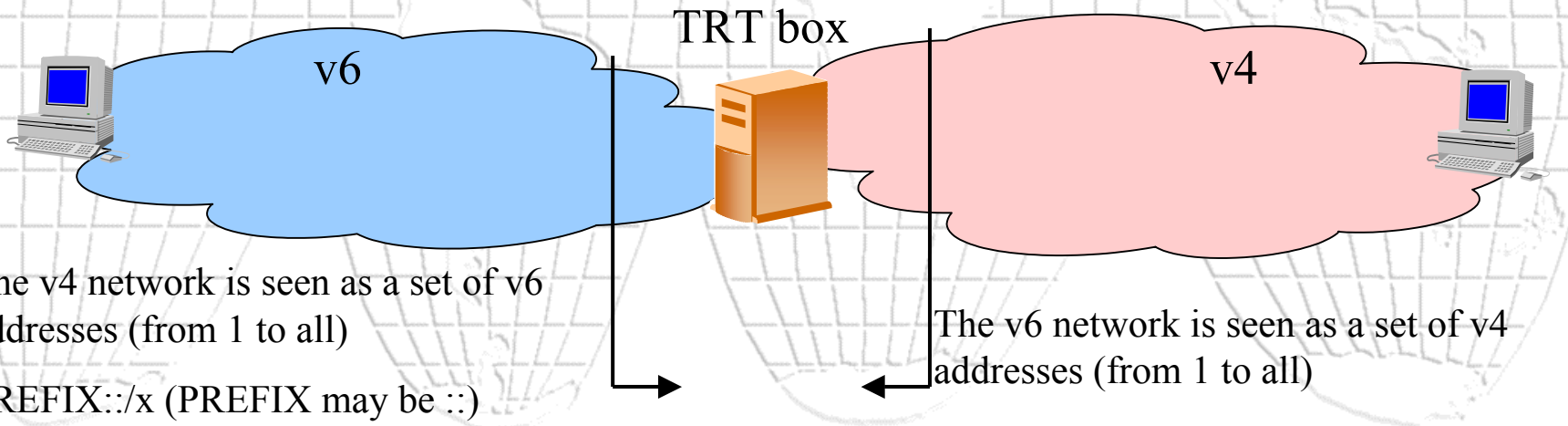
or dynamic with DNS translation

Translation. NAT-PT

- Single failure point
- Scalability problems?
- Not usable for protocols/applications that use addresses in application layer

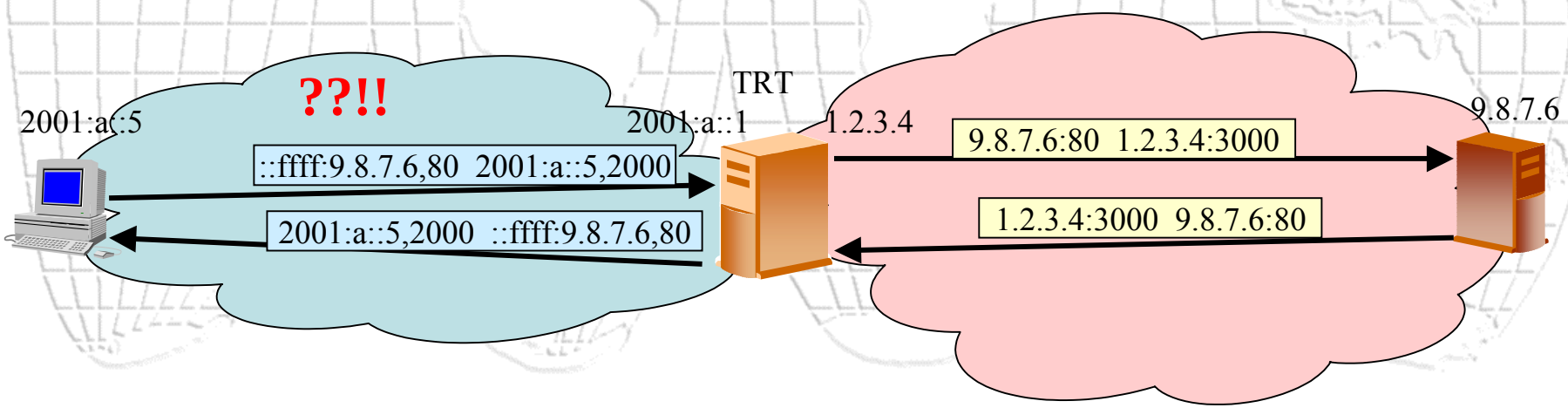
Translation. TRT

- Transport Relay Translator, RFC 3142
- Similar to a “generic application layer gateway” ➔ each side of the translation is a different TCP/UDP connection
- Similar to NAT-PT, but one connection each side



