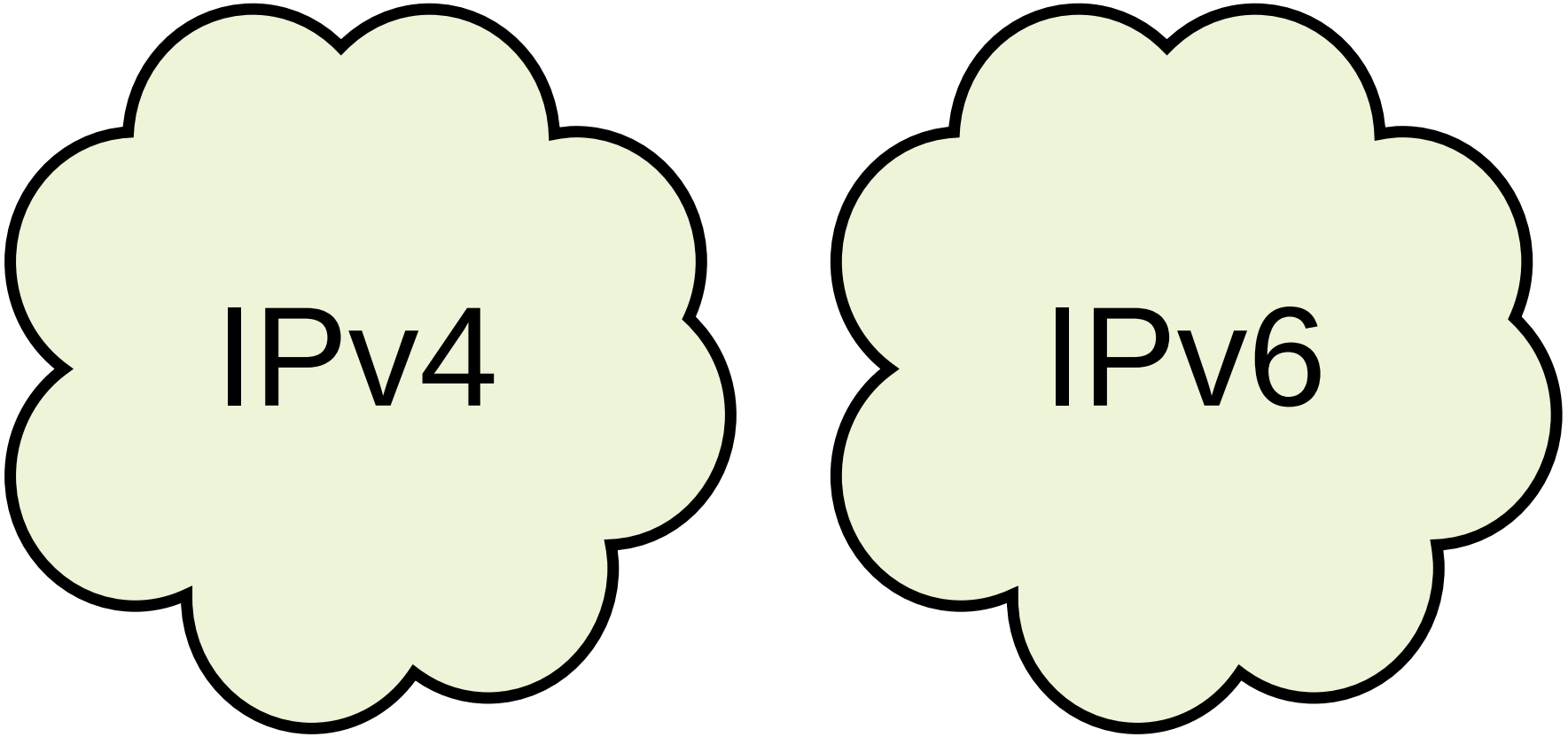
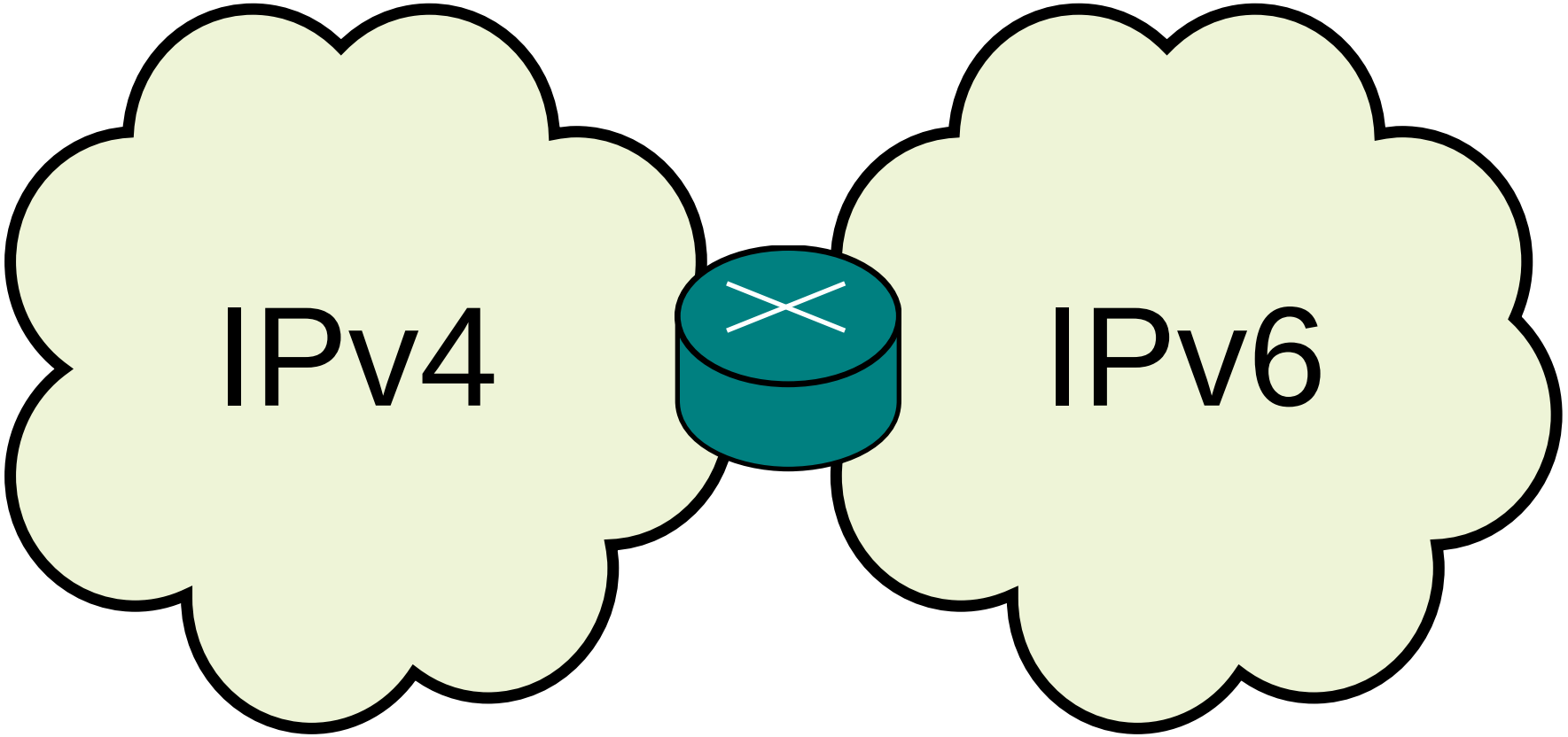
Implementando SIIT / NAT64 usando Jool

lacnic webinars

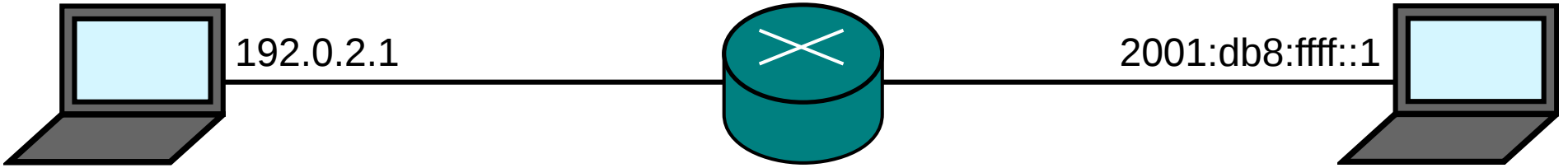
Problemática



Traducción IP



Traducción IP



MAC	01:01:01:01:01:01 02:02:02:02:02:02	03:03:03:03:03:03 04:04:04:04:04:04
IP	192.0.2.1 10.0.0.4	2001:db8:aaaa::1 2001:db8:ffff::1
PORT	6123 80	6123 80
DATA	Hola	Hola

Traducción IP/ICMP

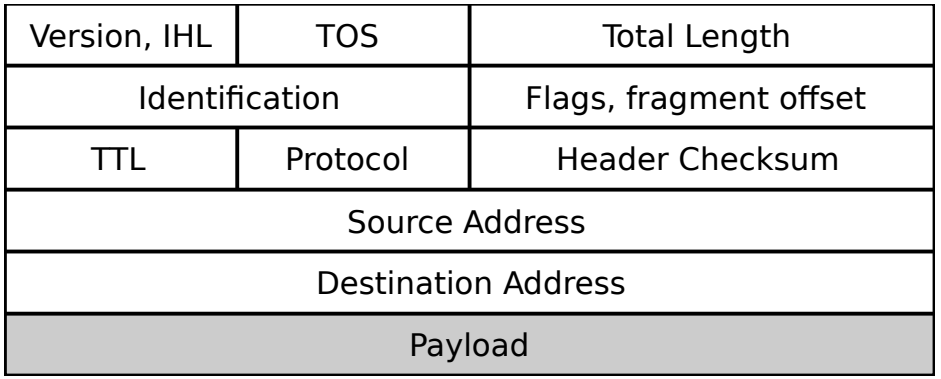
- Traducción de headers:
  - RFC 7915
- Traducción de direcciones:
  - RFC 6052 (Prefijo simple - tradicional)
  - RFC 6146 (Stateful NAT64)
  - RFC 7757 (Explicit Address Mappings)
  - RFC 6791 (Hack para ICMP errors)
  - RFC 7599 (MAP-T)

Traducción IP

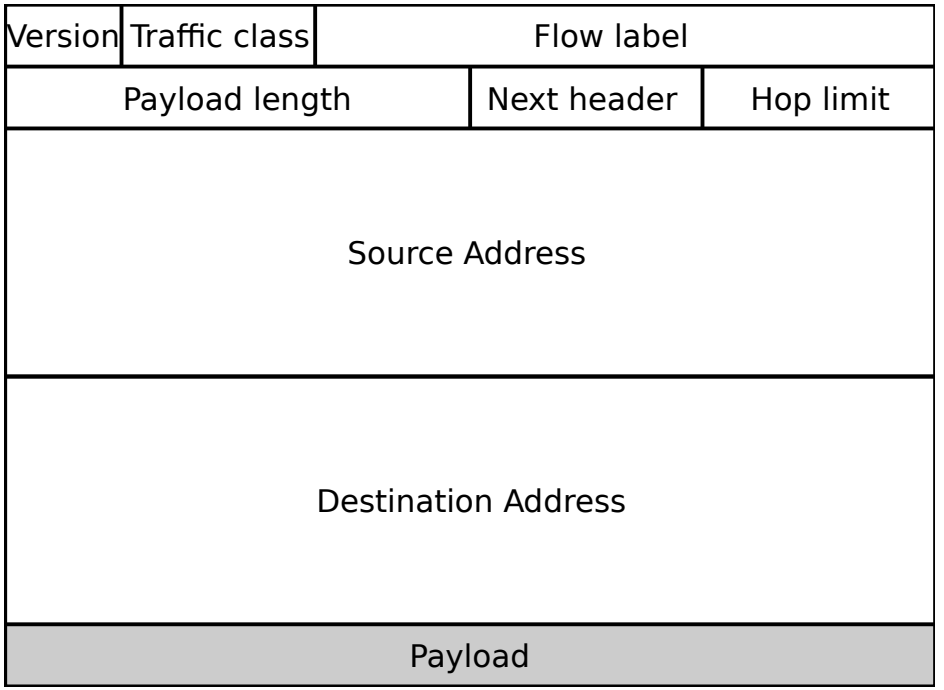
- SIIT
  - “Stateless NAT64”
  - Traductor de capa 3
- “NAT64”
  - Stateful NAT64
  - Traductor de capas 3 y 4

