

# netkit lab

## Dos sistemas

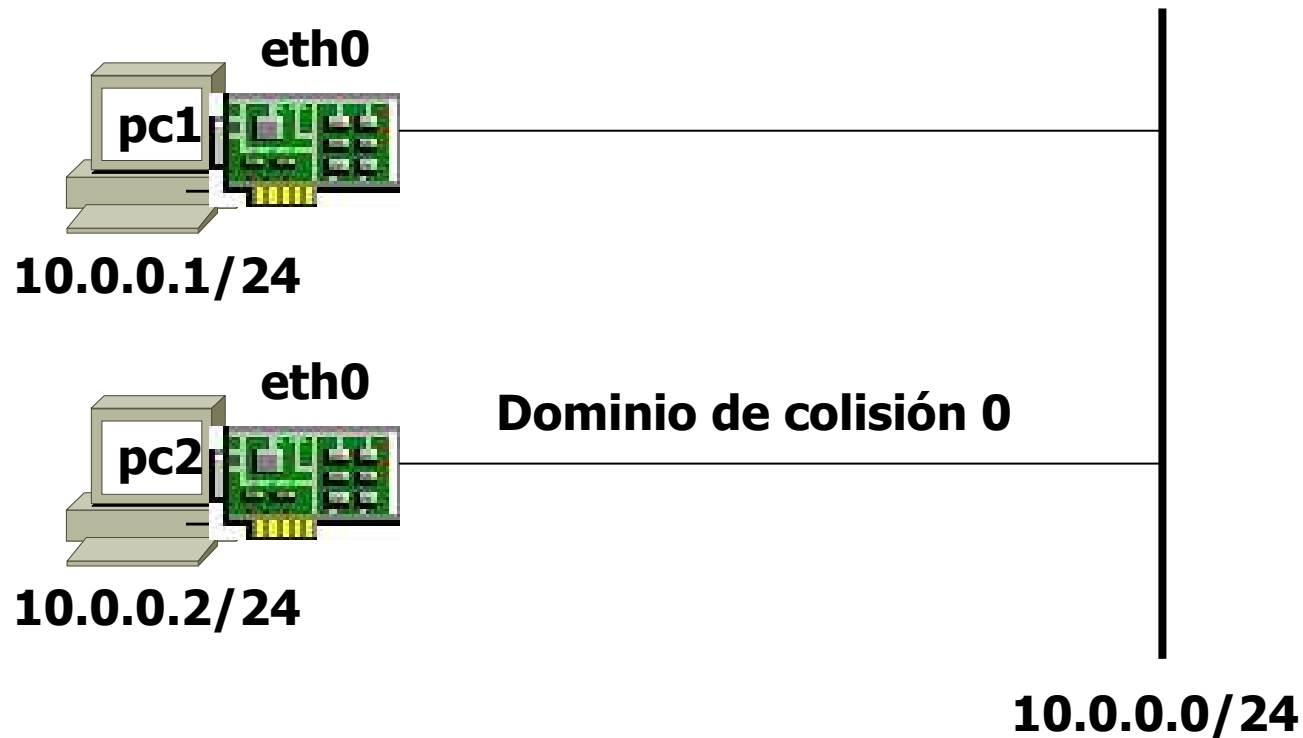
<b>Version</b>	2.2
<b>Author(s)</b>	G. Di Battista, M. Patrignani, M. Pizzonia, M. Rimondini, traducción J.M. San José
<b>E-mail</b>	<a href="mailto:contact@netkit.org">contact@netkit.org</a>
<b>Web</b>	<a href="http://www.netkit.org/">http://www.netkit.org/</a>
<b>Description</b>	Como poner en marcha y gestionar dos máquinas virtuales

# copyright notice

- All the pages/slides in this presentation, including but not limited to, images, photos, animations, videos, sounds, music, and text (hereby referred to as “material”) are protected by copyright.
- This material, with the exception of some multimedia elements licensed by other organizations, is property of the authors and/or organizations appearing in the first slide.
- This material, or its parts, can be reproduced and used for didactical purposes within universities and schools, provided that this happens for non-profit purposes.
- Information contained in this material cannot be used within network design projects or other products of any kind.
- Any other use is prohibited, unless explicitly authorized by the authors on the basis of an explicit agreement.
- The authors assume no responsibility about this material and provide this material “as is”, with no implicit or explicit warranty about the correctness and completeness of its contents, which may be subject to changes.
- This copyright notice must always be redistributed together with the material, or its portions.

