Università degli Studi Roma Tre
Dipartimento di Ingegneria
Computer Networks Research Group

katharà lab

two-hosts

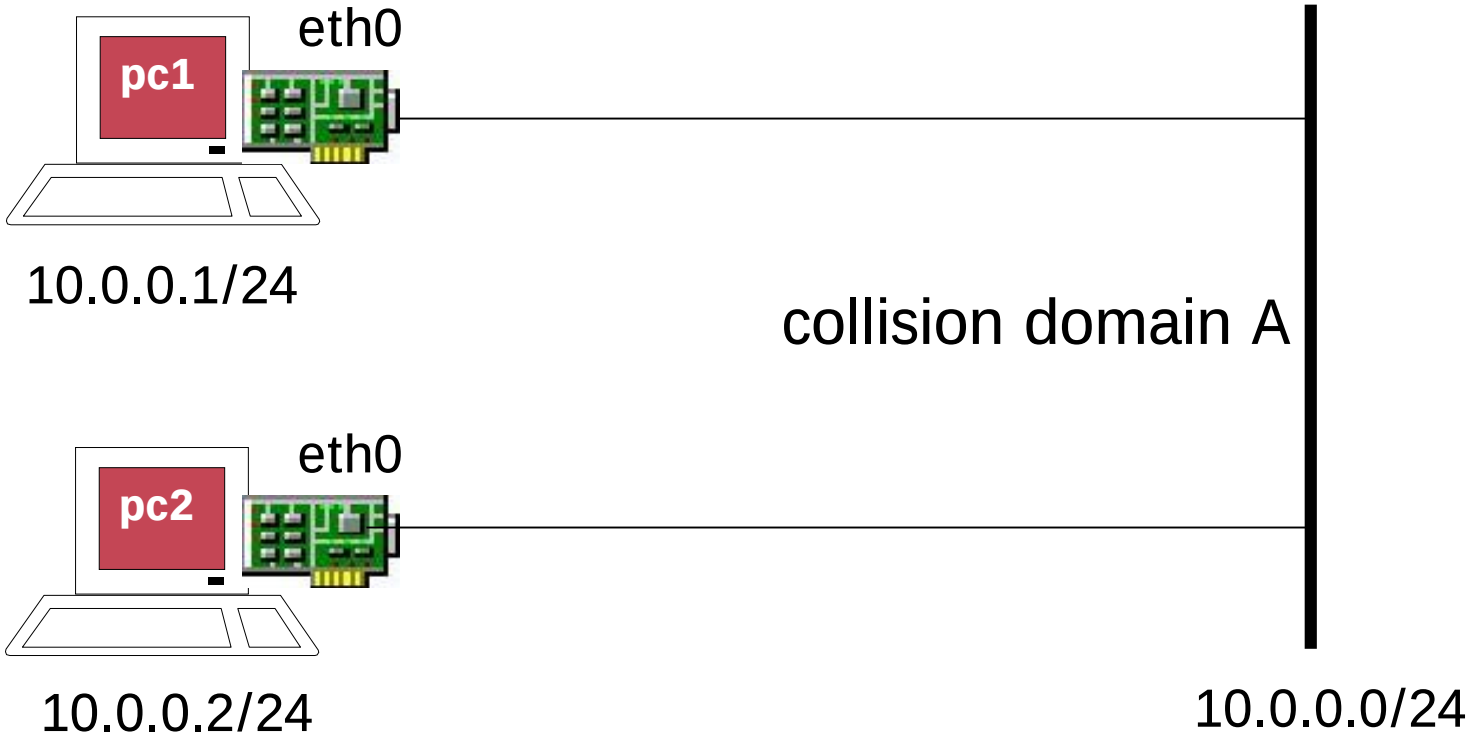
Version	1.0
Author(s)	G. Di Battista, M. Patrignani, M. Pizzonia, M. Rimondini
E-mail	contact@kathara.org
Web	http://www.kathara.org/
Description	setting up a network between two virtual machines; kathara version of netkit lab two-hosts version 2.2

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.

two hosts

- a simple network with two hosts connected to the same collision domain



step 1 – creating the vms

host machine

```
user@localhost:~$ vstart --eth 0:A –n pc1

======================== Starting Lab ========================
bb423a23157cde000990e9a94dab36928d0ba0aecff38570d653e8b60b429550
6ff9da2ece5f418d7fbf89f0b382e33bc9a4aadbe90f9ebaa1c2b203f005a76a
```