Traducción de encabezados

IPv4

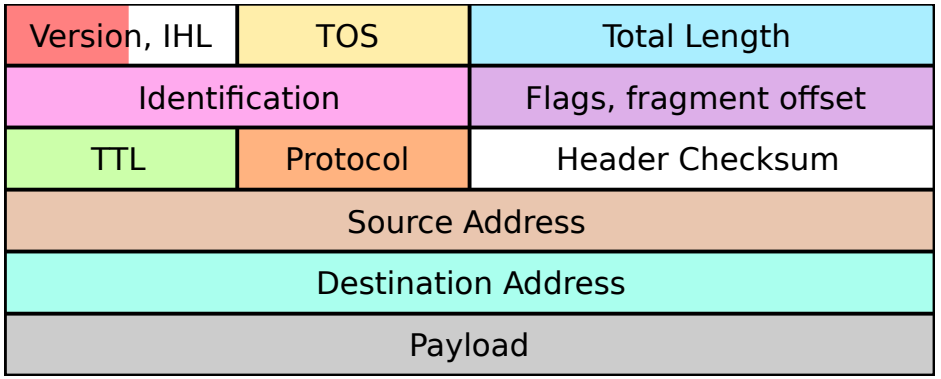


IPv6

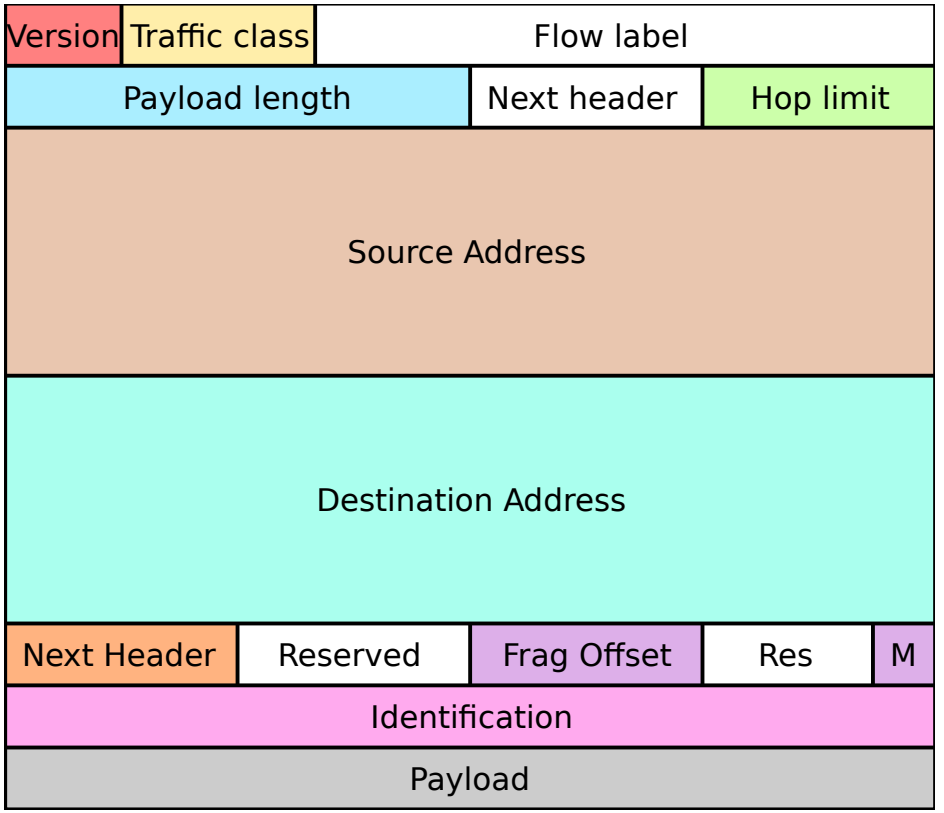


Traducción de encabezados

IPv4



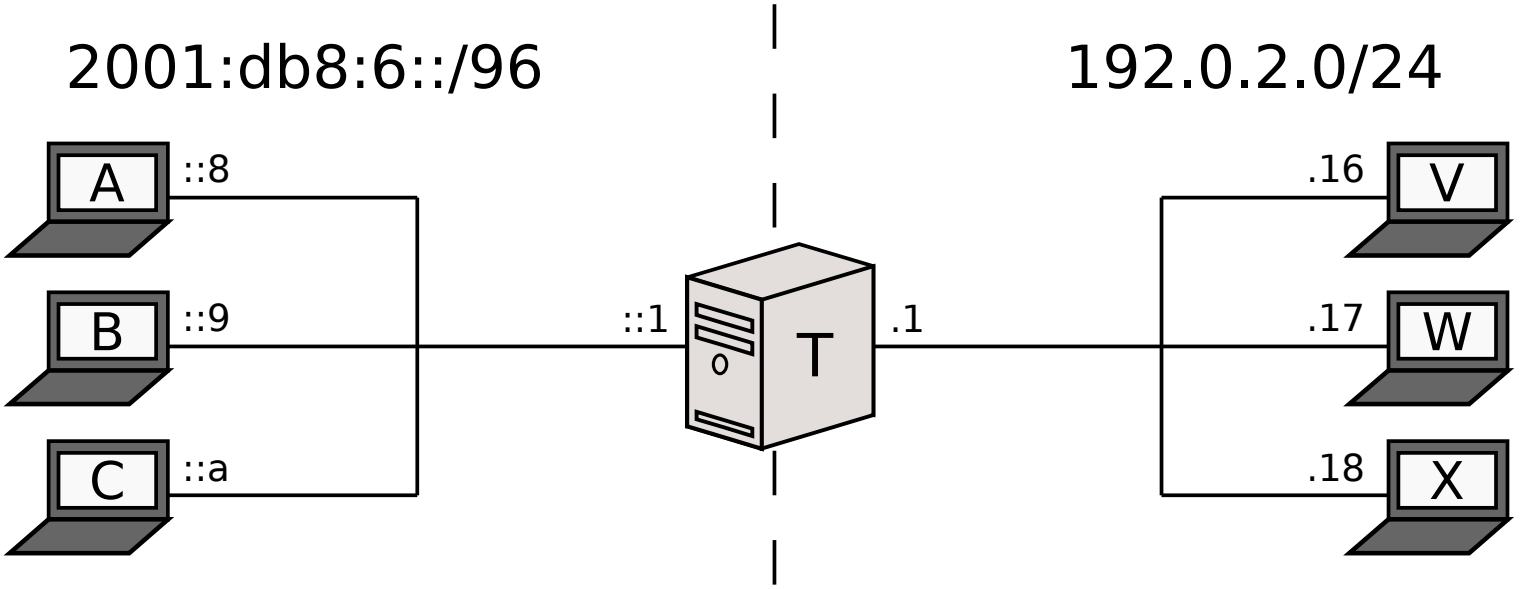
IPv6



Terminología

- SIIT
  - “Stateless NAT64”
- “NAT64”
  - Stateful NAT64

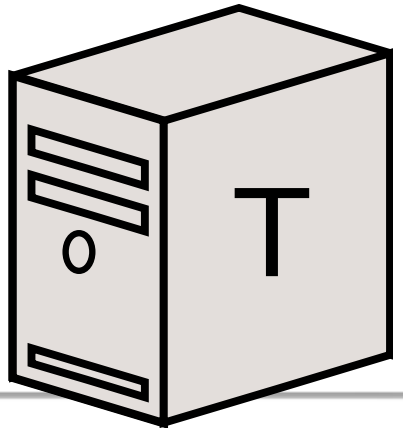
SIIT-EAM



SIIT-EAM

Voy a engañar a los nodos IPv4.
Ellos van a creer que los nodos
2001:db8:6::/120 se llaman 10.0.0.0/24.

También voy a engañar a los nodos de IPv6.
Ellos van a creer que los nodos
192.0.2.0/24 se llaman 2001:db8:4::/120.



2001:db8:6::/120 ↔ 10.0.0.0/24

- /120 y /24 indican prefijo.
- 10.0.0.0/24 → 10.0.0.0
- 2001:db8:6::/120 → 2001:0db8:0006:0000:0000:0000:0000:0000
- La idea de EAM es reemplazar prefijos.
- Por ejemplo, la dirección 10.0.0.5:
  - Quitamos prefijo de IPv4: .5
  - Agregamos prefijo de IPv6: 2001:db8:6::5
  - Done.

2001:db8:6::/120 ↔ 10.0.0.0/24

2001:db8:6::1 ↔ 10.0.0.1

2001:db8:6::2 ↔ 10.0.0.2

2001:db8:6::3 ↔ 10.0.0.3

...

2001:db8:6::9 ↔ 10.0.0.9

2001:db8:6::a ↔ 10.0.0.10

2001:db8:6::b ↔ 10.0.0.11

...

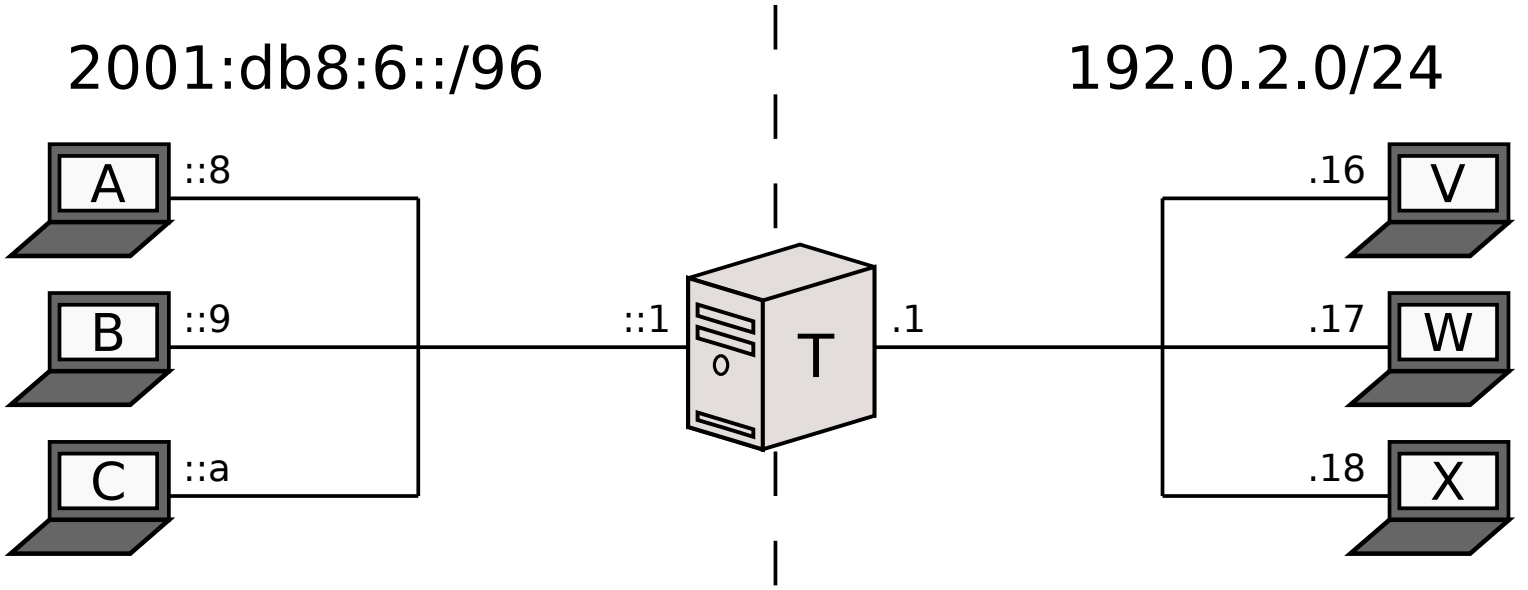
2001:db8:6::ef ↔ 10.0.0.254

2001:db8:6::ff ↔ 10.0.0.255

Tabla EAM

IPv6	IPv4
2001:db8:6::/120	10.0.0.0/24
2001:db8:4::/120	192.0.2.0/24

SIIT-EAM

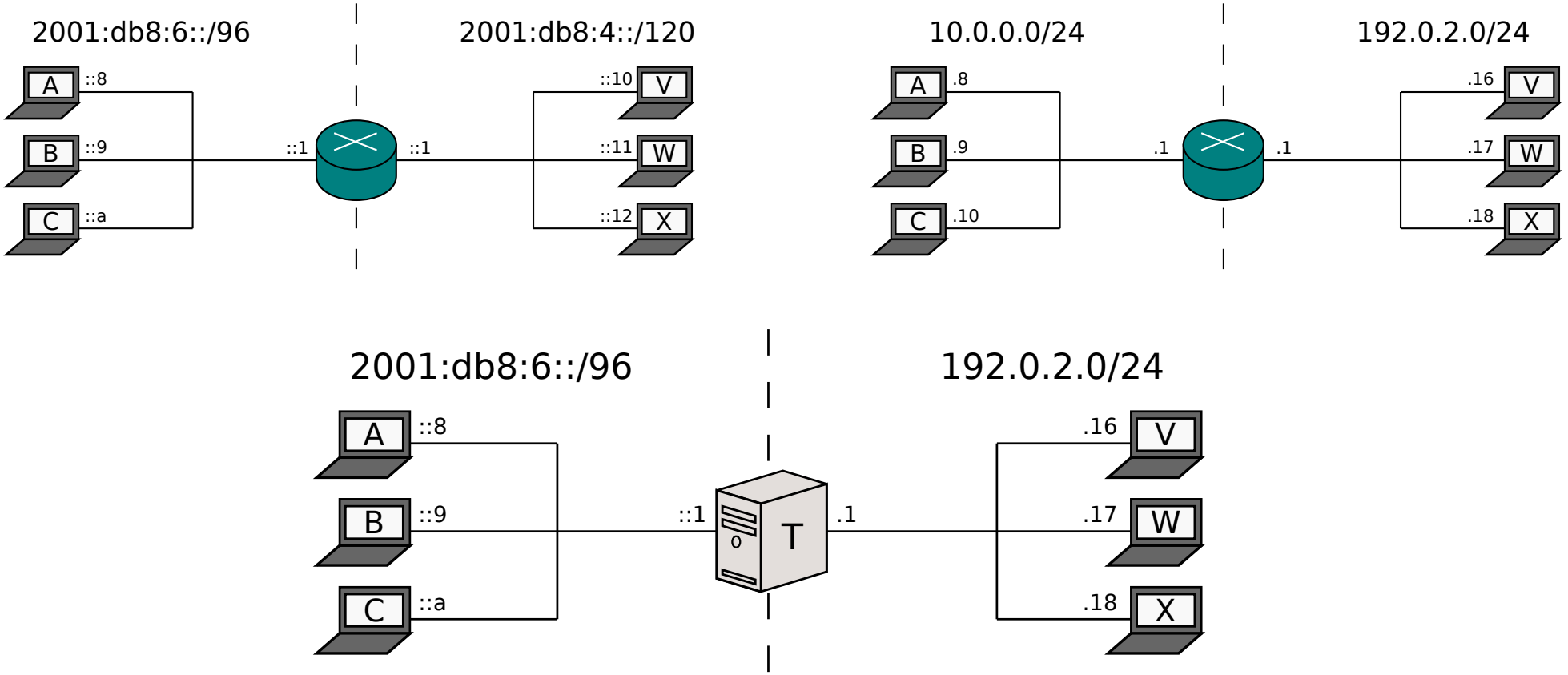


2001:db8:6::8
2001:db8:4::11

10.0.0.8
192.0.2.17

IPv6	IPv4
2001:db8:6::/120	10.0.0.0/24
2001:db8:4::/120	192.0.2.0/24

SIIT-EAM



IPv6	IPv4
2001:db8:6::/120	10.0.0.0/24
2001:db8:4::/120	192.0.0.0/24

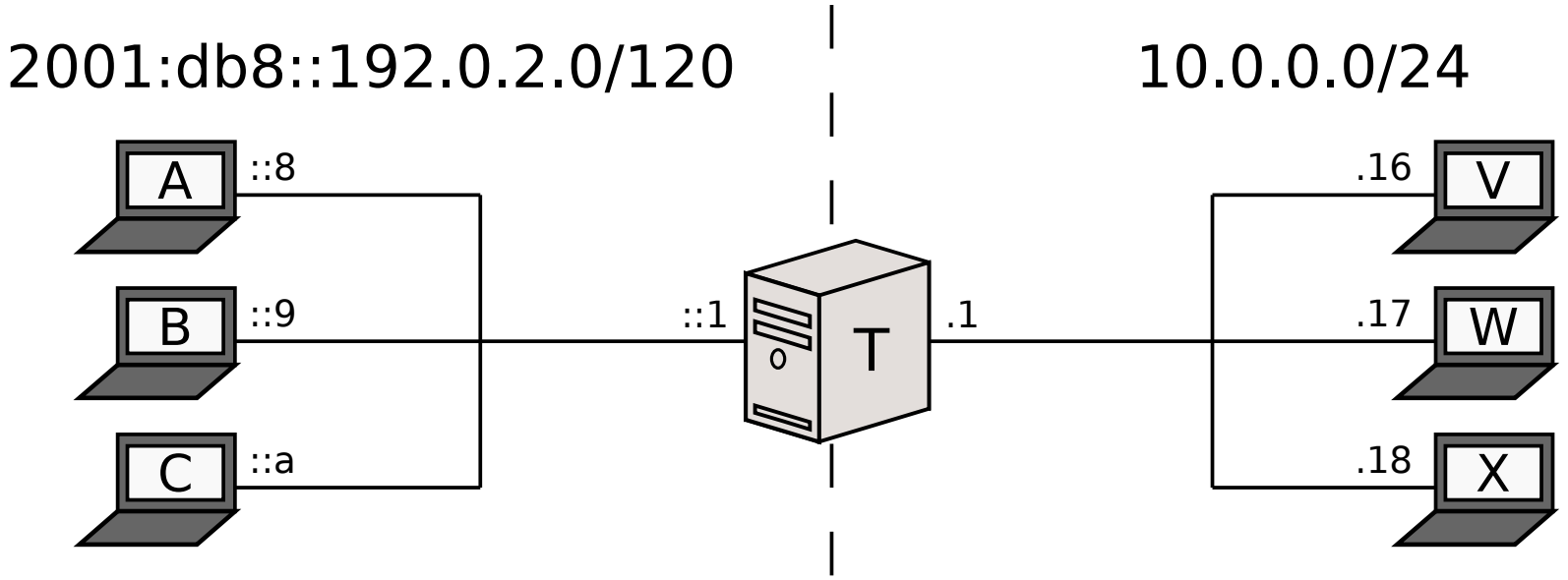
Tabla EAM (simplificada)

IPv6	IPv4
2001:db8:6::1/128	10.0.0.6/32
2001:db8:4::d5/128	192.0.2.10/32
2001:db8:aaaa:bbbb::cccc/128	198.51.100.55/32
1234:5678::abcd:ef/128	10.0.0.254/32

SIIT-tradicional

- SIIT-EAM reemplaza prefijos de ambos protocolos (IPv6/4).
  - Se basa en una tabla, por lo que podemos tener cualquier número de prefijos.
- En contraste, SIIT-tradicional solamente agrega y quita **un** prefijo de IPv6.
  - Es más restrictivo.