# Único sistema

- Una red simple con dos sistemas conectados en el mismo dominio de colisión



# Paso 1 – creando las vms

## ▼ Máquina anfitrión

```
user@localhost:~$ vstart pc1 --eth0=A
```

```
===== Starting virtual machine "pc1" =====
```

```
Kernel:    /home/user/netkit2/kernel/netkit-kernel
```

```
Modules:   /home/user/netkit2/kernel/modules
```

```
Memory:    8 MB
```

**pc1 se crea y una ventana de consola se abre para pc1**

```
user@localhost:~$ vstart pc2 --eth0=A
```

```
===== Starting virtual machine "pc2" =====
```

```
Kernel:    /home/user/netkit2/kernel/netkit-kernel
```

```
Modules:   /home/user/netkit2/kernel/modules
```

```
Memory:    8 MB
```

**pc2 se crea y una ventana de consola se abre para pc2**

# Paso 2 – configurando interfaces de red

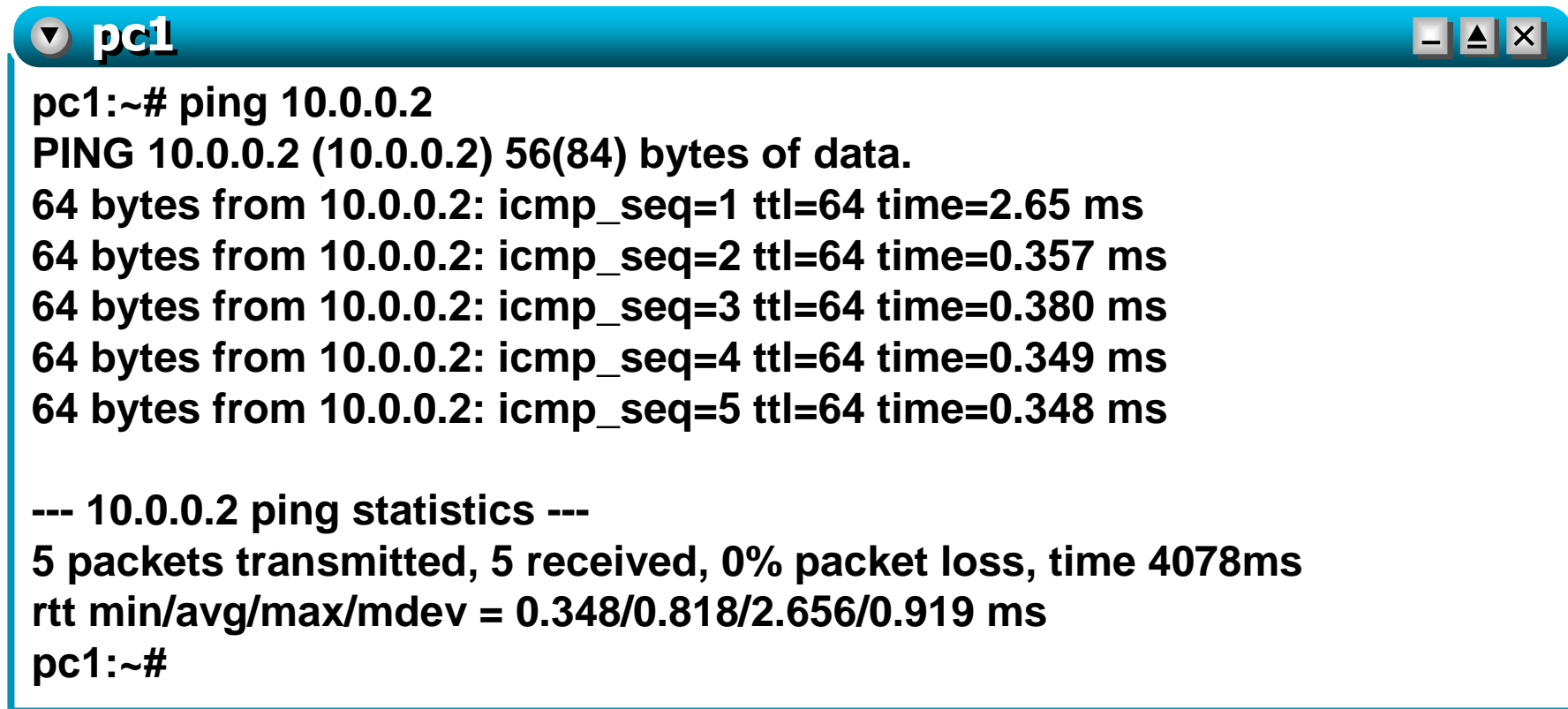
▼ pc1

```
pc1:~# ifconfig eth0 10.0.0.1 netmask 255.255.255.0 broadcast  
10.0.0.255 up  
pc1:~#
```

