COMP09024 Unix System Administration

Lecture 9: Network Configuration

Duncan Thomson/Hector Marco

UWS

Trimester 1 2020/21

Outline

9.1 Configuration

- Network Devices
- Interface Configuration
- Routing
- Name Resolution
- Debian Configuration

9.2 Troubleshooting

- IP connectivity
- DNS Troubleshooting
- `netstat`

9.3 Network Services

- Secure Shell
- Printing
- NFS
- NIS

9.1 Configuration

Network Configuration

- Network configuration (for standard machines) usually involves correctly setting up three things:
  - Setting up the correct address(es) and other parameters on the interfaces
  - Ensuring a route is available for remote networks (a ‘default route’ or ‘default gateway’)
  - Providing a method of resolving textual names into addresses
- This can be done using well known commands
- Most distributions also have a mechanism for doing this using configuration files and/or the GUI

Network Devices

- Network interfaces are (generally) an exception to the ‘everything is a file’ rule
- Most Unices use a symbolic name created from:
  - Letters indicating the type of interface (eg `eth`, `wlan`, `ppp` etc)
  - A number allowing differentiation between multiple interfaces
- Examples in Linux: `eth0`, `eth1`, `wlan0`
- Various commands can examine and configure these interfaces:
  - `ifconfig`
  - `ip`

Predictable Interface Names I

Modern versions of Linux do not use `eth` like names any more:

- `Systemd` parallelises `init.d` scripts to speed up the boot process.
- Don't stop the boot process for the Internet connection to come up, slow DHCP servers, etc.
- Launching scripts in parallel during startup/shutdown introduce issues:
  - There are not "execution order" for those (e.g. S01xxxx could be executed after S02xxx)
  - Systems with more than 1 `ethX` card will assign `eth0` to the **first** driver loaded.
- This introduces security and functionality issues (firewalls, routing, etc.)

Predictable Interface Names II

Tho prevent unpredictable kernel-native ethX naming (e.g. eth0) modern versions of Linux use “Predictable Interface Names”:

- Firmware/bios-provided index numbers for on-board devices (e.g. `eno1`)
- Firmware-provided pci-express hotplug slot index number (e.g. `ens1`)
- Physical/geographical location of the hardware (e.g. `enp2s0`)
- The interface’s MAC address (e.g. `enx78e5d1ea81da`)

ifconfig I

- `ifconfig` on its own prints status of active (‘up’) interfaces
- Output includes addresses, statistics and settings:

```
eth0    Link encap:Ethernet  HWaddr 6c:3b:e5:1c:41:56
        inet addr:10.0.1.1  Bcast:10.0.15.255  Mask:255.255.240.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:62326 errors:0 dropped:296 overruns:0 frame:0
        TX packets:803 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:3880546 (3.7 MiB)  TX bytes:51352 (50.1 KiB)
```

- `-a` flag lists all interfaces (including inactive)
- `-s` gives only a summary with statistics

`ifconfig` II

- `ifconfig` with an interface name can also be used to configure interfaces
  - `up` or `down` bring interface up or down
  - IPv4 addresses can be specified directly
  - Subnet masks (for IPv4) are given using slash-notation (eg `/24`) or using the `netmask` keyword
  - `add` and `del` used to add/delete IPv6 addresses
- Examples:

```
ifconfig eth0 10.11.0.1/24 up
ifconfig eth1 192.168.1.1 netmask 255.255.255.240
ifconfig eth0 add 2001:db8::1/64
ifconfig wlan0 down
```

ip

- The `ip` command is part of the 'new' Linux `iproute2` suite
- It has been available for more than 10 years, but has been slow to replace traditional Unix commands
- `ip link` allows configuration of some layer 2 properties of interfaces (eg VLAN tagging)
- `ip addr` allows configuration of layer 3 addresses on interfaces
- Examples:

```
ip link show dev eth0
ip link dev eth0 up
ip addr show
ip addr add 10.0.1.1/24 dev eth0
ip addr 2001:db8::1/64 dev eth0
```

Wireless Interfaces

- Wireless interfaces typically require additional layer-2 configuration:
  - Mode
  - Channel number
  - ESSID
  - Encryption settings (type, key, etc)
- Linux provides a number of commands in the ‘wireless-tools’ package to help with this, including:
  - `iwlist` can scan for active access points
  - `iwconfig` provides configuration of WLAN interface for settings such as those above

