

Network Administration and Monitoring

Alessandro Barenghi

Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

alessandro.barenghi - at - polimi.it

March 31, 2015

Recap

What did we do last time?

- Became acquainted with the most common system administration tools
- Learnt to monitor the system state in the large (`ps`, `top`)
- Learnt to follow the behaviour of a process at system level (`strace`)
- Learnt to monitor the file/socket activity of the processes on the system (`lsof`)

Network Management

Managing the network

- Now that we have the skills to use a system and manage it properly in local ...
- ... we can tackle networking!
- Before starting to employ the system APIs to program, we will learn how to manage the network facilities
- After learning how to manage the networking facilities, we will learn how to inspect the actual network traffic

Netkit

“The poor man’s system to experiment computer networking”

- To learn the basics of network administration it is useful to have a toy network to play with
- The Netkit toolkit provides a simple and fast way to emulate one faithfully on a host if you do not have a real one
- It employs User Mode Linux to run lightweight virtual machines, and emulates L2 collision domains
- You can emulate an entire network on your machine, with minimal effort and **practice**

Netkit

A minimal bootstrap guide

- The `vstart` command starts a VM and sets the network cards on a specific collision domain
 - e.g `vstart client --eth0=0 --xterm=konsole --mem=128`
- The `vhalt` and `vcrash` halt a VM, either gracefully or not
- The `vlist` command lists the running virtual machines together with their used resources
- The machines are started and a terminal attached to each one of them, logged as the administrator, is available

Network Management Suite

Managing the network

- Network management is intrinsically split between userspace and kernelspace (the network stack resides in kernelspace in Linux, you don't)
- Before 1999 a number of different solutions were employed
- After 1999, the Netlink interface was developed and the IPROUTE2 suite was born
- Old tools (such as `ifconfig` and `route`) are still maintained for compatibility reasons (and somewhat widely used)
- We will focus on the IPROUTE2 suite, since the old toolset is deprecated

A unified tool : ip

One tool to bind them all

- Management of the network is done at each ISO/OSI level from 2 to 5^a
- `ip` commands all share the same structure:
`ip [options] object command`
- `ip link` and `ip neigh` manage Level 2 (MAC)
- `ip addr` and `ip route` manage Level 3 (IP)
- `ip rule` allows for “non orthodox” IP routing (choosing on something different from IP/network prefix)
- Level 4 traffic control is demanded to the `tc` tool

^aLevel 1 management is left to digital electronic courses

Inner working

The NETLINK protocol

- The whole communication between the tool and the kernel network facilities is via NETLINK protocol
- NETLINK sockets are managed exactly as regular socket as far as primitives go (they use `connect`, `send`)
- Custom tools for communication with the kernel facilities can be written simply in C
- This provides a unified interface, with a single communication point, reducing safety/security issues

Link Layer

Modifying link layer addresses

- We will deal only with Ethernet link layer addresses^a
- The tool supports also other, less common, link layers
- `ip link show` will list all the devices and show their L2 address
- `ip link set <device> address <MAC address>` changes your current MAC address with something else
- `ip link set wlan0 arp [on|off]` toggles the ARP protocol, in case you do not want it

^aNote that 802.11b/g/n network emulate an Ethernet MAC layer for our happiness

Link Layer

ARP Tables Management

- ARP tables bind L2 (MAC) to L3 (IP) addresses and are automatically filled if the ARP is enabled on the device
- Adds a line to the ARP table

```
ip neigh add <IP address> lladdr <MAC address> dev <device>
```

- Updates a line in the ARP table

```
ip neigh change <IP address> lladdr <MAC address> dev <device>
```

- It is also possible to set the NUD^a by hand:
 - **permanent** : will never change and is used forever
 - **noarp** : will expire regularly without being checked again
 - **reachable** : regular behaviour
 - **stale** : forces re-checking

^aNeighbour Unreachability State

Link Layer

ARP cache

- The ARP cache in Linux keeps a table with the reachability status and an age counter

| entry state | meaning | action if used |
|-------------|-------------------------------|--------------------------|
| permanent | never expires, never verified | reset ctr |
| noarp | expires, never verified | reset ctr |
| reachable | normal expiration | reset ctr |
| stale | usable, must verify | reset ctr, state → delay |
| delay | schedule request, must verify | reset ctr |
| probe | sending request | reset ctr |
| incomplete | first ARP request sent | send ARP request |
| failed | no response received | send ARP request |

Network Layer

IP address

- IP address management is the by far most common task you'll be performing
- `ip address show` will simply list the ip addresses assigned to the interfaces
- An interface can be bound to more than a single addresses without the need to create an alias as it happened in old times

