Network Administration and Monitoring

Alessandro Barenghi

Dipartimento di Elettronica e Informazione
Politecnico di Milano

barenghi - at - elet.polimi.it

March 30, 2011

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

Network Management Suite

Managing the network

- Network management is intrinsically split between userspace and kernelspace
- Before 1999 a number of different solutions were employed
- After 1999, the Netlink interface was developed and the IPROUTE2 suite was born
- Old tools are still mantained for compatibility reasons (and widely used)
- We will focus on the IPROUTE2 suite, since the old toolset is deprecated

A unified tool : `ip`

Management of the network is done at each ISO/OSI level from 2 to 5[1] :

One tool to bind them all

- `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)
- Level 4 traffic control is demanded to the `tc` tool

[1] Level 1 management is left to digital electronic courses

Inner working

The NETLINK protocol

- The whole communication between the tool and the kernel network facilites is via NETLINK protocol
- NETLINK sockets are managed exactly as regular socket as far as primitives go
- 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 (or equivalent) link layer addresses
- The tools support 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

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
- `ip neigh add <IP address> lladdr <MAC address> dev <device>` adds a line to the ARP table
- `ip neigh change <IP address> lladdr <MAC address> dev <device>` updates a line in the ARP table
- 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

[a] Neighbour Unreachability State

Network Layer

IP address

- IP address management is the 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 in old tools
- `ip addr add <IP address>/<netmask length> dev <device>` will add an address to an interface

Network Layer

IP address

- Different addresses with different network masks are dealt in the regular way (since no aliasing may issue)
- In case more than an address with the same mask length is bound to the interface, one will be selected as the default for the traffic generating from the machine
- the option `broadcast [+|-|address]` allows to specify a broadcast address
- `ip addr del <IP address>/<netmask length> dev <device>` removes an address from the interface
- `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>`
- You can enforce the packets down 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!
- To remove a route simply use `ip route del <address>/<mask length> via <address>`

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]

[a] we 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 informations are passed in clear on the net
- A couple of tools are available to perform network monitoring

Host Network Status

The Socket Stats tool `ss`

- Socket Stats is a part of the IPROUTE2 suite
- 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

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 infos on the sockets such as the average transmission rate
- The `-t | -u | -w` options print only TCP,UDP or RAW sockets respectively

Traffic Dumping

Tools

- A number of tools able to dump the flowing traffic are available
- Almost all of them rely on the `libPCap` libraries
- We will see
  - A plain dumping tool: `tcpdump`
  - A dump + analyse tool: 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"

Traffic Dumping

Wireshark, or "the tool once known as Ethereal"

- In order to perform in depth packet analysis `tcpdump` is not sufficient
- 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]

[a] You 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