

Corso di Configurazione e Gestione di Reti Locali

Marco Bonola
Lorenzo Bracciale

A.A. 2011/2012

TOC

- Netkit: installation, configuration, use
- Lab0-interfaces: basic IP configuration
- IP Networking
 - (ifconfig, route, arp), iproute2 (neigh, link, addr, etc..), boot configuration, /proc entries
- Dynamic configuration with DHCP
 - DHCP server with netkit (Lab0-dhcp)
- Linux network tools
 - ping, traceroute, netcat, telnet, ssh, ss, tcpdump/wireshark, rsync, wget
- Routing protocol essentials
- Lab2-quagga: dynamic routing on Linux with QUAGGA
 - RIP, OSPF

NETWORK EMULATION WITH NETKIT

NETKIT

- **Homepage**
 - http://wiki.netkit.org/index.php/Main_Page
- **Download**
 - http://wiki.netkit.org/index.php/Download_Official
- **Slides and Labs**
 - http://wiki.netkit.org/index.php/Labs_Official
- **For the NETKIT introduction we'll use the slides at the following URL:**
 - http://wiki.netkit.org/netkit-labs/netkit_introduction/netkit-introduction.pdf

IP STATIC CONFIGURATION

IP configuration

- IP Networking control files
- NIC layer 2 configuration
 - `ip link`
- ARP configuration
 - `ip neigh`
- IP address configuration
 - `ip addr`
- IP routing/forwarding configuration
 - `ip route`

IP Networking Control Files

- Different Linux distributions put their networking configuration files in different places in the filesystem
- EX:
 - Debian: `/etc/network/interfaces`
 - Gentoo: `/etc/conf.d/net`
 - Slackware: `/etc/rc.d/rc.inet1`
- We'll refer to Debian based distros as the NETKIT virtual machines are Debian

Debian interfaces

- Complete doc: `man interfaces`

Essentials:

- `/etc/init.d/networking` (start | restart | stop)

- static:

```
auto eth0                # bring the interface up automatically
iface eth0 inet static   # static configuration
address 192.168.1.5      # set ip address
netmask 255.255.255.0    # set netmask
gateway 192.168.1.254    # set default GW route
```

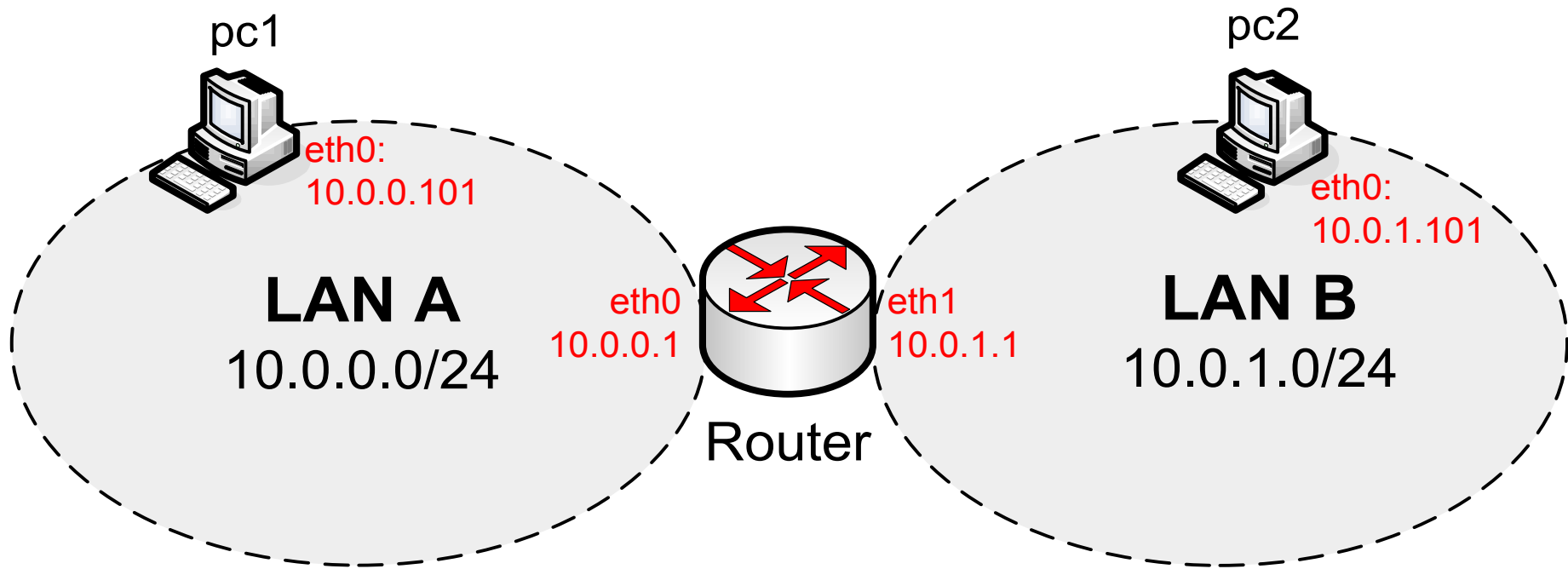
- dhcp:

```
iface eth0 inet dhcp     # use DHCP for IP configuration
```