▼ pc2

```
pc2:~# ifconfig eth0 10.0.0.2 netmask 255.255.255.0 broadcast  
10.0.0.255 up  
pc2:~#
```

## Paso 3 – ping



```
pc1:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=2.65 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.357 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.380 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.349 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.348 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4078ms
rtt min/avg/max/mdev = 0.348/0.818/2.656/0.919 ms
pc1:~#
```

- pc1 y pc2 pueden alcanzarse mutuamente

# Paso 4 – una mirada a los paquetes

- Veamos los paquetes intercambiados en el dominio de colisión A
- Se usa tcpdump, un analizador que muy usado en linux

TCPDUMP(8) TCPDUMP(8)

NAME

tcpdump - dump traffic on a network

SYNOPSIS

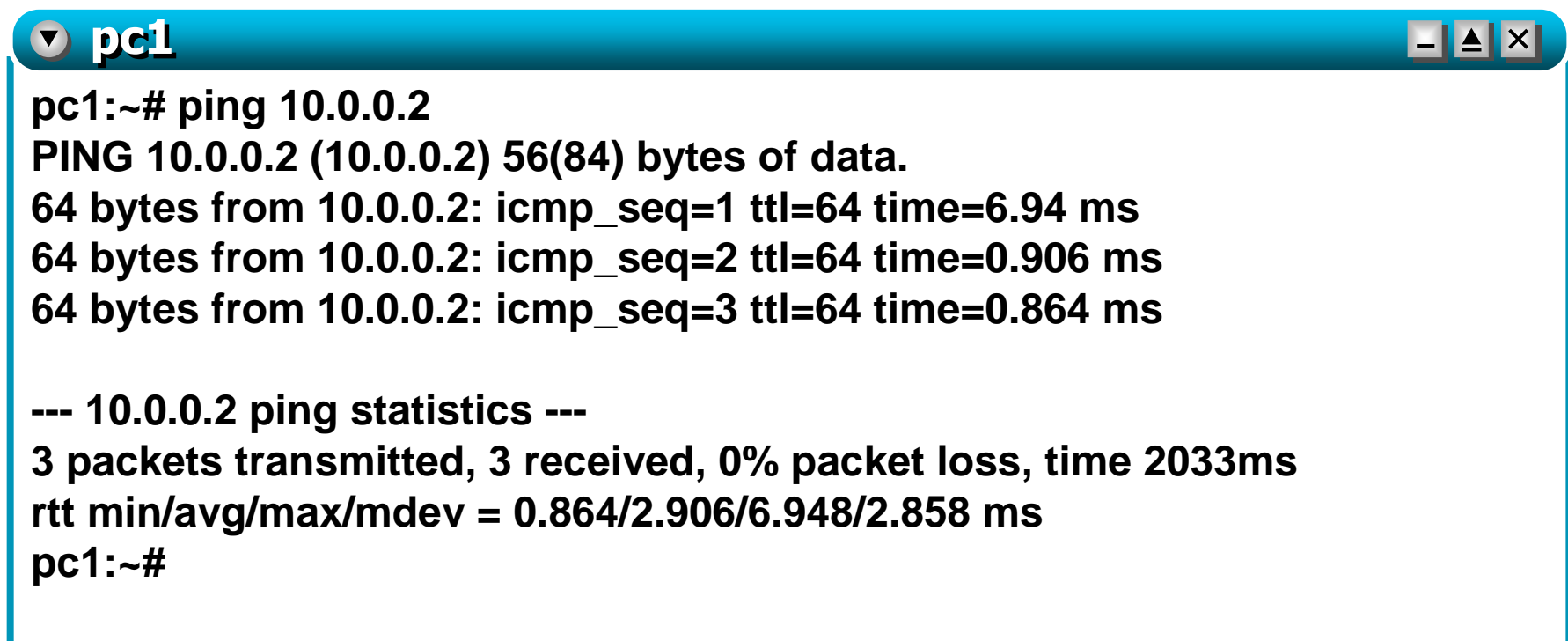
```
tcpdump [ -AdDeflLnNOpPqRtUvX ] [ -c count ]  
[ -C file_size ] [ -F file ]  
[ -i interface ] [ -m module ] [ -n ]  
[ -s snaplen ] [ -T type ] [ -w file ]  
[ -E spi@ipaddr algo:secret,... ]
```