- To add an address to an interface:

```
ip addr add <IP address>/<netmask length> dev <device>
```

- To remove an address from an interface:

```
ip addr del <IP address>/<netmask length> dev <device>
```

Network Layer

IP address

- Different addresses with different network masks bound to the same interface are dealt in the regular way (since no aliasing may issue)
- The default broadcast address is set to `0.0.0.0`, which may not be what you want...
- The option `broadcast <address>` allows to specify a broadcast address
- `ip addr flush to <IP address>/<netmask length>` will wipe a class of addresses from any interfaces

Network Layer

Routing

- Route table management is still performed via the `ip` tool
- The IP routing tables perform exactly as you have seen in the previous courses :
 - The address with the longest matching prefix is selected
 - If two address with the same prefix are matched, the one with matching TOS is selected
 - If both address prefix *and* TOS match, the first route is selected
- As always , the default route is specified as the `0.0.0.0/0` address

Network Layer

Routing

- Adding a route is as simple as

```
ip route add <address>/<mask length> via <address>
```

- To remove a route simply use

```
ip route del <address>/<mask length> via <address>
```

- You can enforce the packets going out of a specific interface by adding `dev <interface>` at the end
- You can specify more than one device, exploiting kernel multipath, but be careful on handling the different addresses!

Network Layer

Routing - 2

- Coherently with the link layer, `ip route flush` wipes all the routes
- In need to know where your packets are going?
`ip route get <address>/<mask length>` will return the route
- `ip route show` instead shows all the routes on the system
- It is also possible to specify NAT routes via
`ip route add nat <address> via <router>`^a

^awe will deal with Network Address Translation in details during the NetFilter lessons

Network Monitoring

What should we look for?

- Network monitoring relies on either capturing the network traffic or monitoring the connection statuses
- It's useful to debug ill behaved configurations or programs
- It's also useful to understand whether or not sensitive information is transmitted on the net without being encrypted
- A couple of tools are available to perform network monitoring and dump the passing data to disk

Host Network Status

The Socket Stats tool `ss`

- Socket Stats is a part of the IPROUTE2 suite, its old-school counterpart is `netstat`
- Invoking the tool without parameters lists all the sockets open on the platform
- The output is formatted in such a way to be easy on the eyes when piped into `less`
- By default the *known ports* are listed with the service name instead of the port number
 - Known ports can be read from the `/etc/services` file in your distribution

Host Network Status

ss: useful options

- The **-n** option prints the numerical values for the ports
- The **-l** option prints only the listening sockets
- The **-i** option prints extensive info on the sockets such as the average transmission rate
- The **-t | -u | -w** options print only TCP,UDP or RAW sockets respectively

Live Traffic Analysis and Dumping

Tools

- A number of tools able to analyze live traffic and dump it to file are available
- Almost all of them rely on the libPCap libraries to perform captures and read from dump files
- We will see:
 - A selective inspection and dumping tool: `tcpdump`
 - A dump and inspection tool with a GUI: Wireshark
 - A basic dissection tool: `ngrep`

Traffic Dumping

TcpDump

- `tcpdump` provides a way to collect packets from one (or more) interfaces
- The default behaviour of the tool is to print out on screen a description of the packets flowing
- The `-i <device>` option restricts the sniffing to a single device
- The `-w <filename>` saves the eavesdropped packets to a file for “future reuse”
- The `-s <snaplen>` instructs tcpdump to capture only the first `<snaplen>` bytes of the packets
- You can specify any standard Berkley Packet Filter expression to restrict the packets to be captured

Traffic Dumping

Wireshark, or “the tool once known as Ethereal”

- In order to perform in depth packet analysis `tcpdump` is not really comfortable
- Wireshark provides a comfortable GUI to dig into the packet contents
- The program is also equipped with a number of protocol dissector covering a *large* amount of communication protocols
- We will now see a couple of samples from packet captures^a

^aYou can get more from here

<http://uluru.ee.unsw.edu.au/~tim/zoo/index.html>

Traffic Dumping

Ngrep

- Wireshark is well suited for precise analysis of reasonably small packet quantities
- As the name suggests, `ngrep` acts exactly as the `grep` tool, just on packet dumps or live interfaces
- The common use is `ngrep -d <device> [bpf]` or `ngrep -I <input file> [bpf]`
- The `-W byline` option controls output formatting enabling greater readability
- The `-K` option kills (sending a RST packet) the tcp connections matching the BPF expression