- up and down scripts

```
up route add default gw 192.168.1.200
down route del default gw 192.168.1.200
(up|down) /etc/init.d/whatever-script.sh
```


Lab0-interfaces



Lab0-interfaces set-up

lab.conf:

```
router[0]=A
router[1]=B
router[mem]=64
pc1[0]=A
pc2[0]=B
```

To start the lab (on the host):

```
$ lstart
```

pc1/etc/network/interfaces

```
auto lo
iface lo inet loopback
```

```
auto eth0
iface eth0 inet static
    address 10.0.0.101
    netmask 255.255.255.0
    gateway 10.0.0.1
```

pc1.startup

```
/etc/init.d/networking start
```

pc2/etc/network/interfaces

```
auto lo
iface lo inet loopback
```

```
auto eth0
iface eth0 inet static
    address 10.0.1.101
    netmask 255.255.255.0
    gateway 10.0.1.1
```

pc2.startup

```
/etc/init.d/networking start
```

Lab0 set-up

```
router/etc/network/interfaces:
```

```
auto lo
```

```
    iface lo inet loopback
```

```
auto eth0
```

```
    iface eth0 inet static
```

```
    address 10.0.0.1
```

```
    netmask 255.255.255.0
```

```
auto eth1
```

```
    iface eth0 inet static
```

```
    address 10.0.1.1
```

```
    netmask 255.255.255.0
```

```
router.startup:
```

```
/etc/init.d/networking start
```

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

/proc/sys/net/ipv4

- The /proc filesystem acts as an interface to internal data structures in the kernel
- It can be used to obtain information about the system and to change certain kernel parameters at runtime
- Nice link:
 - http://www.linuxinsight.com/proc_filesystem.html
- The /proc/sys/net contains subdirectories concerning various networking topics
- In particular, /proc/sys/net/ipv4 contains sysctls which tune different parts of the IPv4 networking stack
 - EX: `ip_forward`, `ip_default_ttl`, `ip_echo_ignore_all`, `tcp_congestion_control`
- “`echo 1 > /proc/sys/net/ipv4/ip_forward`” means: “enable IP forwarding”, i.e.: forward IP packets not addressed to us
- “`echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all`” means: don’t reply to ICMP echo request

How do I configure the IP stack at runtime?

- GNU Linux provides different tools for network configuration (`net-utils`)
 - `ifconfig`, `route`, `arp`, `netstat`, etc...
- `net-utils` have not been maintained since 2001
- `iproute2` is a collection of utilities that replaces `net-utils`

purpose	legacy "net-tools"	iproute2
Address and link configuration	<code>ifconfig</code>	<code>ip addr</code> , <code>ip link</code>
Routing tables	<code>route</code>	<code>ip route</code>
Neighbors	<code>arp</code>	<code>ip neigh</code>
VLAN	<code>vconfig</code>	<code>ip link</code>
Tunnels	<code>iptunnel</code>	<code>ip tunnel</code>
Multicast	<code>ipmaddr</code>	<code>ip maddr</code>
Statistics	<code>netstat</code>	<code>ss</code>

Essential `iproute2` commands

- For a complete doc → `man ip`
- Let's see the “must know” commands (assuming `eth0` interface available)
- Note: commands can be truncated. Ex: `ip r`, `ip n`, `ip addr`, etc...

- Show interfaces
 - `ip link show`
- Bringing interface up/down
 - `ip link set eth0 (up|down)`
- Set MAC address
 - `ip link set eth0 address 00:11:22:33:44:55`
- Set MTU
 - `ip link set eth0 mtu 1486`
- Enable/disable ARP
 - `ip link set eth0 arp (on|off)`

iproute2 essentials cont.d...

- Show IP address
 - `ip address show [dev eth0]`
- Add/remove IP address
 - `ip address (add|del) 10.0.0.1/8 dev eth0`
- Flush all address
 - `ip address flush [dev eth0]`
- Q: what if you forget the “/network_prfx_len” suffix?