Número de bytes capturados por paquete (el defecto es 88)

Guarda los paquetes en un fichero

## Paso 4 – ping

### ■ Ping desde pc1



```
pc1:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=6.94 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.906 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.864 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2033ms
rtt min/avg/max/mdev = 0.864/2.906/6.948/2.858 ms
pc1:~#
```



## Paso 4 – una mirada en los paquetes

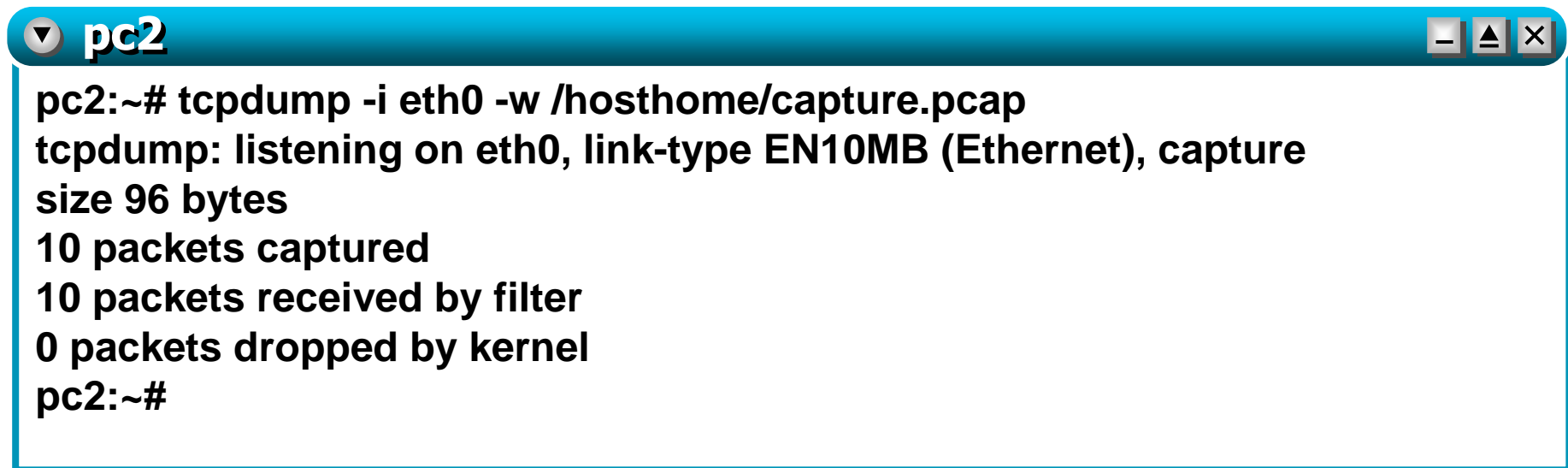
- Al mismo tiempo capturamos en pc2 (ctrl-C para terminar)

▼ **pc2**

```
pc2:~# tcpdump -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
19:27:17.899782 arp who-has 10.0.0.2 tell 10.0.0.1
19:27:18.002578 arp reply 10.0.0.2 is-at fe:fd:0a:00:00:02
19:27:18.004384 IP 10.0.0.1 > 10.0.0.2: icmp 64: echo request seq 1
19:27:18.005806 IP 10.0.0.2 > 10.0.0.1: icmp 64: echo reply seq 1
19:27:18.920463 IP 10.0.0.1 > 10.0.0.2: icmp 64: echo request seq 2
19:27:18.920605 IP 10.0.0.2 > 10.0.0.1: icmp 64: echo reply seq 2
6 packets captured
6 packets received by filter
0 packets dropped by kernel
pc2:~#
```

## Paso 4 – mirando los paquetes con el interfaz gráfico

- Del mismo modo que en el caso anterior pero guardando los paquetes capturados en el fichero **capture.pcap** (en la máquina)
  - El directorio (real) del usuario está disponible dentro de la vm bajo el directorio **/hosthome**



```
pc2:~# tcpdump -i eth0 -w /hosthome/capture.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture
size 96 bytes
10 packets captured
10 packets received by filter
0 packets dropped by kernel
pc2:~#
```

# Paso 4 – mirando los paquetes con el interfaz gráfico

- Abrir **capture.pcap** en el sistema real usando un analizador de paquetes (como **ethereal**)