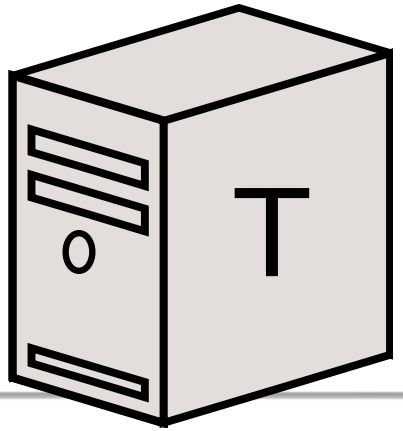
SIIT-tradicional



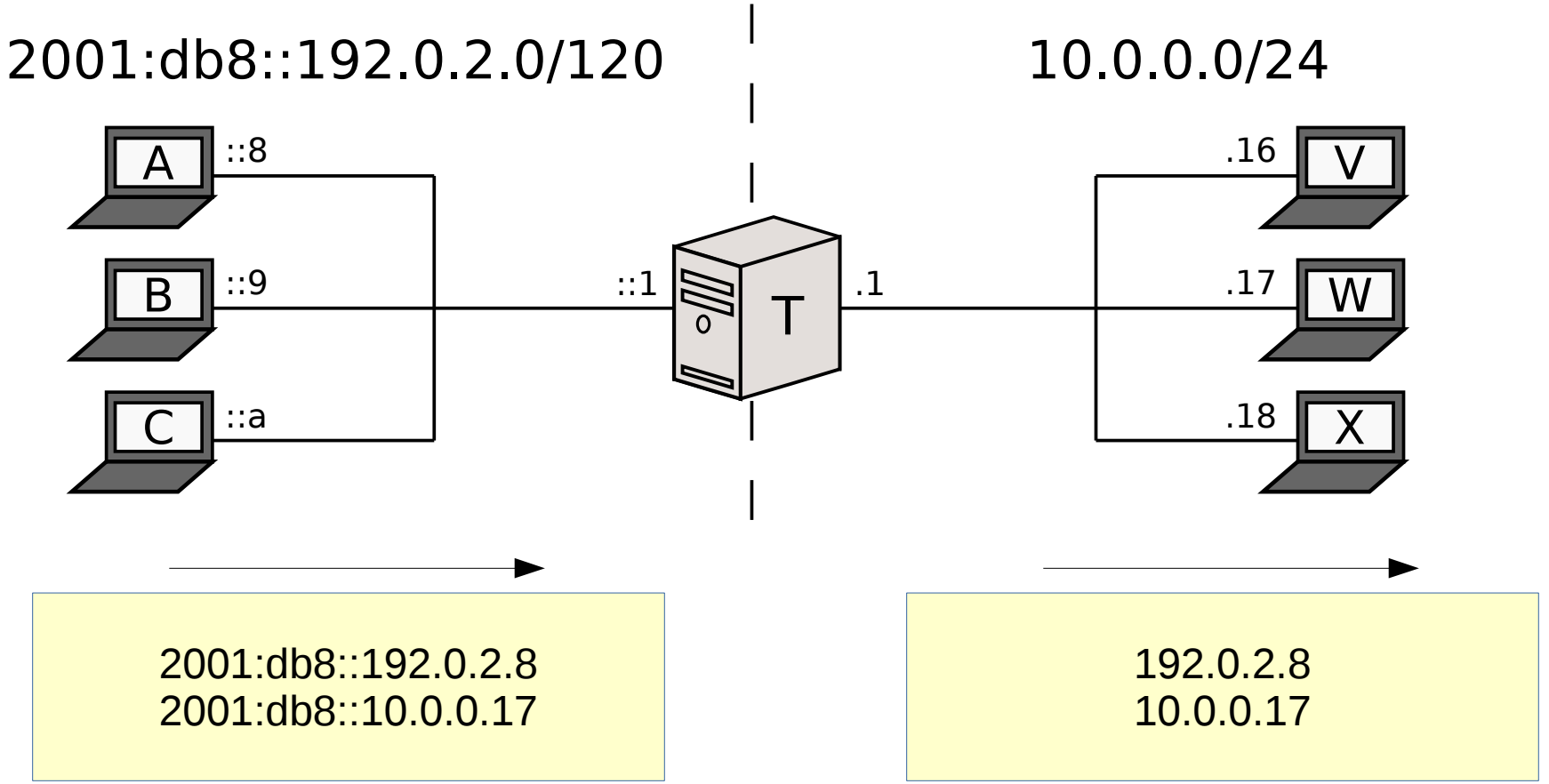
SIIT-tradicional

Cuando me toque traducir un paquete de 4 a 6,
voy a agregar el prefijo 2001:db8::/96.

Cuando me toque traducir de 6 a 4,
voy a quitar el prefijo 2001:db8::/96.

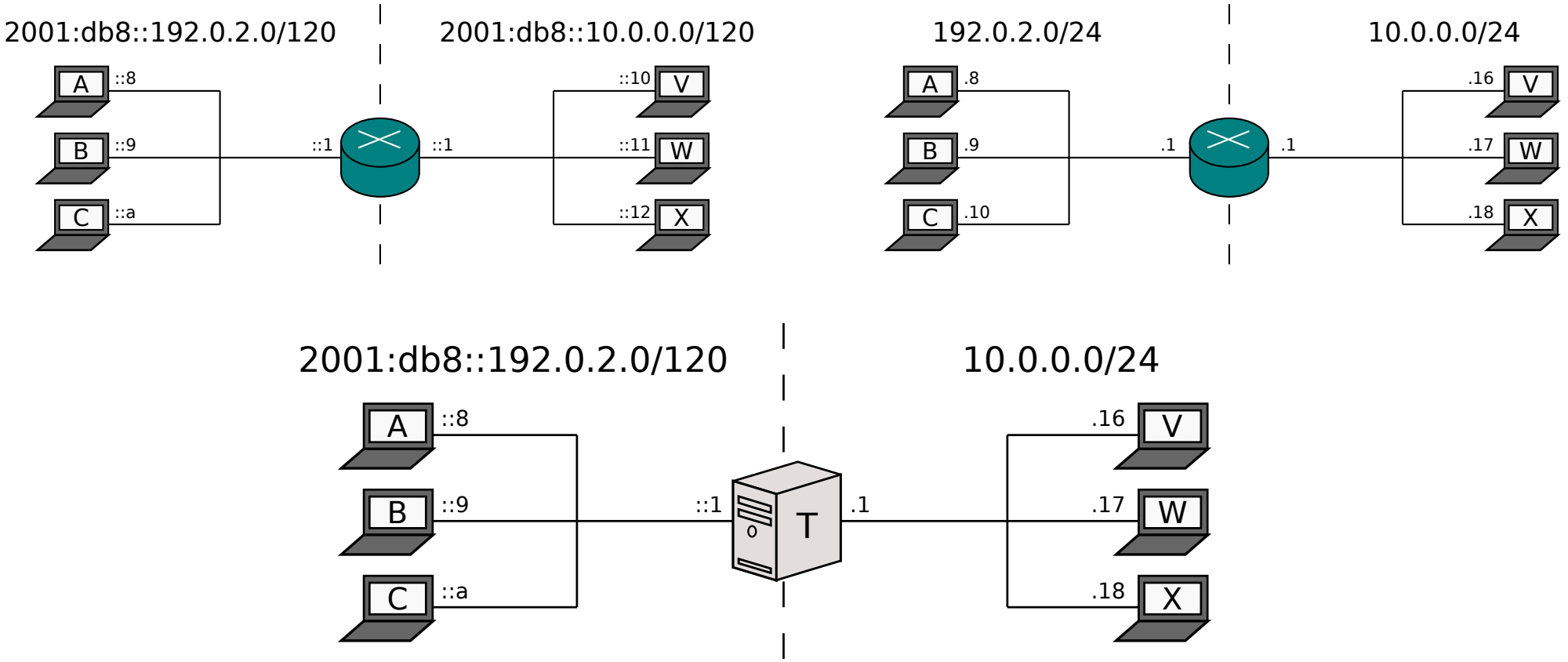


SIIT-tradicional



- Toda la dirección de IPv4 sobrevive.
- Todos los nodos de IPv6 que quieran hablar con IPv4 necesitan el prefijo y una dirección IPv4 válida dentro.

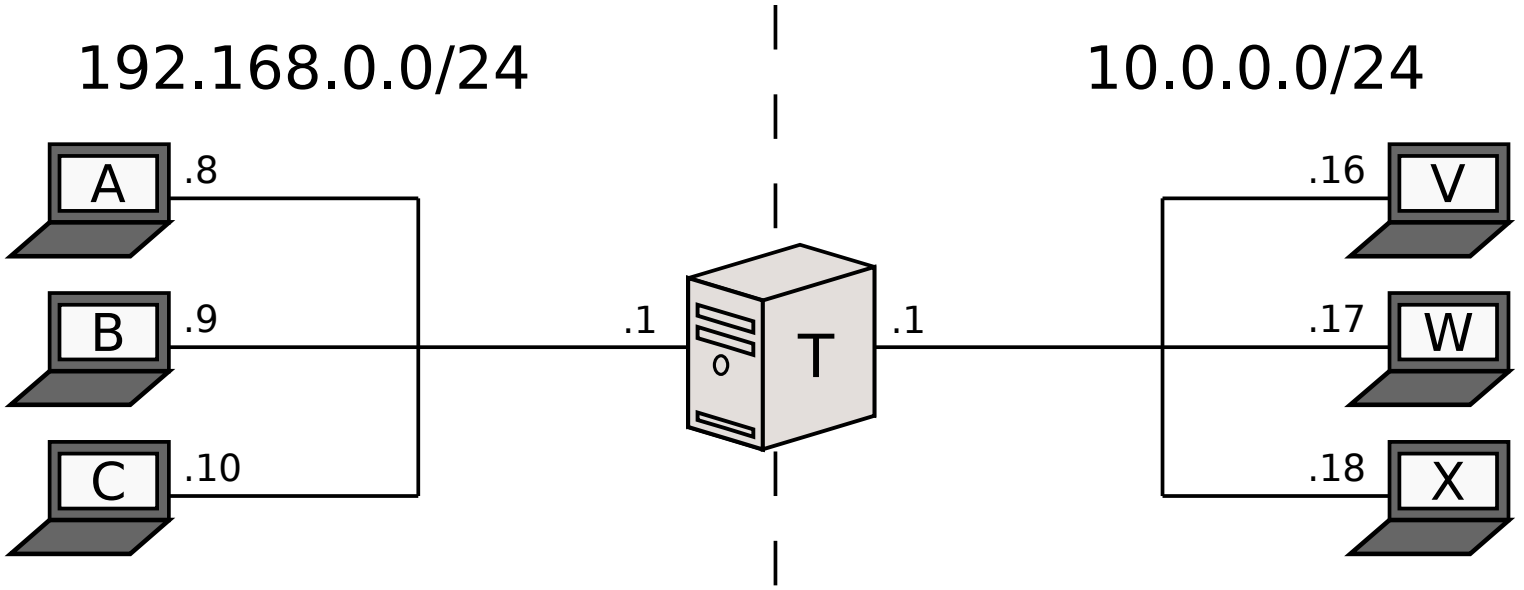
SIIT-tradicional



NAT64

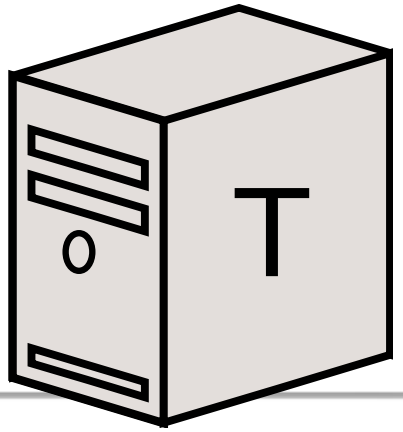
- SIIT es solo un mecanismo para permitir que IPv6 e IPv4 coexistan; no soluciona el problema del agotamiento de direcciones de IPv4 porque el mapeo de direcciones es 1 a 1.
- NAT64 combina SIIT y NAT para lograr que *n* direcciones de IPv6 puedan mapearse a *unas cuantas* direcciones de IPv4.

NAT

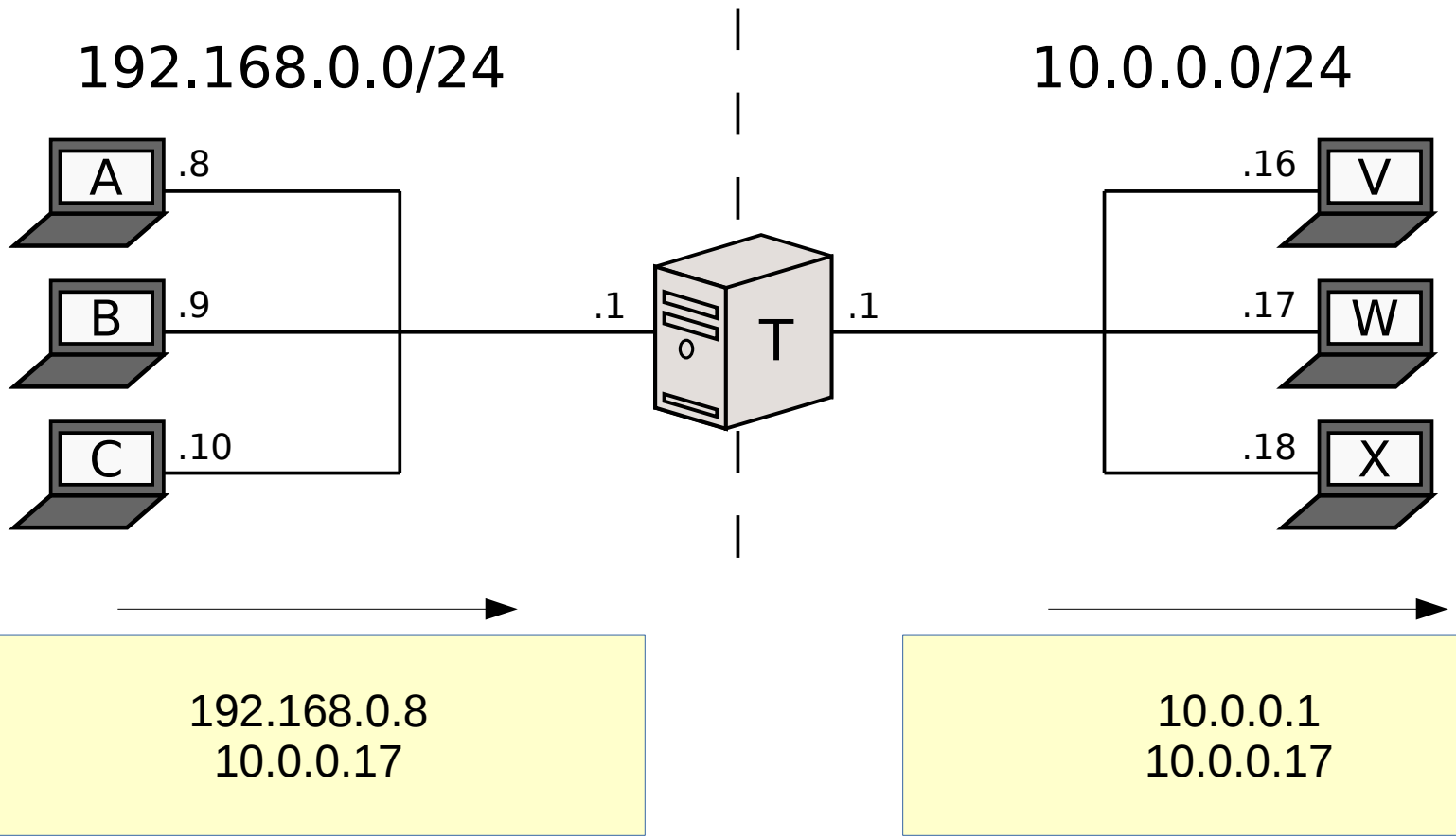


NAT

Voy a enmascarar a los nodos
de la red 192.168.0.0/24
Con mi propia dirección de IPv4.