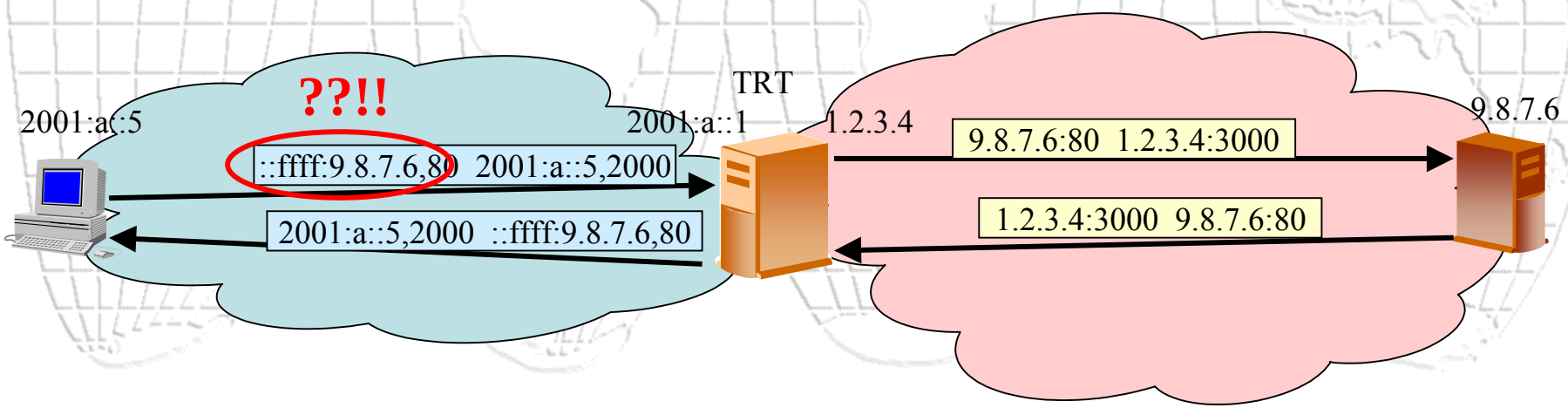
Translation. TRT

- The client connects to ::ffff:9.8.7.6,80
- TRT ends the connection and creates a new TCP connection to 9.8.7.6
- UDP connection? ;-)
- Explicit rules for servers behind the TRT