- List/flush routing table
 - `ip route (list|flush)`
- Add/del route (next hop, default, direct forwarding)
 - `ip route (add|del) 100.0.0.0/8 via 10.0.0.1`
 - `ip route (add|del) default via 10.0.0.1`
 - `Ip route (add|del) 10.0.0.0/24 dev eth0`

iproute2 essentials cont.d...

- Show ARP cache

 - `ip neigh show [dev eth0]`

- Flush ARP cache

 - `ip neigh flush dev eth0`

- Add/del ARP cache entry

 - `ip neigh (add|del) to 10.0.0.2
lladdr 00:11:22:33:44:55 dev eth0
state "state_name"`

(state_name: permanent, stale, noarp, reachable)

iproute2 advanced...

- IP policy based routing
 - `ip rule`
- IP xfrm framework configuration (eg: IPSEC)
 - `ip xfrm`
- Monitor IP events
 - `ip monitor`
- IP tunneling (IPinIP, IPinGRE, IPv6 tunneling)
 - `ip tunnel`

Exercise 1 (in class)

- Let's go back to Lab0-interfaces
- Remove `/etc/network/interfaces` files for all VMS and the networking script startup
- Reconfigure everything with `iproute2`
- Put the configuration commands in the startup scripts (e.g.: `router.startup`)

Solution (Lab0-manual)

pc1.startup:

```
ip link set eth0 up
ip address add 10.0.0.101/24 dev eth0
ip route add default via 10.0.0.1
```

pc2.startup:

```
ip link set eth0 up
ip address add 10.0.1.101/24 dev eth0
ip route add default via 10.0.1.1
```

router.startup:

```
ip link set eth0 up
ip link set eth1 up
ip address add 10.0.0.1/24 dev eth0
ip address add 10.0.1.1/24 dev eth1
```

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

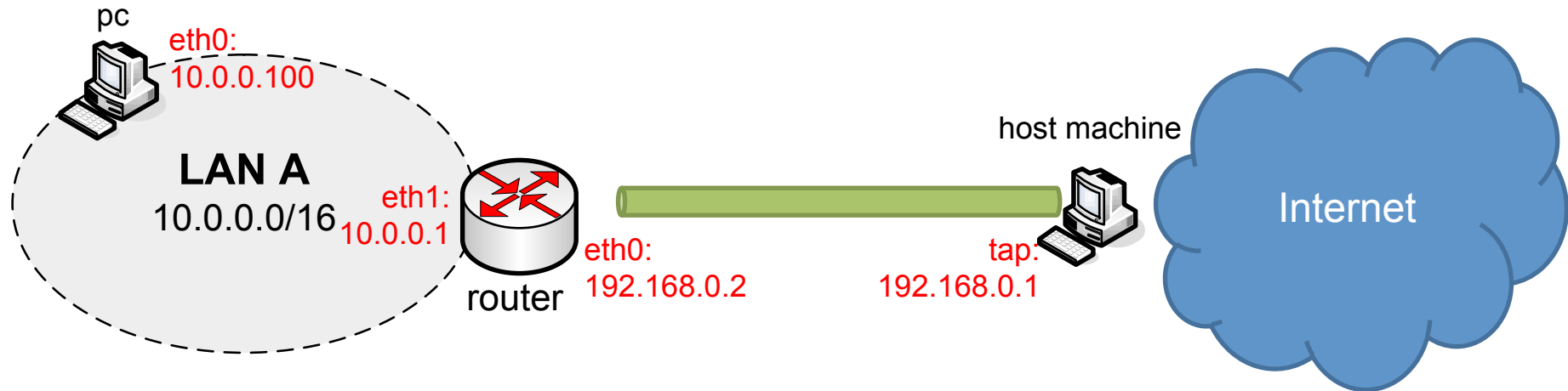
SOMETHING MORE ABOUT NETKIT

How to connect to the real world

- `/hosthome` directory in the VM is a link to the home of the user that launched the VM

- A VM can be set with a TAP interface
 - a TAP interface is a network interface connecting the VM and the host machine
 - If the host machine is connected to the internet the VM can use the host machine as the default GW to the internet
 - With `vstart`:
 - `vstart vm --eth0=tap,10.0.0.1,10.0.0.2`
 - “vm” is whatever name
 - The first IP address is the TAP address on the host machine
 - The second IP address is the address of eth0 on the VM machine
 - The IP addresses can be whatever IP addresses as long as they are in different subnet with respect to any other interfaces
 - With `lstart`:
 - `vm[0]=tap,10.0.0.1,10.0.0.2`
 - To delete a “zombie” TAP (you may need to bring it down first...)
 - `tunctl -d "tap_name"`