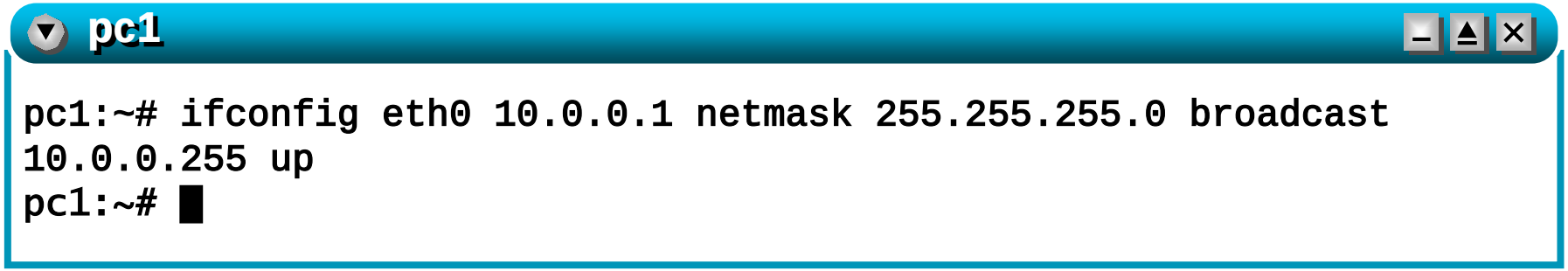
pc1 is created and a console window opens for **pc1**

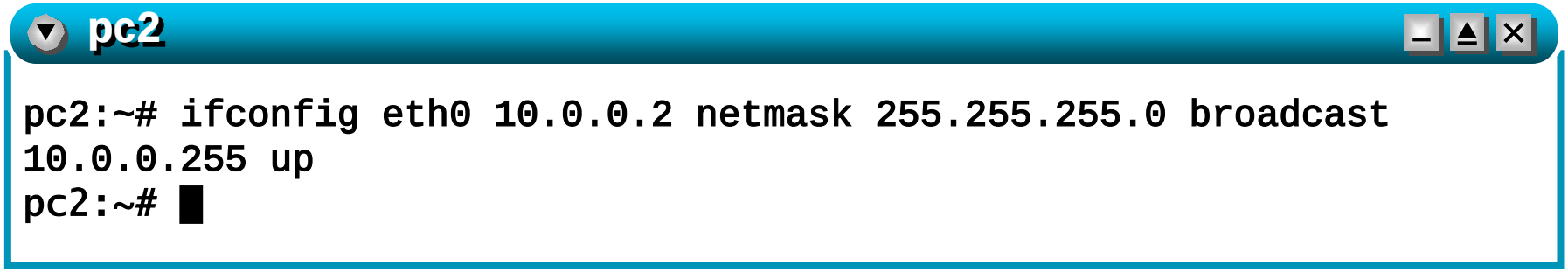
```
user@localhost:~$ vstart --eth 0:A –n pc2

======================== Starting Lab ========================
Error response from daemon: network with name netkit_nt_A already exists
c2ad58fae2a38b7ad7f003695c20bdac192b14f7b3bdd2b0f3294741d7b21f1
```

pc2 is created and a console window opens for **pc2**

step 2 – configuring network interfaces





step 3 - ping

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=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** and **pc2** can reach each other

step 4 – a look at the packets

- let's look at the packets exchanged on collision domain A
- we use tcpdump, a network sniffer that is widely available on linux boxes

```
TCPDUMP(8)

NAME
       tcpdump - dump traffic on a

SYNOPSIS
       tcpdump [ -AdDeflLnNO      tuUvxX ] [ -c cou
               [ -C file_s     ] [ -F file ]
               [ -i int  face ] [ -m module ] [    r file ]
               [ -s snaplen ] [ -T type ] [ -w file ]
               [ -E spi@ipaddr algo:secret,...  ]
```

number of bytes captured per packet (default is 68)

stores the packets to file

step 4 – a look at the packets

- ping from **pc1**

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:~#
```

step 4 – a look at the packets

- at the same time, sniff from **pc2** (ctrl+C to interrupt)

pc2

```
pc2:~# tcpdump -tenni 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:~#
```

step 4 – looking at the packets with a graphical interface

- same as before, but store sniffed packets into file **`capture.pcap`** (on the host machine)
  - the (real) home directory of the current user is made available inside the vm under **`/hosthome`**

pc2

```
pc2:~# tcpdump -tenni 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:~#
```

step 4 – looking at the packets with a graphical interface

- open **`capture.pcap`** on the real host machine using a packet dissector (like, e.g., **`ethereal`**)