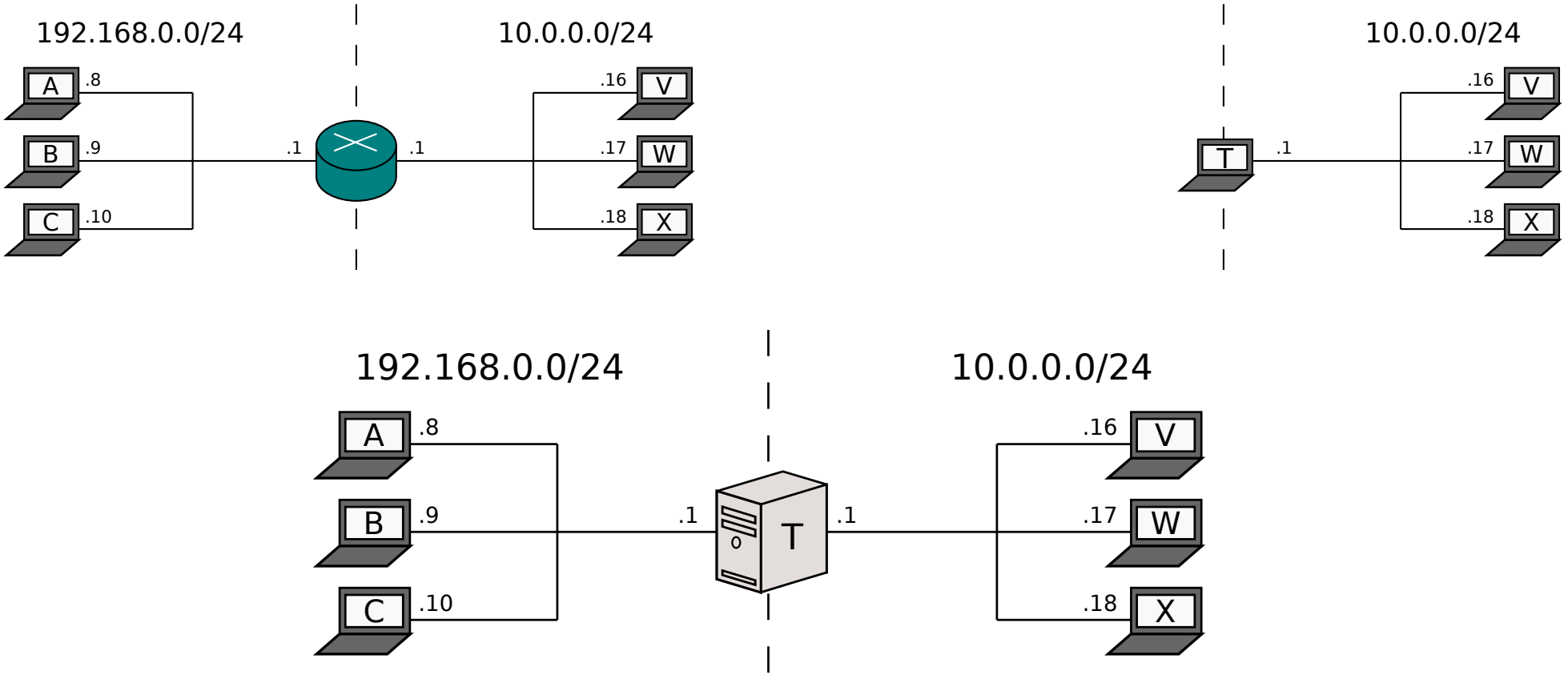


NAT

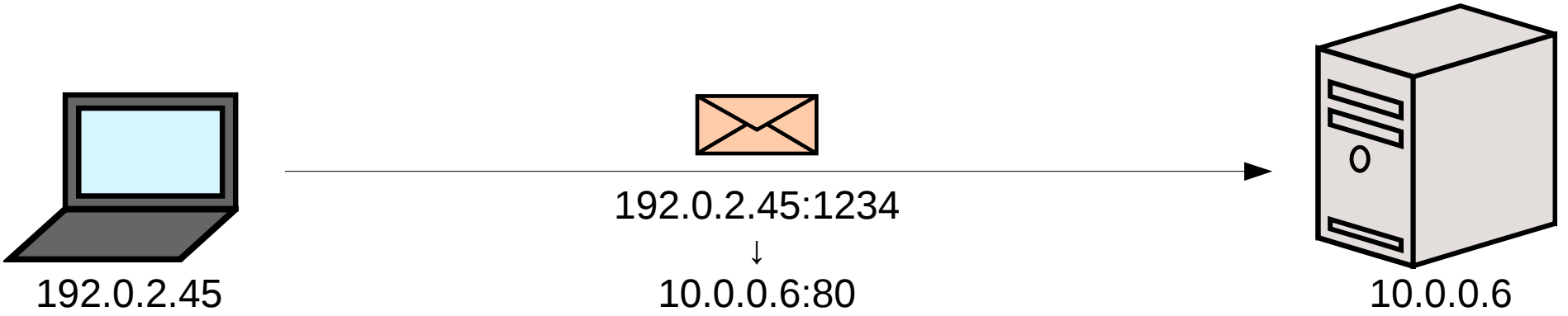


Soluciona el problema de agotamiento de direcciones porque hace que A, B, C y T tengan la misma dirección a los ojos de cualquier otra red.

NAT

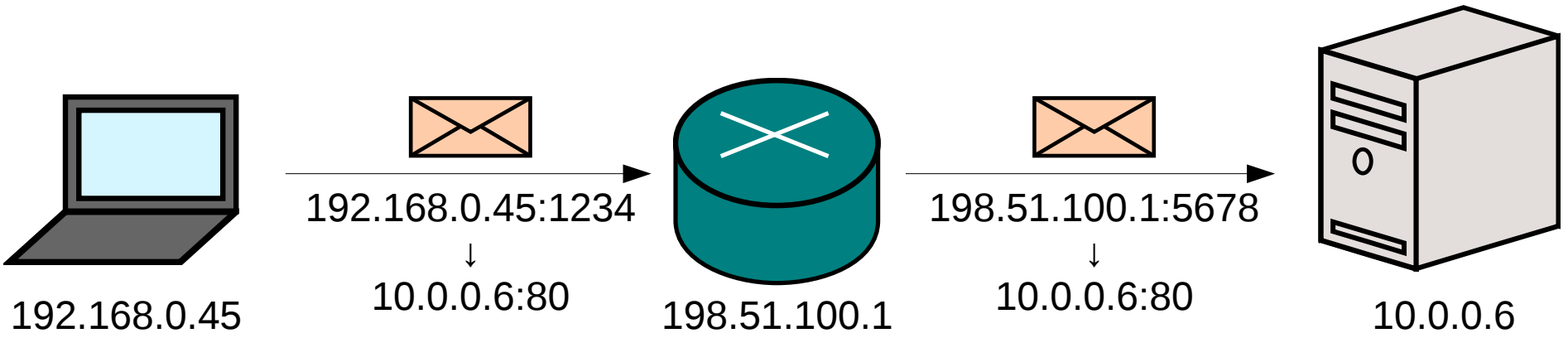

192.168.0.0/24
10.0.0.0/24
A
.8
B
.9
C
.10
.1
.1
.16
V
.17
W
.18
X
10.0.0.0/24
T
.1
.16
V
.17
W
.18
X
192.168.0.0/24
10.0.0.0/24
A
.8
B
.9
C
.10
.1
T
.1
.16
V
.17
W
.18
X

Sockets



Socket	Cliente	Servidor
Dirección	192.0.2.45	10.0.0.6
Puerto	1234	80

Sockets, NAT

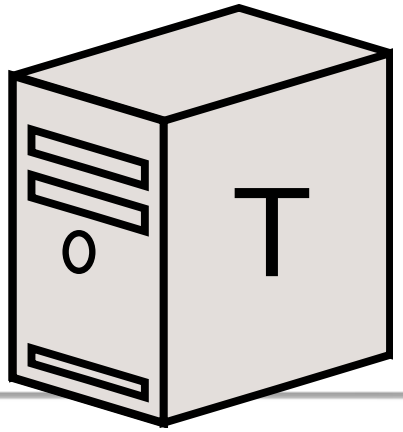


Socket	Cliente	Servidor
Dirección	192.168.0.45	10.0.0.6
Puerto	1234	80

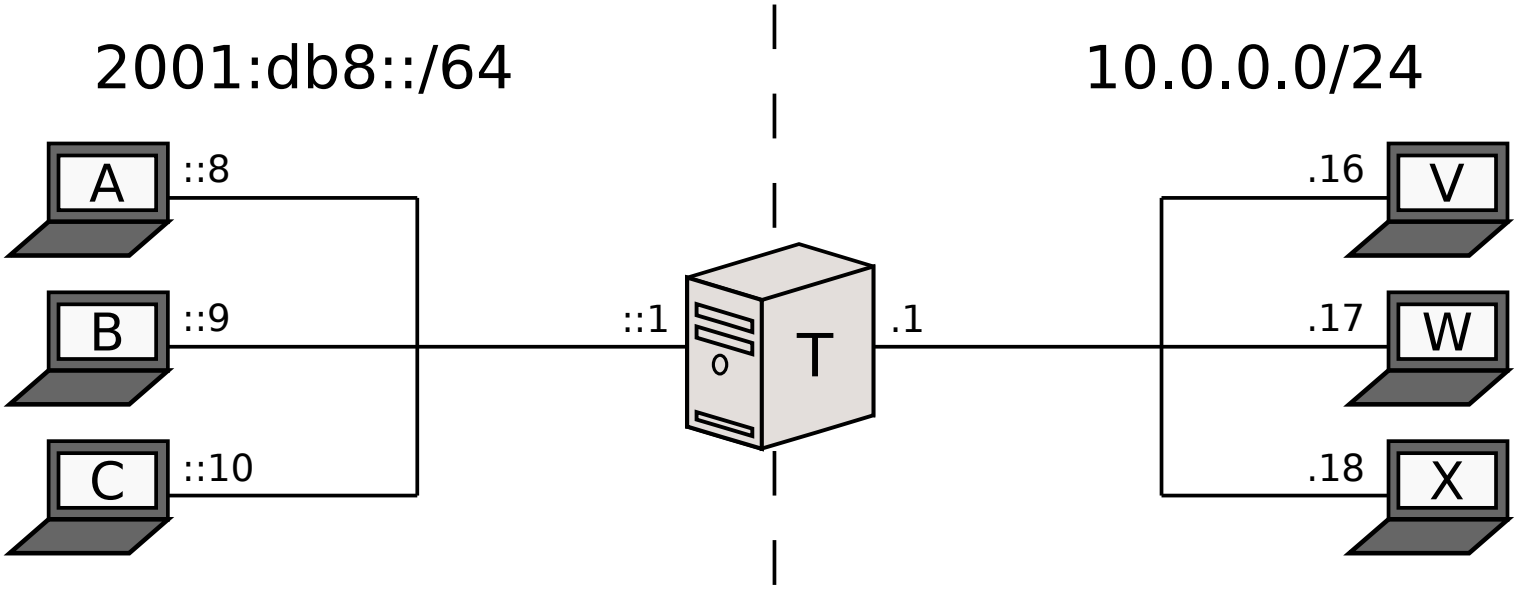
Socket	Cliente	Servidor
Dirección	198.51.100.1	10.0.0.6
Puerto	5678	80

NAT64

Voy a enmascarar a los nodos de IPv6
Con mi propia dirección de IPv4.



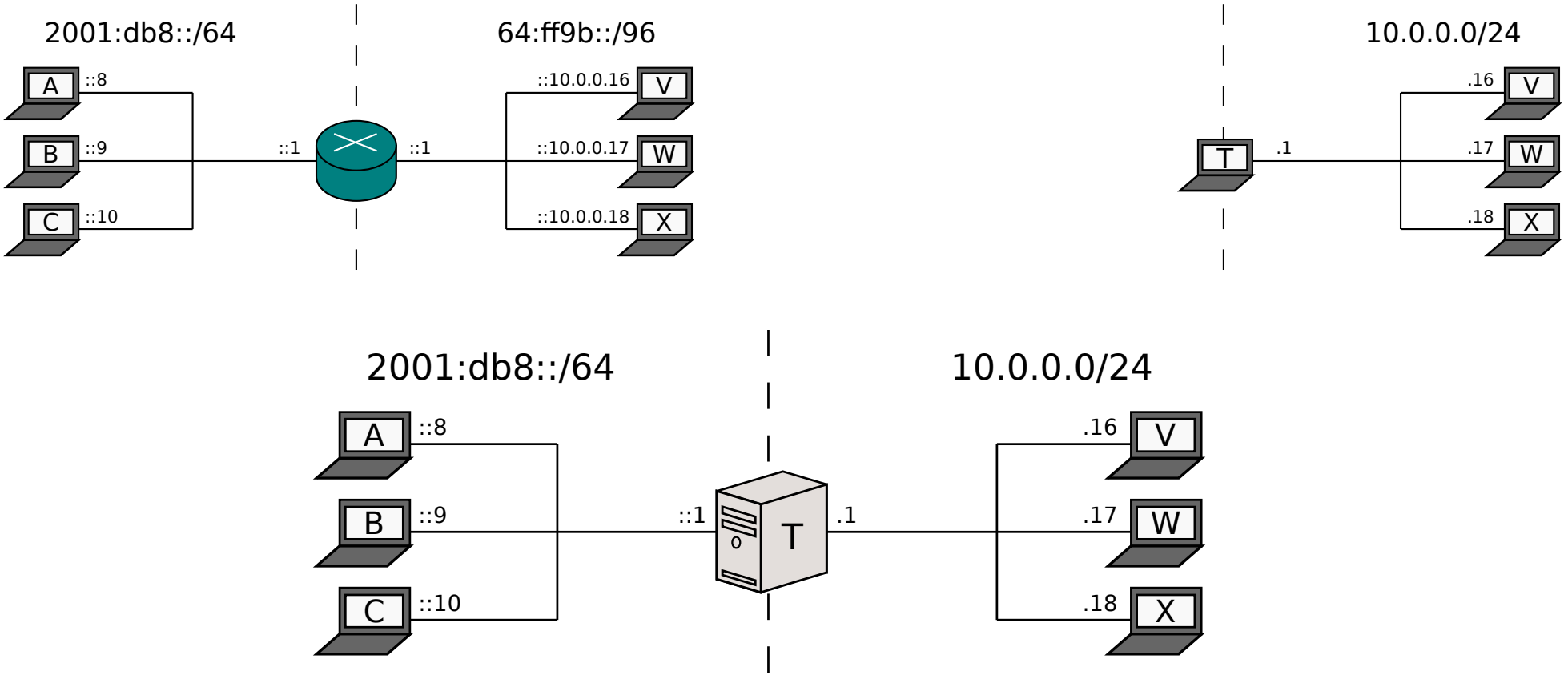
NAT64



2001:db8::8
64:ff9b::10.0.0.17

10.0.0.1
10.0.0.17

NAT64


2001:db8::/64
64:ff9b::/96
10.0.0.0/24
A
::8
B
::9
C
::10
::1
::1
::10.0.0.16
::10.0.0.17
::10.0.0.18
V
W
X
T
.1
.16
.17
.18
2001:db8::/64
10.0.0.0/24
A
::8
B
::9
C
::10
::1
T
.1
.16
.17
.18
V
W
X

NAT64

- Notar:
  - Soluciona el problema de agotamiento de direcciones porque hace que A, B, C y T tengan la misma dirección a los ojos de IPv4.
  - La dirección fuente se destruye completamente.
    - Esto implica que el NAT64 tiene que guardar una tabla que mapea los sockets.
  - La dirección destino se traduce mediante el mecanismo tradicional.