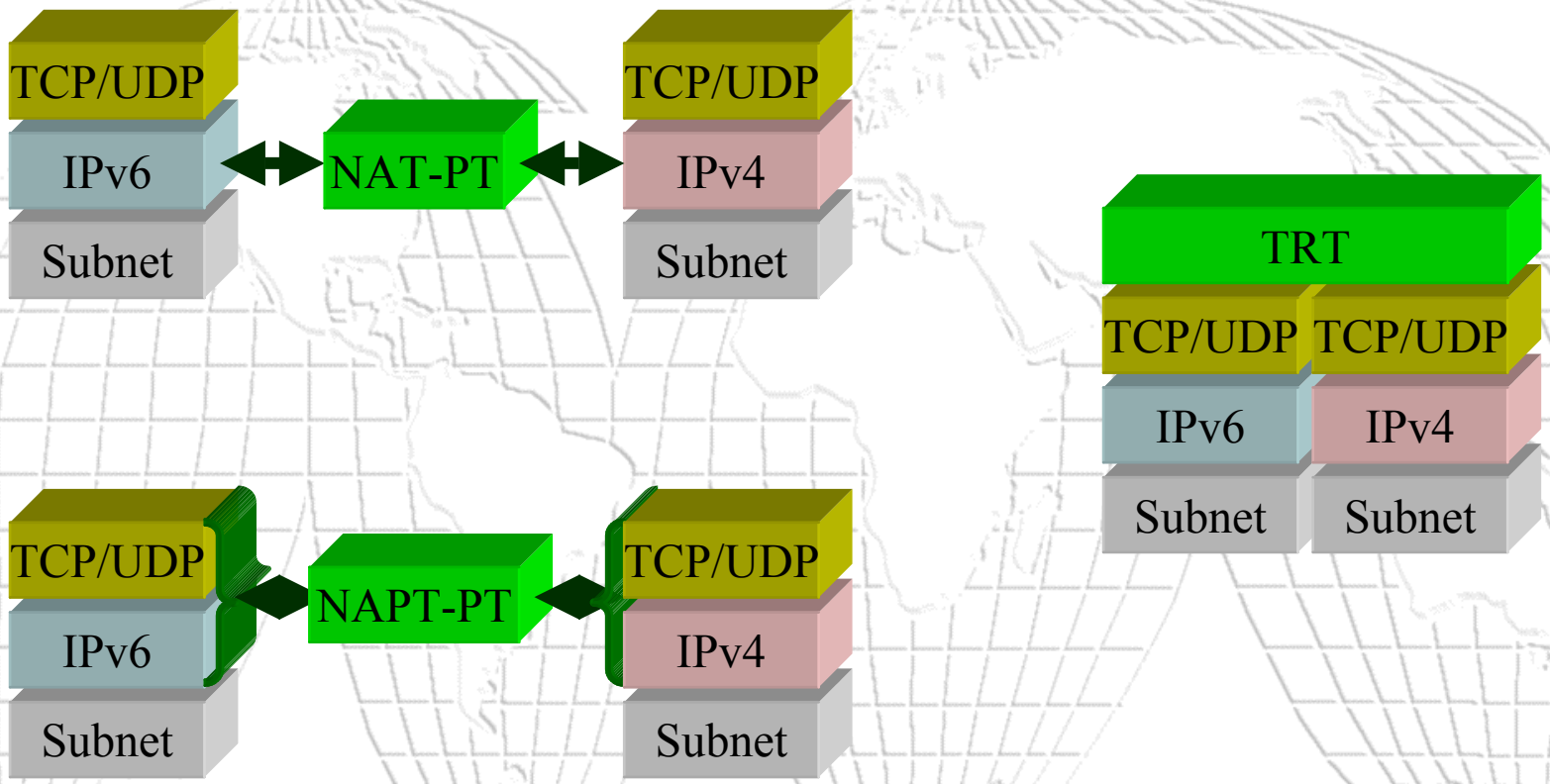


Translation. TRT

- The client connects to ::ffff:9.8.7.6,80
- TRT ends the connection and creates a new TCP connection to 9.8.7.6
- UDP connection? ;-)
- Explicit rules for servers behind the TRT



Translation. TRT ↔ NAT-PT



Translation. Others

- BIA (Bump In the API):
  - Translator layer 4.5, to deceive the `socket()` API
  - Includes changes in the resolver, to use a temporary IPv4 address
- BIS (Bump In the Stack):
  - Translator layer 2.5, to deceive the upper layers
  - Includes changes in the resolver to use a temporary IPv4 address
- mBIS (multicast BIS)
- Socks64:
  - If the application is “socksified”, use a dual-stacked socks server

Index

- Introduction
- Dual stack
- Tunneling
- Translation
- Conclusions

Conclusions. Choosing

- Implications on Applications:
  Whether they have to be modified or not
- IPv4 address requirements
  How many IPv4 addresses are required to implement the mechanism
- Host/Site mechanism
  If the mechanism is designed for isolated hosts or complete sites or both
- Scalability
  How the mechanism scales to larger sites or higher traffic

"Transition scenarios" (Alberto García)

Conclusions

- Tens (tons) of transition mechanism
  - A few general solutions
- The transition is possible:
  - Incremental deployment
  - Interoperability v4 – v6
  - Scalable solutions
- Warning: network transition + application porting
- Keep attending:
  - “Transition scenarios”, Alberto García
  - “Application porting”, Tomás de Miguel, Eva Castro
  - Demos in the demo room

Transition Mechanisms

Javier Sedano (javier.sedano@agora-2000.com)
David Fernández (david@dit.upm.es)