TAP interface example



Lab0-tap

lab.conf

```
router[0]=tap,192.168.0.1,192.168.0.2  
router[1]=A  
pc[0]=A
```

router.startup

```
ip link set eth1 up  
ip address add 10.0.0.1/16 dev eth1  
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

pc.startup

```
ip link set eth0 up  
ip address add 10.0.0.100/16 dev eth0  
ip route add default via 10.0.0.1
```

Lab0-tap

host machine

```
marlon@marlon-vmxnb:~/Labs/Lab0-tap$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:e2:37:0e
          inet addr:172.16.166.147  Bcast:172.16.255.255  Mask:255.255.0.0
          inet6 addr: fe80::20c:29ff:fee2:370e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:893530 errors:0 dropped:0 overruns:0 frame:0
          TX packets:402290 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:986911571 (986.9 MB)  TX bytes:98860361 (98.8 MB)
          Interrupt:19 Base address:0x2000

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:41240 errors:0 dropped:0 overruns:0 frame:0
          TX packets:41240 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:26599945 (26.5 MB)  TX bytes:26599945 (26.5 MB)

nk_tap_marlon Link encap:Ethernet  HWaddr c2:fd:58:73:d7:31
          inet addr:192.168.0.1  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::c0fd:58ff:fe73:d731/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:46 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:468 (468.0 B)  TX bytes:11025 (11.0 KB)

marlon@marlon-vmxnb:~/Labs/Lab0-tap$ ip r
default via 172.16.166.2 dev eth0
172.16.0.0/16 dev eth0 proto kernel scope link src 172.16.166.147
192.168.0.0/24 dev nk_tap_marlon proto kernel scope link src 192.168.0.1
```


Lab0-tap

router virtual machine

```
router
router login: root (automatic login)
Last login: Thu Mar  8 00:06:19 UTC 2012 on tty1
router:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 0a:ab:64:91:09:80
          inet addr:192.168.0.2  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::8ab:64ff:fe91:980/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:21  errors:0  dropped:0  overruns:0  frame:0
          TX packets:6  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4311 (4.2 KiB)  TX bytes:468 (468.0 B)
          Interrupt:5

eth1      Link encap:Ethernet  HWaddr a2:3a:ea:e6:6e:43
          inet addr:10.0.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::a03a:eaff:fee6:6e43/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3  errors:0  dropped:0  overruns:0  frame:0
          TX packets:6  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:188 (188.0 B)  TX bytes:468 (468.0 B)
          Interrupt:5

router:~# ip r
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.2
10.0.0.0/16 dev eth1 proto kernel scope link src 10.0.0.1
default via 192.168.0.1 dev eth0
router:~#
```

How to permanently write the netkit FS

Two ways:

1. Launch the VM with the “-W” option