Jool

- SIIT y NAT64
- Linux (kernels 3.2 en adelante)
- Gratis y código abierto (GPLv2)
- Desarrollado en C
- https://jool.mx

Instalar Jool

- Módulos de Kernel
- Aplicaciones de configuración y control

Documentación completa y ejemplos en https://www.jool.mx

Instalar Jool: Módulos del Kernel

1.Revisar versión del kernel

```
$ /bin/uname -r
3.5.0-45-generic
```

2. Build Essentials

```
# apt-get install build-essential
```

3. Kernel headers

```
# apt-get install linux-headers-$(uname -r)
```

4. DKMS / Kbuild

```
$ unzip Jool-<version>.zip
# dkms install Jool-<version>
```

```
$ unzip Jool-<version>.zip
$ cd Jool-<version>/mod
$ make
# make install
```

Instalar Jool: Aplicaciones

1. Build Essentials

```
# apt-get install gcc make pkg-config
```

2. libnl-genl-3

```
# apt-get install libnl-genl-3-dev
```

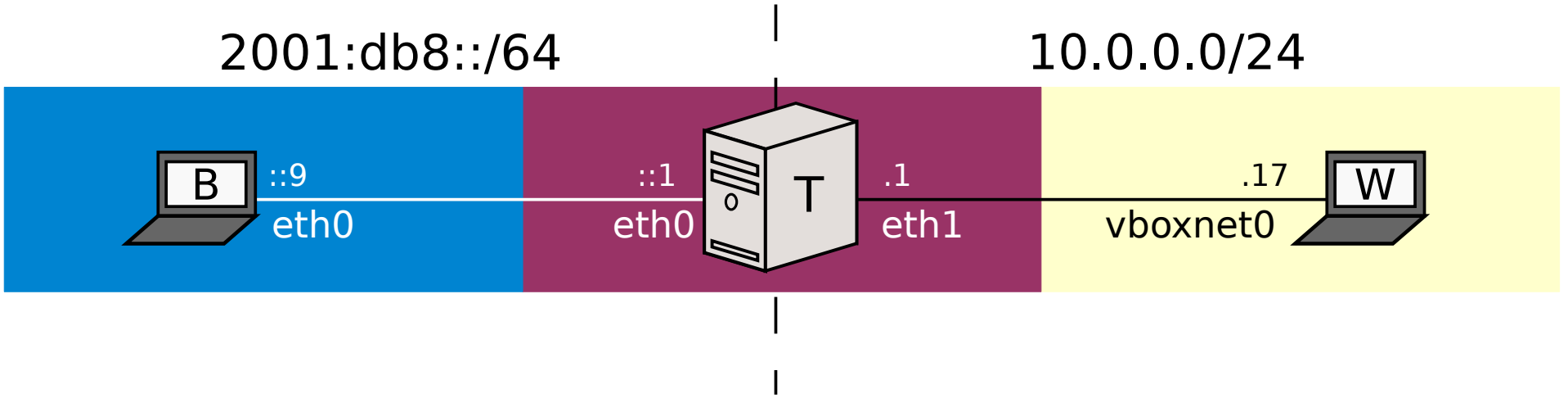
3. Autoconf

```
# apt-get install autoconf
```

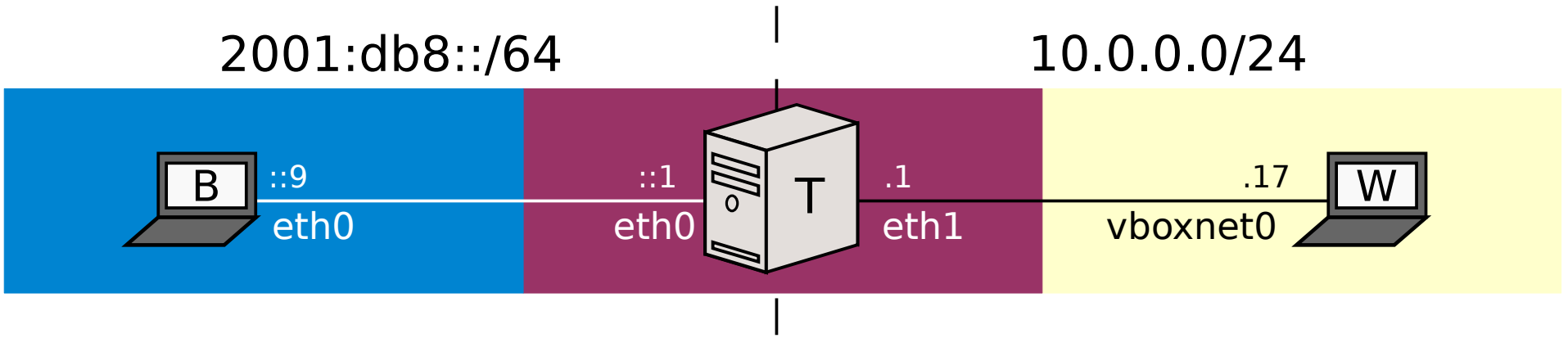
4. Compilar e Instalar

```
$ unzip Jool-<version>.zip
$ cd Jool-<version>/usr
$ ./configure
$ make
# make install
```

NAT64 (demo)

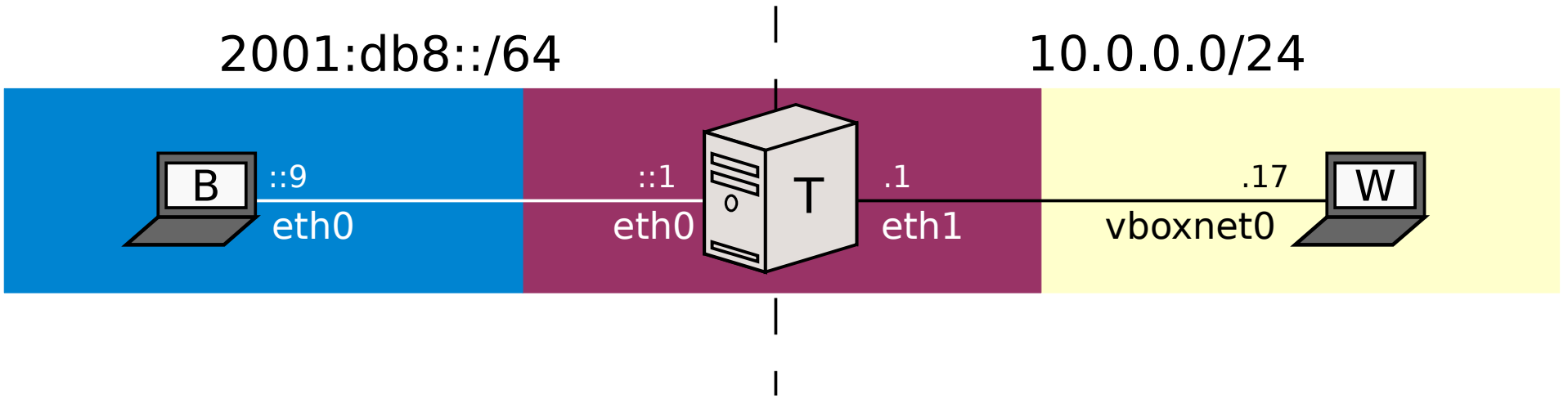


NAT64 (demo)




```
user@T:~# service network-manager stop
user@T:~#
user@T:~# /sbin/ip link set eth0 up
user@T:~# /sbin/ip address add 2001:db8::1/96 dev eth0
user@T:~#
user@T:~# /sbin/ip link set eth1 up
user@T:~# /sbin/ip address add 10.0.0.1/24 dev eth1
user@T:~#
user@T:~# sysctl -w net.ipv4.conf.all.forwarding=1
user@T:~# sysctl -w net.ipv6.conf.all.forwarding=1
user@T:~#
user@T:~# ethtool --offload eth0 gro off
user@T:~# ethtool --offload eth0 lro off
user@T:~# ethtool --offload eth1 gro off
user@T:~# ethtool --offload eth1 lro off
```

NAT64 (demo)



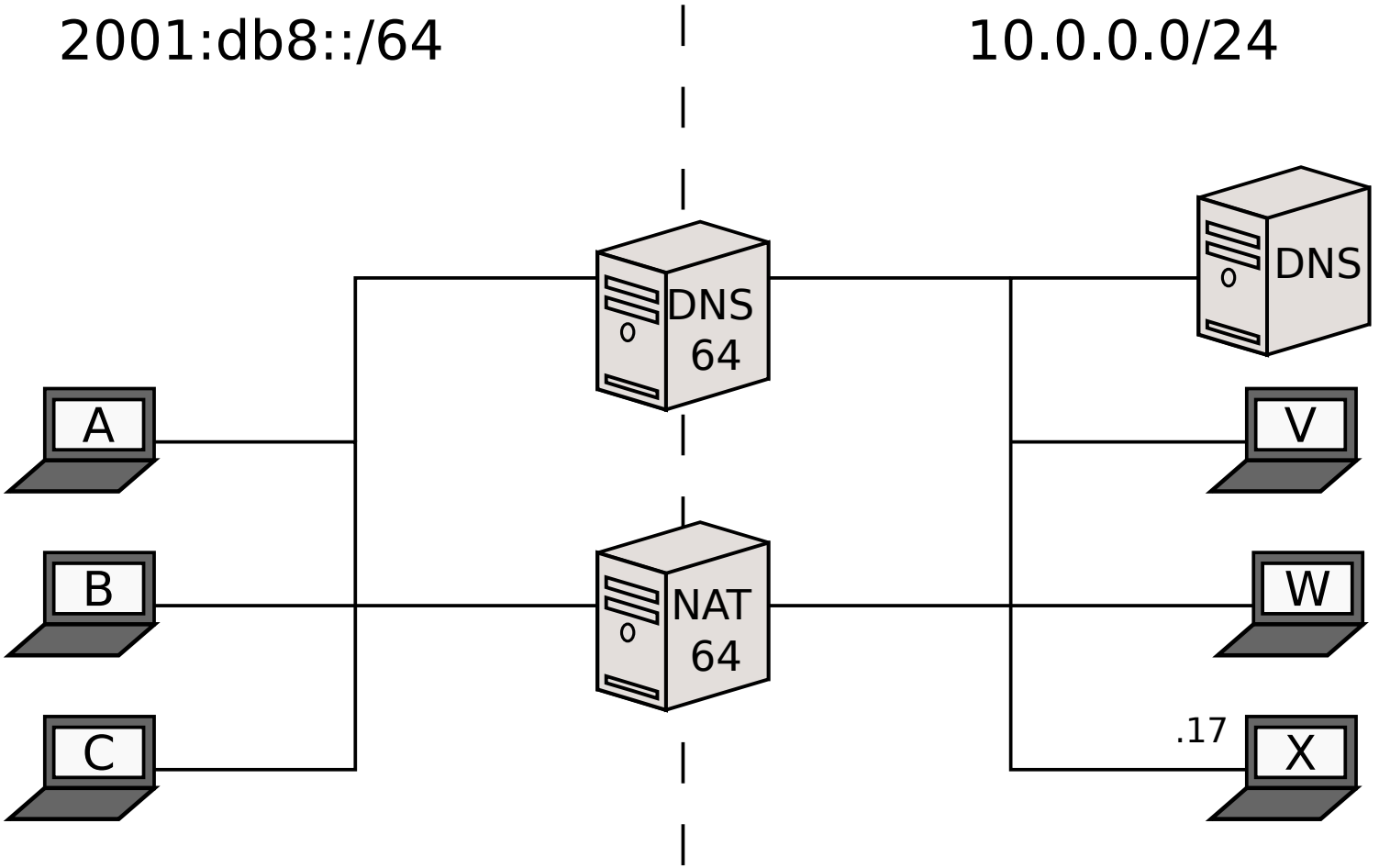

```
user@T:~# /sbin/modprobe jool pool6=64:ff9b::/96


user@B:~$ ping6 64:ff9b::10.0.0.17
PING 64:ff9b::10.0.0.17(64:ff9b::a00:11) 56 data bytes
64 bytes from 64:ff9b::a00:11: icmp_seq=1 ttl=63 time=1.13 ms
64 bytes from 64:ff9b::a00:11: icmp_seq=2 ttl=63 time=4.48 ms
64 bytes from 64:ff9b::a00:11: icmp_seq=3 ttl=63 time=15.6 ms
64 bytes from 64:ff9b::a00:11: icmp_seq=4 ttl=63 time=4.89 ms
^C
--- 64:ff9b::10.0.0.17 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 1.136/6.528/15.603/5.438 ms
```

Jool en VM

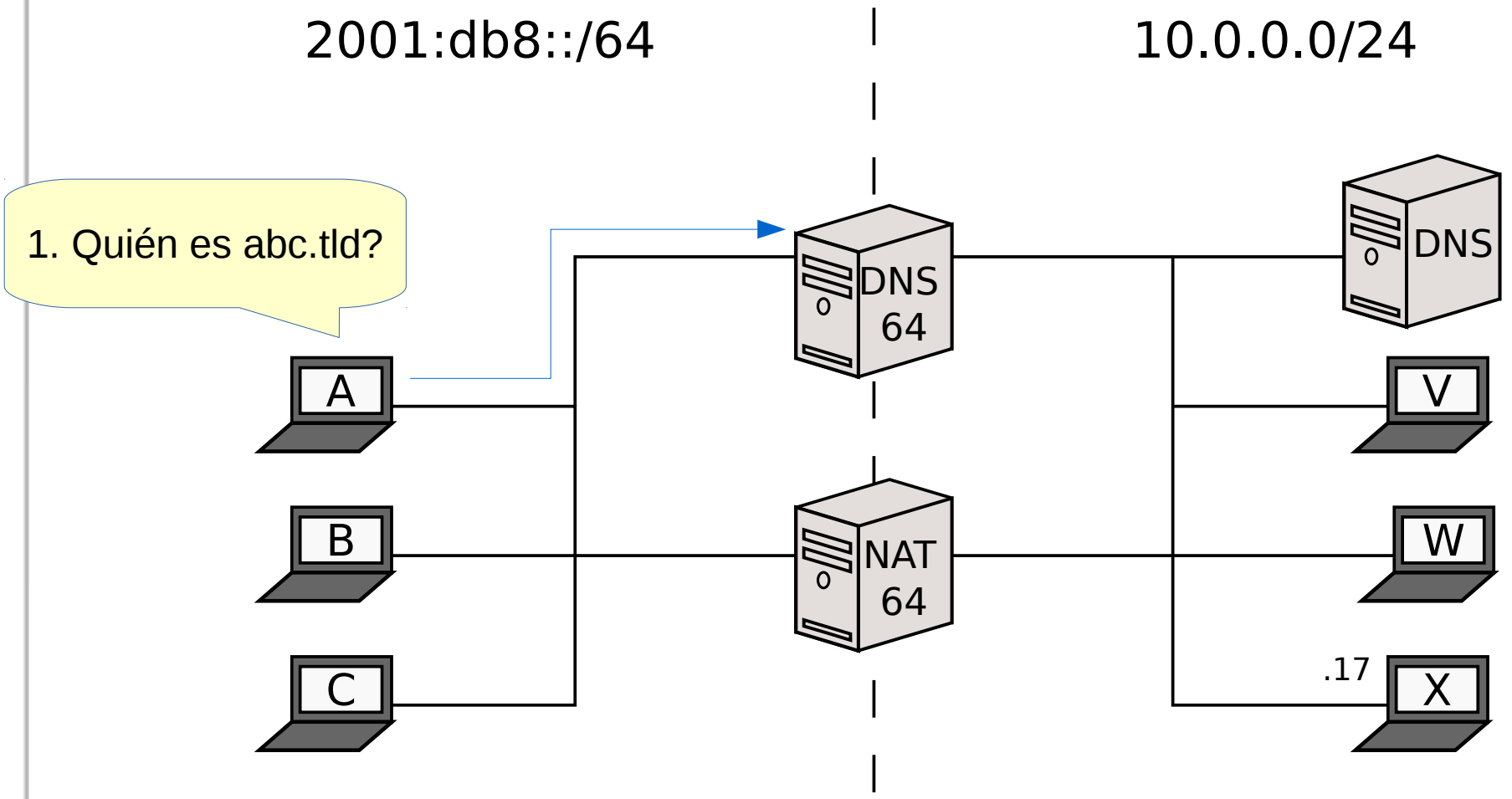
Siempre que se ejecuta Jool en una máquina virtual es muy importante apagar los offloads en la interfaz de red de la máquina host.

```
$ sudo apt-get install ethtool

$ sudo ethtool –-offload [interface] lro off
$ sudo ethtool –-offload [interface] gro off
```

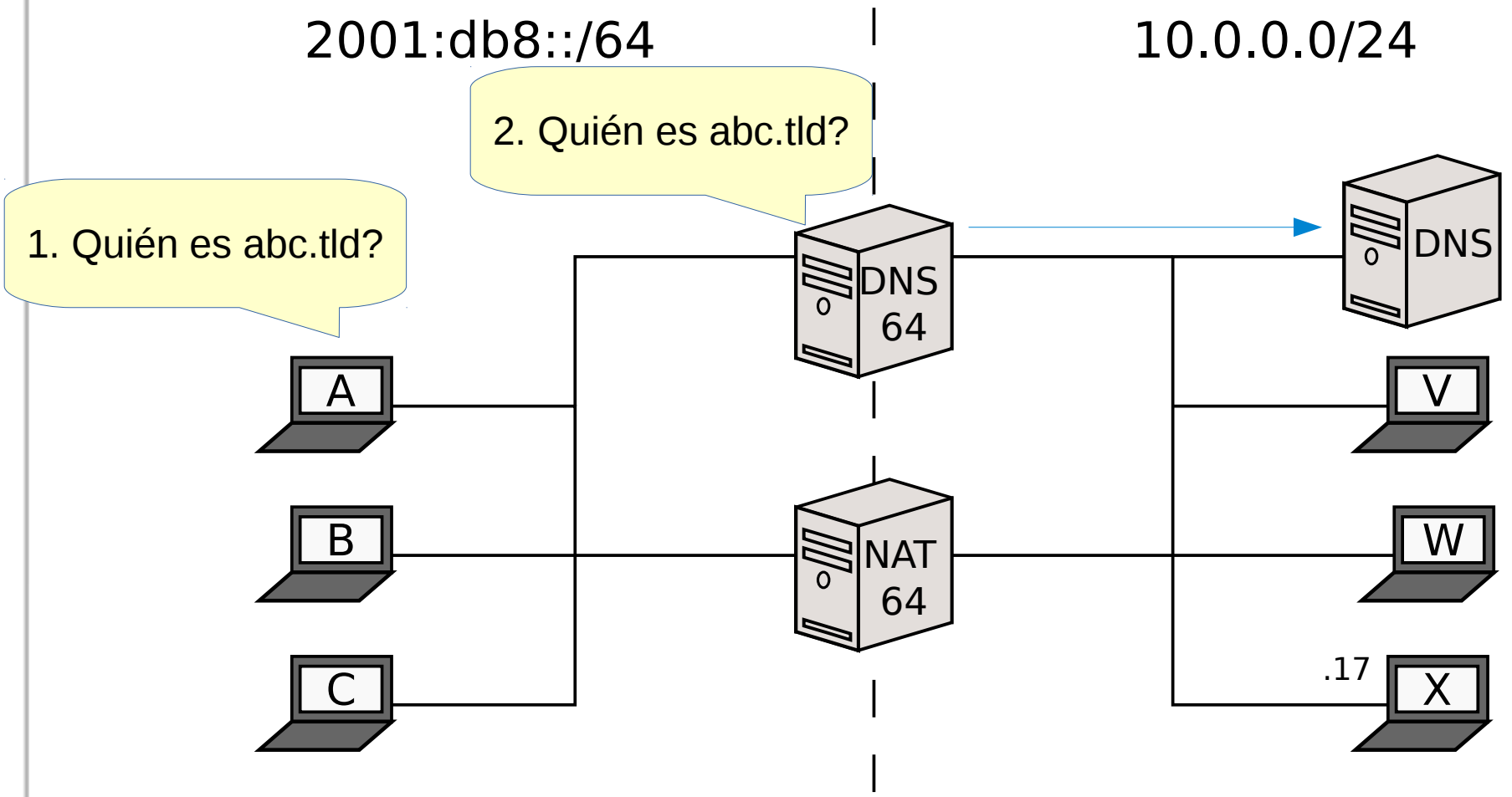
DNS64



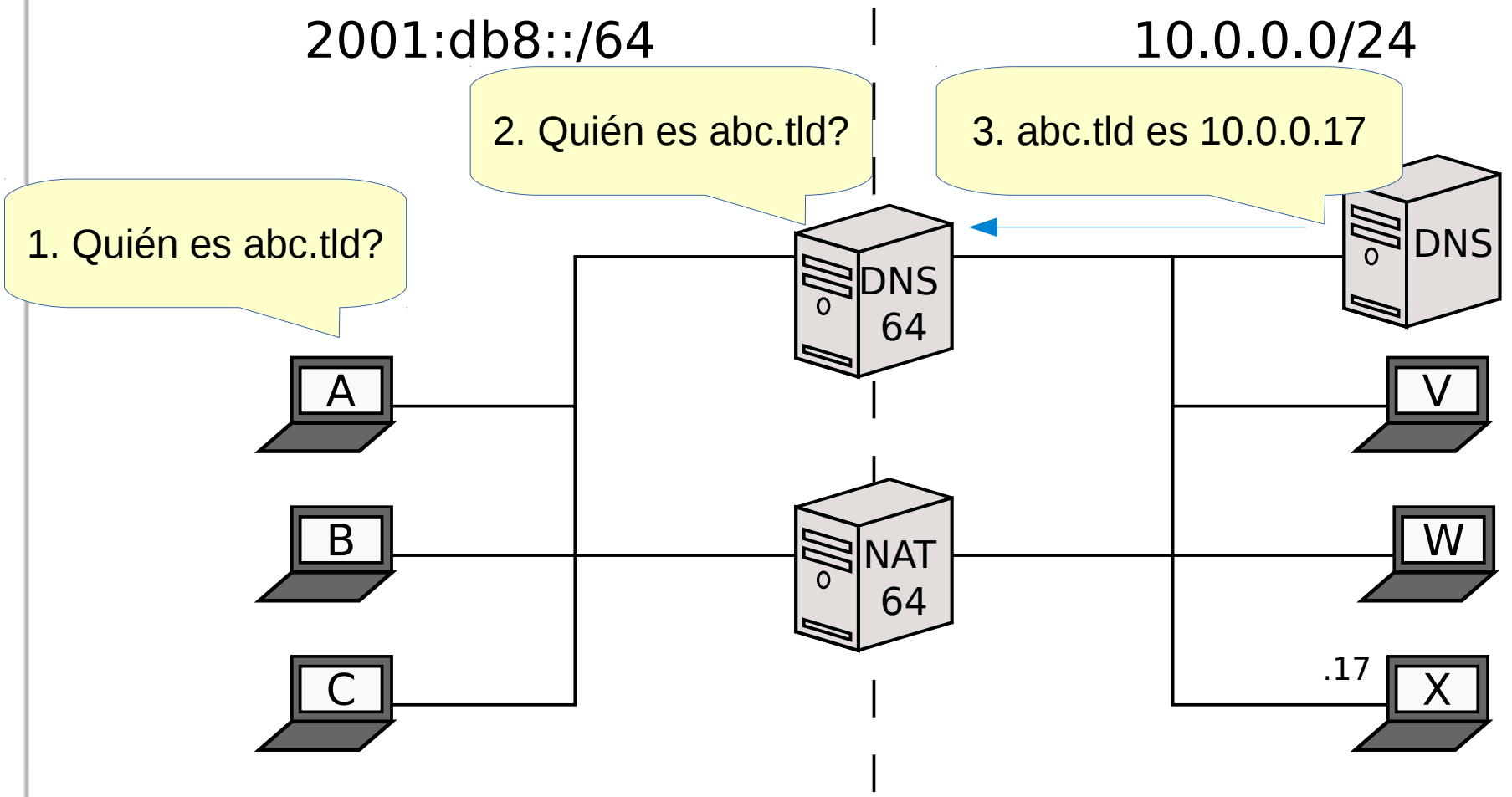
DNS64


2001:db8::/64
10.0.0.0/24
1. Quién es abc.tld?
A
B
C
DNS
64
NAT
64
DNS
V
W
.17
X

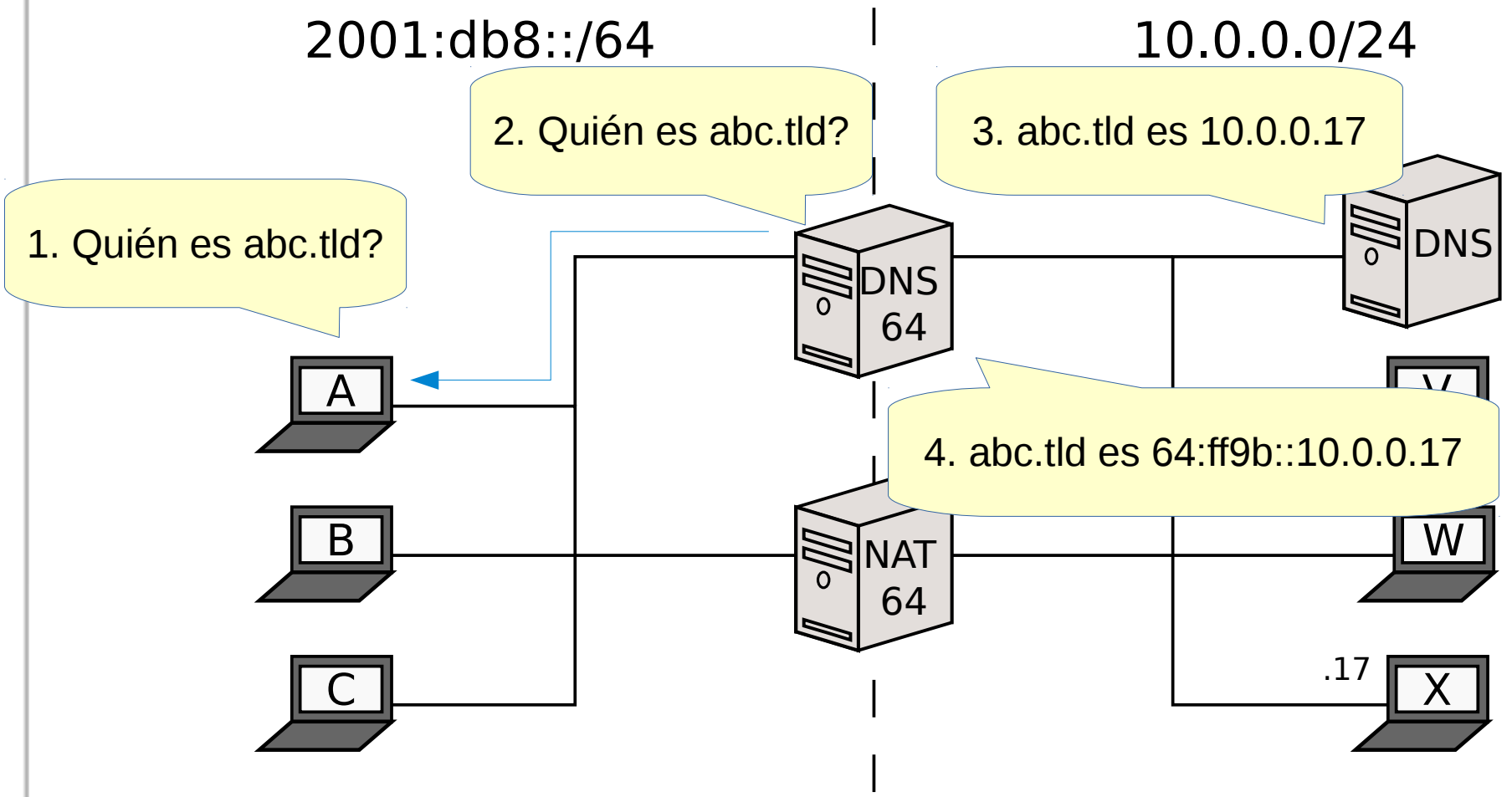
DNS64


2001:db8::/64
10.0.0.0/24
1. Quién es abc.tld?
2. Quién es abc.tld?
DNS
64
DNS
A
B
C
NAT
64
V
W
.17
X

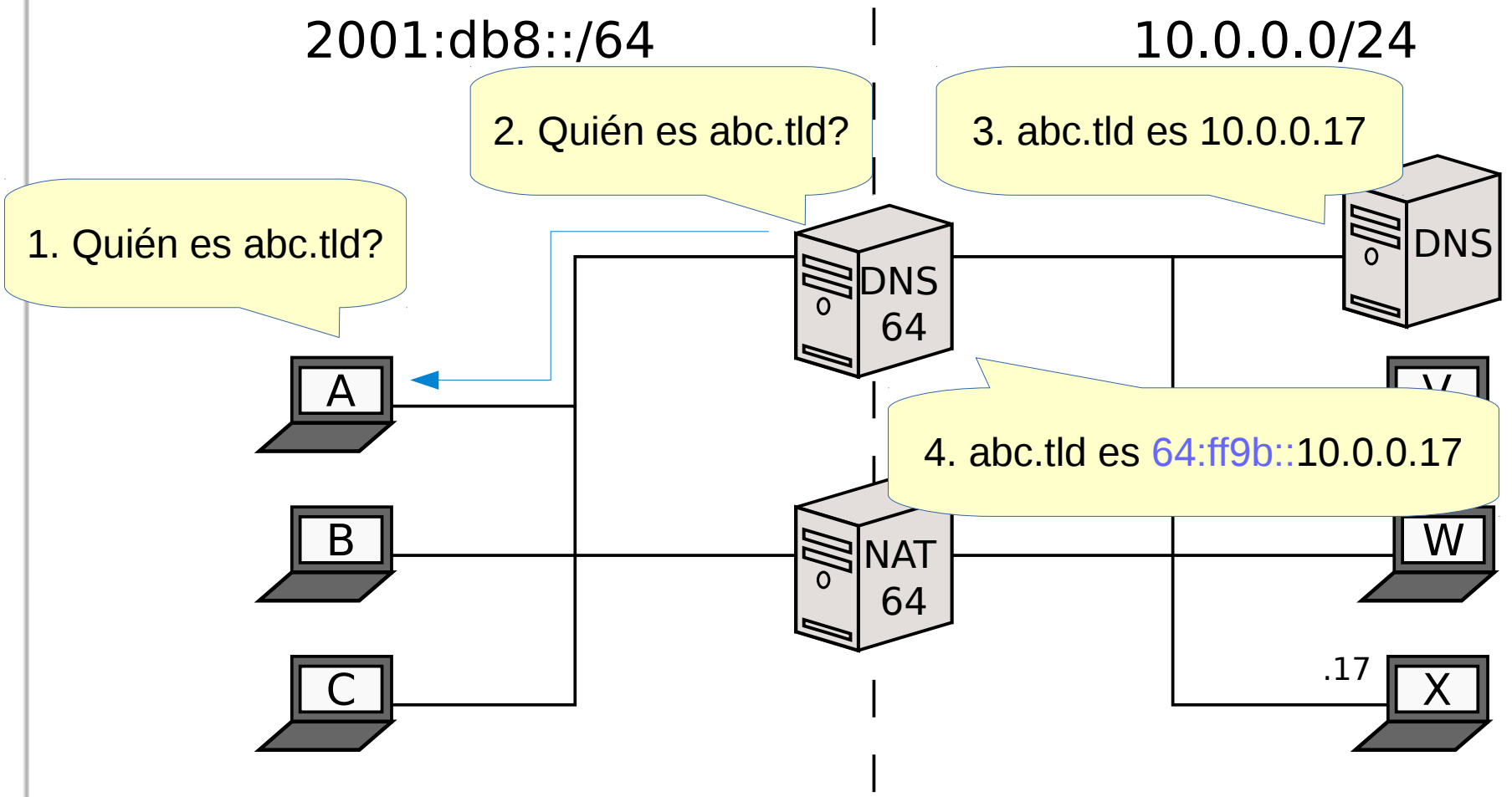
DNS64


2001:db8::/64
10.0.0.0/24
1. Quién es abc.tld?
2. Quién es abc.tld?
3. abc.tld es 10.0.0.17
DNS 64
DNS
A
B
C
NAT 64
V
W
.17
X

DNS64


2001:db8::/64
10.0.0.0/24
1. Quién es abc.tld?
2. Quién es abc.tld?
3. abc.tld es 10.0.0.17
4. abc.tld es 64:ff9b::10.0.0.17
DNS 64
DNS
NAT 64
A
B
C
W
.17
X

DNS64

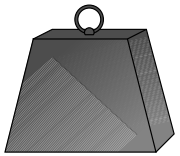

2001:db8::/64
10.0.0.0/24
1. Quién es abc.tld?
2. Quién es abc.tld?
3. abc.tld es 10.0.0.17
4. abc.tld es 64:ff9b::10.0.0.17
DNS 64
DNS
NAT 64
A
B
C
W
.17
X

DNS64 (BIND)