```
vstart vm -W
```

1. Mount the FS file in loop

```
mount -o loop,offset=32768 \  
$NETKIT_HOME/fs/netkit-fs /mnt/nkfs
```

How can I permanently add packages to the VM?

- The “TAP way”

1. Start a VM with a tap and with the `-W` option
2. Connect the Host machine to the internet.
3. configure a name server inside the vm `/etc/resolv.conf`
4. Run `apt-get` on the VM (perhaps you will need to run `apt-get update` first)

- The “chroot way”

1. Bind the `proc/` and `dev/` in the netkit FS
2. `chroot` inside the FS mounted as in the previous slide
3. configure a name server inside the chrooted `/etc/resolv.conf`
4. Run `apt-get update` and install

DYNAMIC CONFIGURATION WITH DHCP

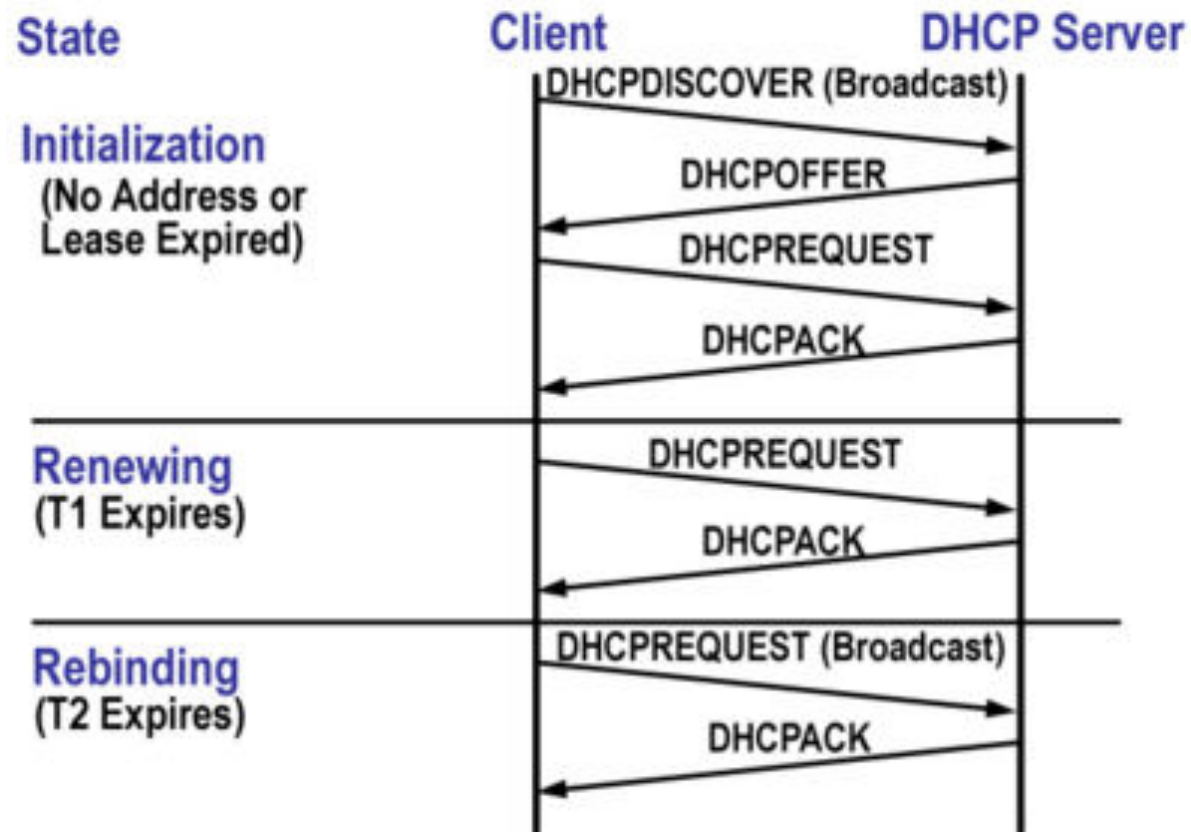
DHCP

- Dynamic Host Configuration Protocol (DHCP) is a network configuration protocol for hosts on IP networks
- A DHCP client obtains from a DHCP server a set of configuration parameters, typically:
 - IP address/netmask (for a given lease time)
 - Default GW
 - DNS server
 - Domain name
 - Search name list
 - NetBIOS name server
 - SMTP server

DHCP basics

- 4 way handshake
 - Discover, Offer, Request, ACK
- Works with multiple DHCP servers on the same LAN (DHCP Release message)
- The Client broadcast (typically at startup) the discover and receive one or more offer from the Server(s) – (then the protocol continues, but we don't care for now...)
- The Client can Renew (/Rebind) a lease for a previously assigned IP address
- 1 DHCP server for each LAN
 - DHCP-Relays allow DHCP communication through routers

DHCP handshakes



DHCP in Linux

- ISC DHCP is the most used opensource DHCP implementation
 - <http://www.isc.org/software/dhc>
- Provides:
 - DHCP Client (`dhclient`), Server (`dhcp3-server`), Relay (`dhcrelay`)
- ISC DHCP client and sever are already in the NETKIT VM filesystem
 - DHCP relay can be installed with `apt-get`

NETKIT lab with DHCP

Lab0-dhcp

Same topology as in Lab0-interfaces.

Differences:

1) In `router.startup` add the following command:

```
/etc/init.d/dhcp3-server start
```

2) In `pc{1,2}/etc/network/interfaces` remove the static configuration and add:

```
auto eth0
iface eth0 inet dhcp
```

3) Create the DHCP server configuration file in `router/etc/dhcp3/dhcpd.conf` (see the next slide)

4) Router has also a tap to the outside world:

```
router[2]=tap,192.168.0.1,192.168.0.2
```

5) `lab.dep` to start router first

DCHP server configuration

```
default-lease-time 3600;
option domain-name-servers 8.8.8.8;
option domain-name "lab0-dhcp.org";
option domain-search "lab0-dhcp.org";

subnet 10.0.0.0 netmask 255.255.255.0 {
    range 10.0.0.100 10.0.0.254;
    option routers 10.0.0.1;
}

subnet 10.0.1.0 netmask 255.255.255.0 {
    range 10.0.1.100 10.0.1.254;
    option routers 10.0.1.1;
}
```