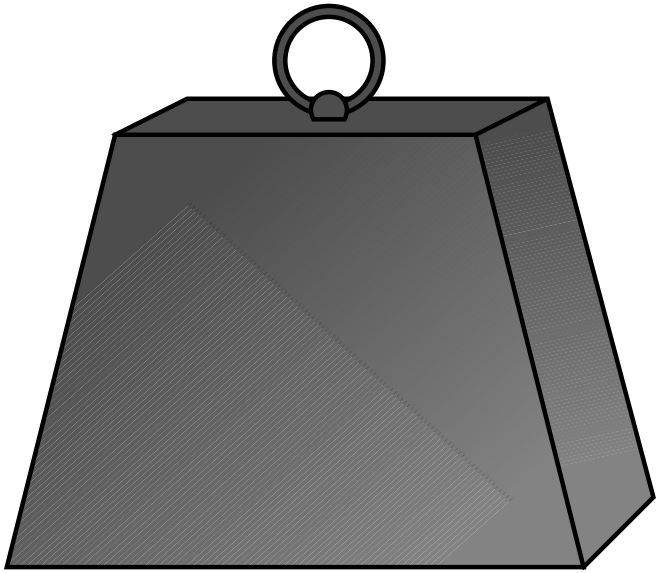
```
options {
        (...)

        # BIND no escucha en IPv6 por defecto.
        listen-on-v6 { any; };

        # Aquí se le indica a BIND el prefijo que
        # el NAT64 está agregando y quitando.
        dns64 64:ff9b::/96 {
                # Opciones
        };
};
```

SIIT vs NAT64

- SIIT
  - Stateless

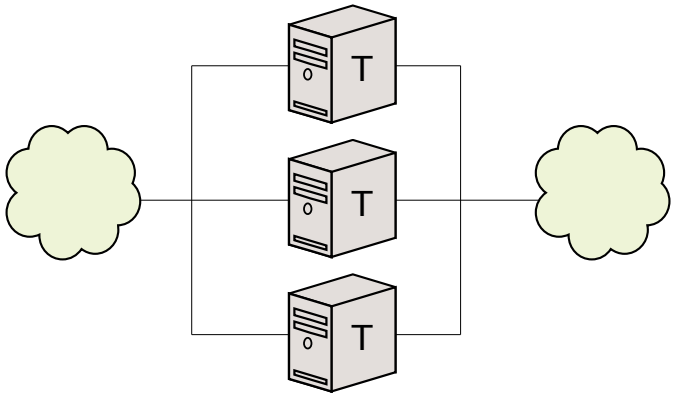


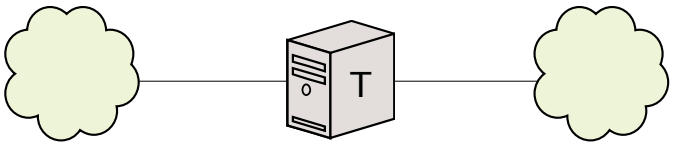
- NAT64
  - Stateful



SIIT vs NAT64

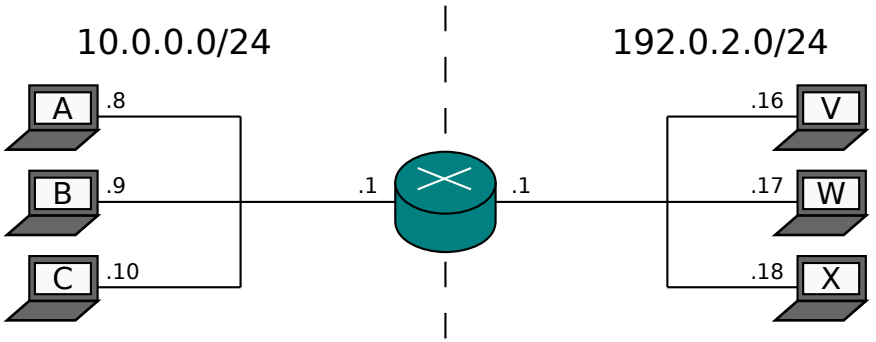
- SIIT
  - Stateless
- NAT64
  - Stateful

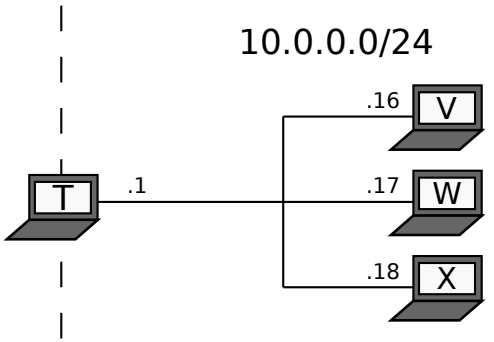




SIIT vs NAT64

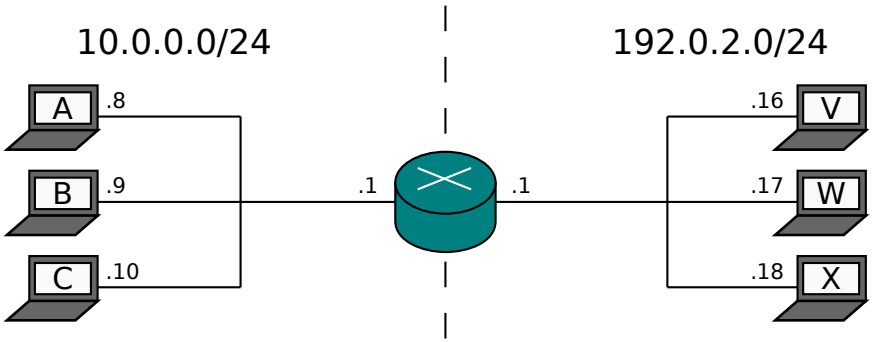
- SIIT
  - Transparente
- NAT64
  - No transparente

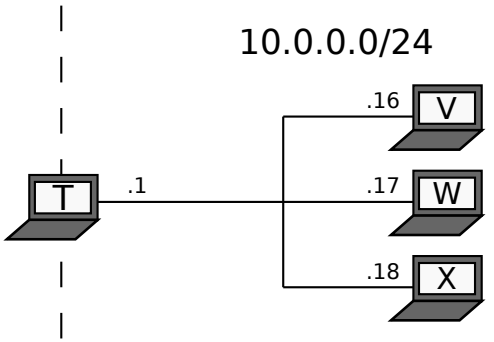




SIIT vs NAT64

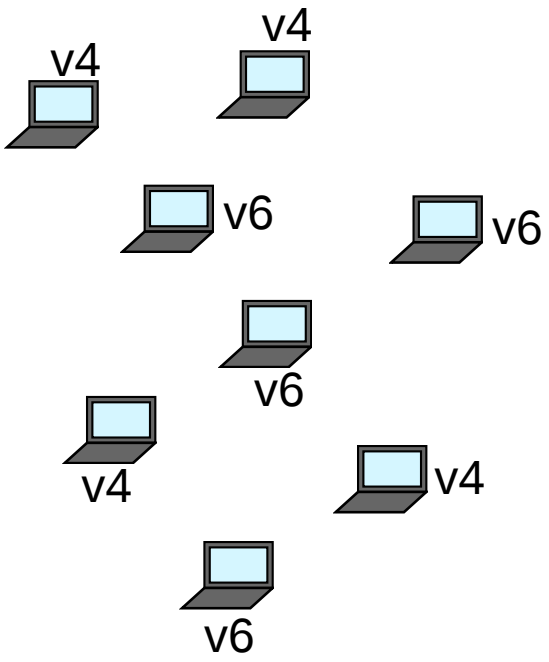
- SIIT
  - 1 a 1
- NAT64
  - 1 a N



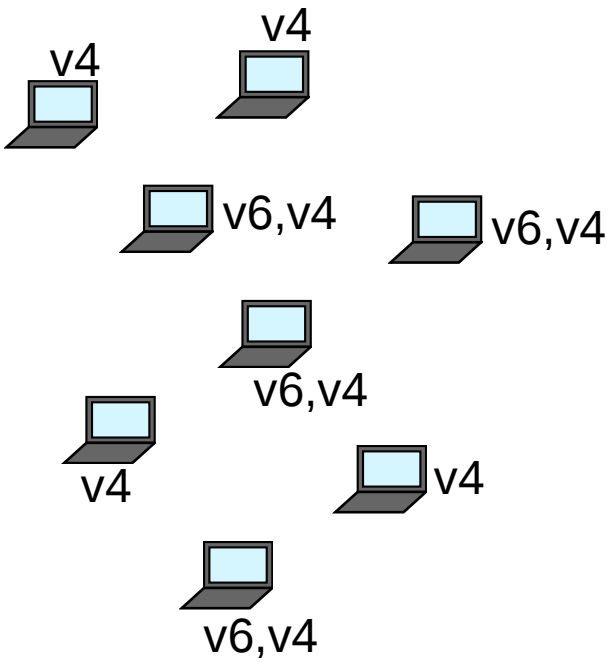


SIIT vs NAT64

- SIIT
  - Soluciona coexistencia

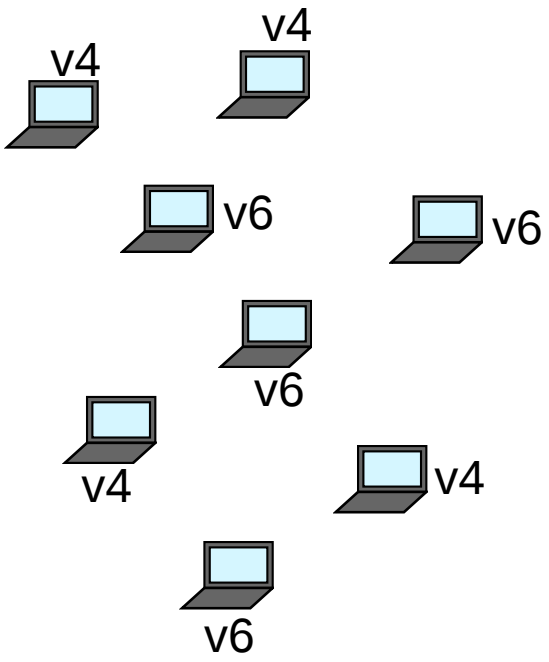


- NAT64
  - Soluciona coexistencia y agotamiento

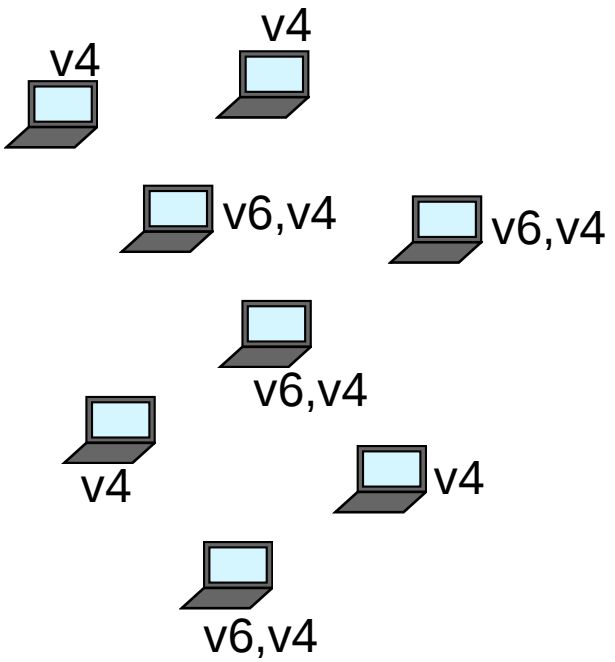


SIIT vs NAT64

- SIIT
  - Comunicación iniciable desde v4 y v6



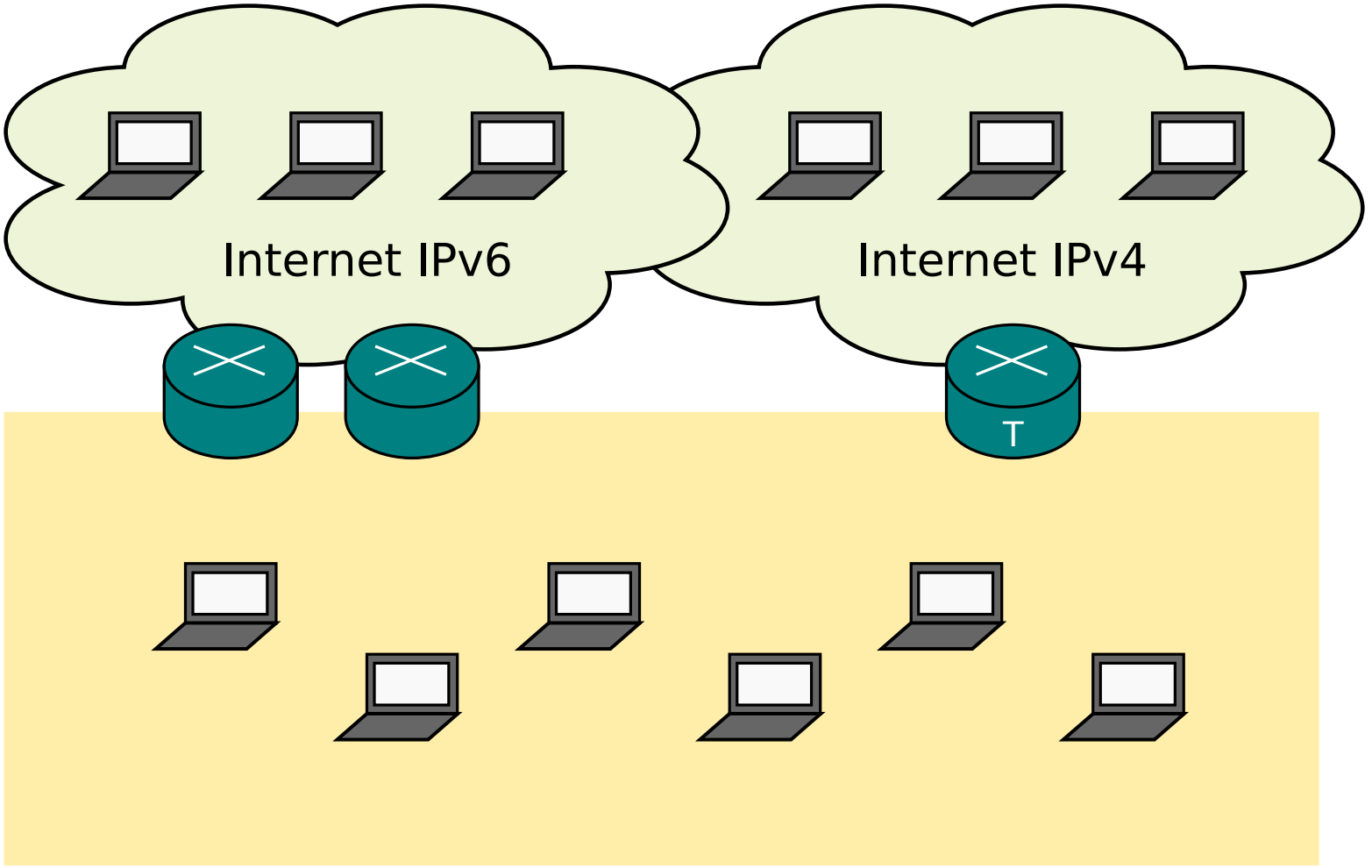
- NAT64
  - Requiere estado para iniciar desde v4



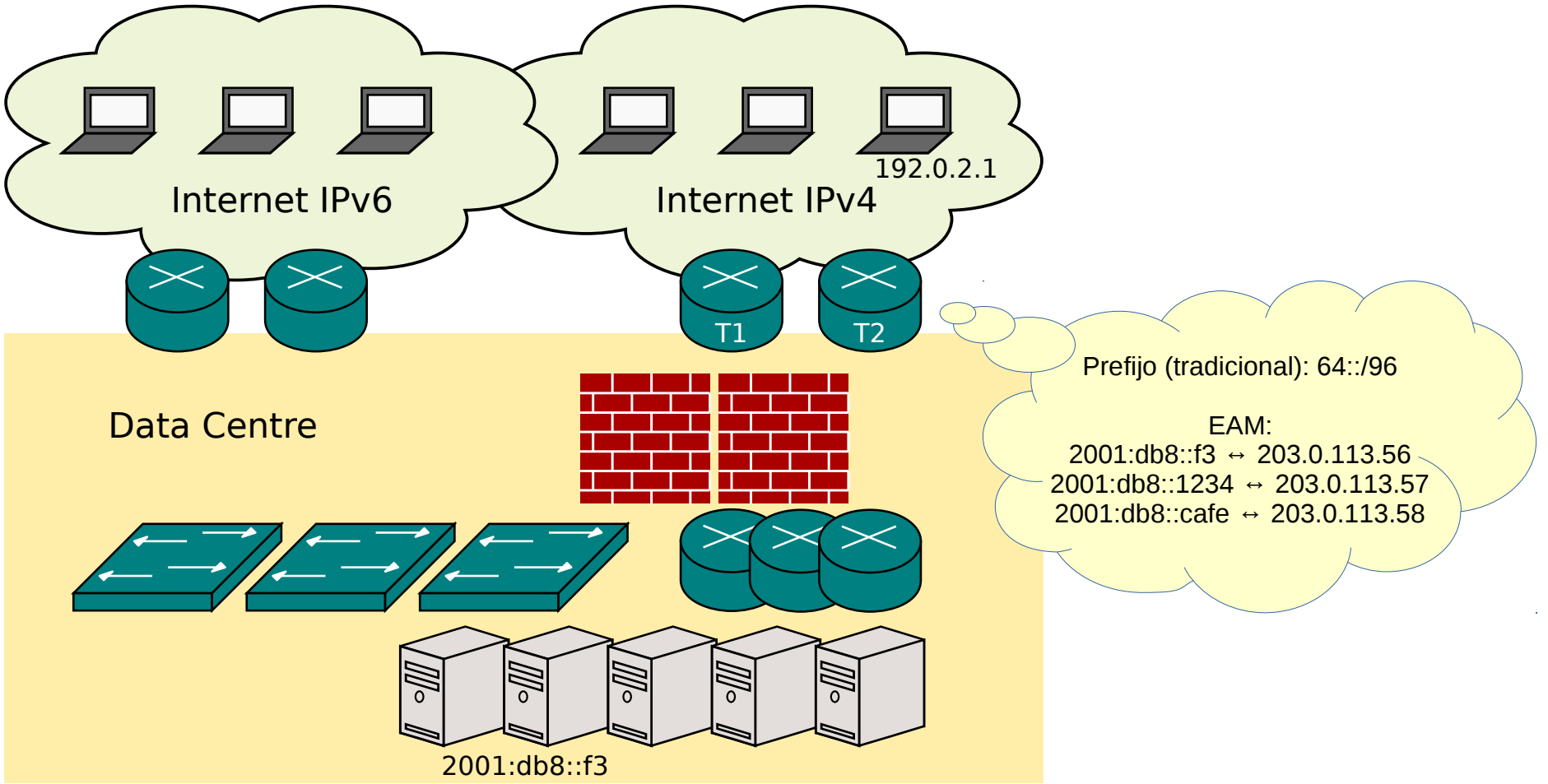
SIIT vs NAT64

- SIIT
  - TCP, UDP y otros
  - Capa 3 solamente
- NAT64
  - TCP y UDP
  - Capa 3 y 4

NAT64 (Vida real)



SIIT-DC


Internet IPv6
Internet IPv4
192.0.2.1
T1
T2
Data Centre
2001:db8::f3
Prefijo (tradicional): 64::/96
EAM:
2001:db8::f3 ↔ 203.0.113.56
2001:db8::1234 ↔ 203.0.113.57
2001:db8::cafe ↔ 203.0.113.58

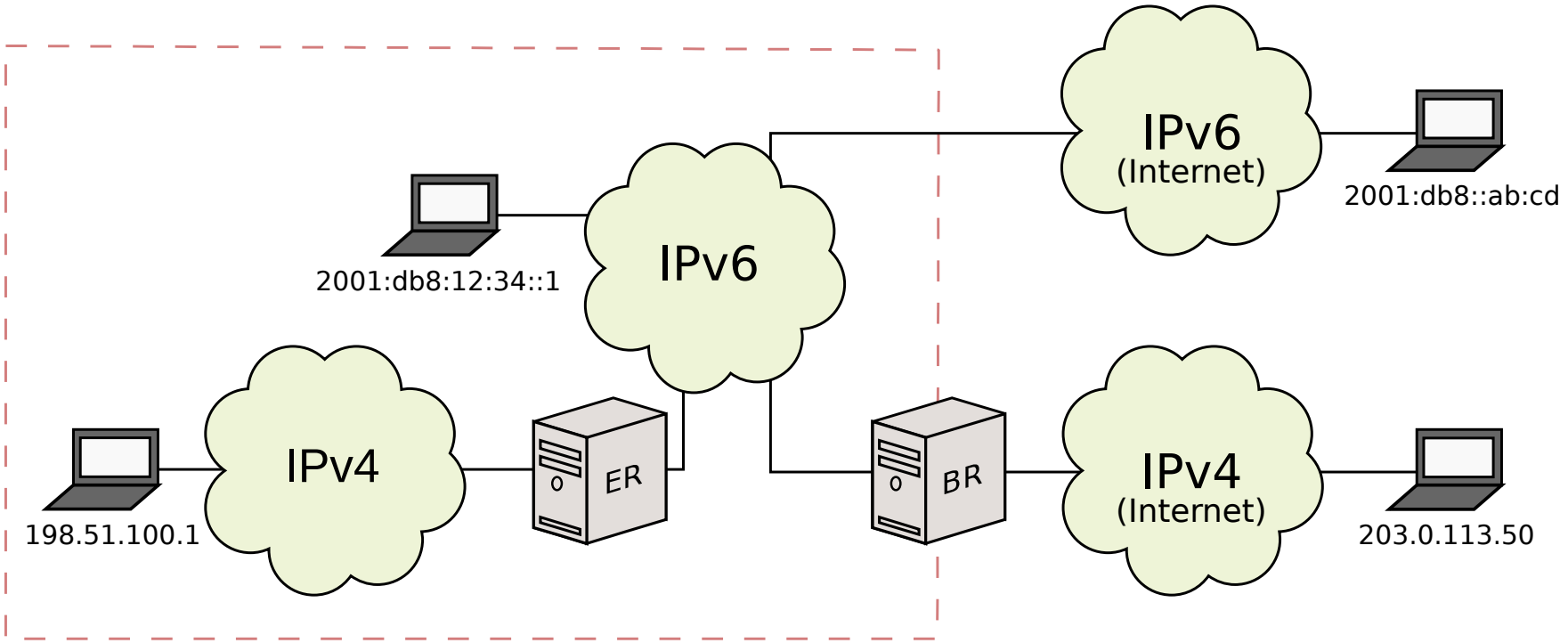
SIIT-DC

- Similar a NAT64,
  - Acceso a IPv4 se convierte en una especie de “servicio” proveído por la red.
  - No requiere que las direcciones de IPv6 tengan direcciones de IPv4 embebidas.
- A diferencia de NAT64,
  - No guarda estado.

Desventajas

- Literales
  - <a href=”http://192.0.2.1/index.html”>
- Algunas aplicaciones no son compatibles con Ipv6.

464XLAT y SIIT-DC Dual



Bitácora de traducción

Jool puede guardar en la bitácora todas las traducciones que realiza.

```
$ jool --logging-bib true

$ dmesg

[  312.493235] 2015/4/8 16:13:42 (GMT) - Mapped 2001:db8::5#19945 to
192.0.2.2#8208 (UDP)

[  373.724229] 2015/4/8 16:14:23 (GMT) - Mapped 2001:db8::8#46516 to
192.0.2.2#12592 (TCP)

[  468.675524] 2015/4/8 16:15:38 (GMT) - Forgot 2001:db8::5#19945 to
192.0.2.2#8208 (UDP)
```

Bitácora de sesiones

Jool también puede guardar en la bitácora todas las sesiones que son creadas y destruidas.

```
<date / time> - <action> session <IPv6 node>|<IPv6 representation of
IPv4 node>|<IPv4 representation of IPv6 node>|<IPv4 node>|<Protocol>
```

```
$ jool –-logging-session true
```

```
$ dmesg
```

```
[ 3238.087902] 2015/4/8 17:1:47 (GMT) - Added session 1::5#47073|64:ff9b::c000:205#80|
192.0.2.2#63527|192.0.2.5#80|TCP

[ 3238.099997] 2015/4/8 17:1:47 (GMT) - Added session 1::5#47074|64:ff9b::c000:205#80|
192.0.2.2#42527|192.0.2.5#80|TCP

[ 3478.498559] 2015/4/8 17:5:48 (GMT) - Forgot session 1::5#47073|64:ff9b::c000:205#80|
192.0.2.2#63527|192.0.2.5#80|TCP

[ 3478.499758] 2015/4/8 17:5:48 (GMT) - Forgot session 1::5#47074|64:ff9b::c000:205#80|
192.0.2.2#42527|192.0.2.5#80|TCP
```

Desempeño

- Medir el desempeño de Jool todavía es un “Trabajo en Progreso”
- Pero tenemos algunos puntos de referencia que podemos reportar:
  - Respuesta de la comunidad
  - Prueba de T-Rex

Respuesta de la comunidad

- En general hemos recibido buenos comentarios de la comunidad sobre el desempeño de Jool.
- Se han abierto algunos bugs al respecto, pero la solución siempre ha resultado ser un ajuste ajeno a Jool.

Respuesta de la comunidad

[Nuestros SIIT-BRs] están prácticamente ociosos. Están traduciendo aproximadamente 100Mb/s de casi exclusivamente tráfico web. De hecho, el hardware es bastante viejo; Sun X4170s con 2x quad-core Intel L5520 CPUs. Menos de la cuarta parte de un solo CPU se usa para todo el sistema (así que no solamente Jool), los 7.75 restantes están libres.

Otra cosa que vale la pena mencionar es que dado que SIIT es stateless, puedes escalar horizontalmente fácilmente. Solo agrega más máquinas traductoras con configuración idéntica, y corre balanceo de carga entre ellas.

En general alcanzas los límites de memoria de bus antes que los del CPU.

https://mail-lists.nic.mx/pipermail/jool-list/

Prueba de T-Rex

- Prueba de carga natural implementada por Cisco. La variante “Advance Stateful Support” soporta NAT64.

  https://trex-tgn.cisco.com

- CentOS 7 Dell R630 con dual Xeon E5-2680 v4 (28 cores en total), 64 GB de RAM y NIC Intel X710.

Prueba de T-Rex

```
-Per port stats table
      ports |               0 |               1
 -----------------------------------------------------------------------------------------
   opackets |        14987324 |        41611624
     obytes |      1776772320 |     56340402998
   ipackets |        41610306 |        14987530
     ibytes |     57170300382 |      1477031674
    ierrors |               0 |               0
    oerrors |               0 |               0
      Tx Bw |      23.74 Mbps |     752.99 Mbps

-Global stats enabled
 Cpu Utilization : 4.4  %  35.0 Gb/core
 Platform_factor : 1.0
 Total-Tx        :     776.73 Mbps
 Total-Rx        :     783.78 Mbps
 Total-PPS       :      94.55 Kpps
 Total-CPS       :       2.78 Kcps

 Expected-PPS    :       0.00  pps
 Expected-CPS    :       0.00  cps
 Expected-BPS    :       0.00  bps

 Active-flows    :     6665  Clients :      256   Socket-util : 0.0413 %
 Open-flows      :  1665014  Servers :    65536   Socket :     6665 Socket/Clients :  26.0
 drop-rate       :       0.00  bps
 current time    : 603.5 sec
 test duration   : 0.0 sec
```

- ~780Mbps
- 0 drop rate
- ~1% utilización de CPU (el 4.4% de la izquierda se refiere al cliente, no al traductor.)

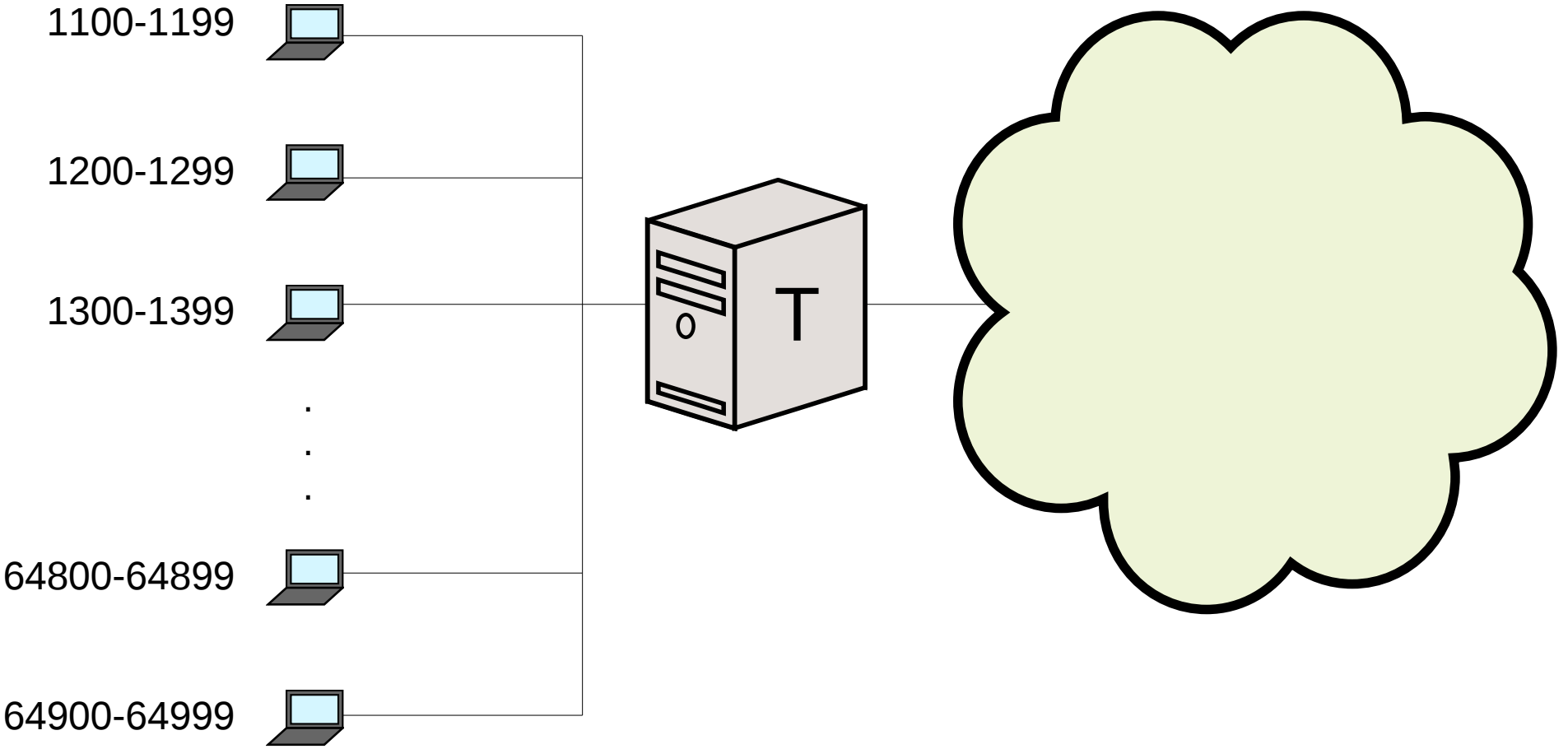
- Jool es 100% CPU, de modo que no parece estar imponiendo significativo overhead

Trabajo en Proceso

Mapeo de direcciones determinístico

- Definido en el RFC 7422.
- Busca reducir o eliminar la necesidad de tener una bitácora de traducción.
- Desventajas:
  - Los recursos pueden no ser utilizados al 100%.
  - Es un esquema un poco más complejo que el mapeo de direcciones dinámico.

Mapeo de direcciones determinístico



Contacto

https://jool.mx	Página oficial
https://github.com/NICMx/Jool	Repositorio del código
jool-list@nic.mx	Discusión pública y noticias
jool-news@nic.mx	Noticias
jool@nic.mx	Discusión privada

Muchas gracias