Routing and the Routing Table

- Once an interface is configured, this provides access to the local network
- For access to remote networks, the address of a ‘next hop’ router is required
- The routing table contains information about routes (for both local and remote routes)
- Routing table entries specify:
  - Destination (specified as a network and subnet mask)
  - How to reach the destination (interface or next-hop IP address)
  - Metric (lowest metric is best)
- If more than one routing table entry matches:
  - The most specific (longest subnet mask) route is used
  - Then the route with the lowest metric is used

Examining the Routing Table with `route`

- The `route` command (without arguments) shows the routing table
- The `-n` can prevent name lookup for speed
- Example output of the `route -n` command (in Linux):

```
Kernel IP routing table
Destination Gateway     Genmask       Flags Metric Ref Use Iface
0.0.0.0     10.0.1.254 0.0.0.0        UG    0      0     0 eth0
10.0.1.0    0.0.0.0    255.255.255.0 U     0      0     0 eth0
```

- The `ip route` command can also be used with Linux for examining and changing the routing table

Adding / Removing Routes with `route`

- Routes can also be added and removed with `route`:
  - `route add` adds a route
  - `route del` removes a route
- Destinations can be specified as:
  - `-net` with a network and subnet mask (slash notation or `netmask` keyword)
  - `-host` to specify a single host (/32)
  - `default` for 0.0.0.0/0 network (default)
- Next hop (gateway) specified with `gw` and IP address
- `metric` keyword can specify metric (1 is default)
- Examples:

```
route add -net 10.1.0.0/16 gw 10.0.1.9 metric 3
route add default gw 10.0.1.254
route del -host 10.0.1.5
```

Name Resolution

- Computers use layer 3 (eg IPv4, IPv6) addresses to send packets to one another
- Humans generally prefer to use textual names to refer to machines
- Name resolution is the process by which names are translated to IP addresses
- Two main methods are used for this:
  - Local lookup using the `/etc/hosts` file
  - Using Domain Name System (DNS) server
- `hosts` entry in `/etc/nsswitch.conf` controls how this is done — normally `/etc/hosts` first and then DNS

The /etc/hosts File

- Every Unix machine has an `/etc/hosts` file
- Each line consists of:
  - An IPv4 (or IPv6) address
  - One or more names which can be used to refer to the machine with this address
- Example:

```
# comments begin with the hash symbol
127.0.0.1   localhost  # IPv4 loopback address
10.0.1.7    grabowski.example.org  grabowski
10.0.1.11   kalocsay.example.org  kalocsay
::1         localhost ip6-localhost ip6-loopback
```

Configuring DNS Operation

- DNS operation is usually controlled from the `/etc/resolv.conf` (sic) file
- `nameserver` entries specify IP address of best DNS servers to use
- Always best to specify more than one (for backup)
- Other entries in this file may include:
  - `domain` — domain name, added to short names before searching
  - `search` — list of multiple domains in which to search for short names
- Example `/etc/resolv.conf`:

```
domain example.org
nameserver 10.0.1.250
nameserver 10.0.1.251
```

Network Configuration in Debian

- Normally the commands outlined above are not used for day-to-day configuration
- In Debian, there are two methods generally used for configuring network interfaces:
  - File-based, using `/etc/network/interfaces` and other files
  - GUI-based, using NetworkManager
- If an interface is configured in `/etc/network/interfaces`, it cannot be configured via the GUI

Using `/etc/network/interfaces` I

- Contains two main types of entry:
  - Entries specifying when interfaces should be brought up
  - `iface` entries specifying interface configuration
- Examples of the first include:
  - `auto` — interface should be brought up automatically
  - `allow-hotplug` — interface automatically configured by hotplug system
- `iface` lines should include:
  - Interface name
  - Address family (`inet` or `inet6`)
  - Method of configuration (eg `static`, `dhcp` or `loopback`)
- `static` configuration entries require further configuration parameters on following line

Using /etc/network/interfaces II

- Static configurations can include (usually as a minimum):
    - `address` — specifying IP address
    - `netmask` — specifying subnet mask
    - `gateway` — specifying default gateway's IP address
- Additional lines under `iface` might include:
    - Entries beginning with `dns-` for DNS parameters to be added to `/etc/resolv.conf` (if `resolvconf` package installed)
    - Entries beginning with `wireless-` for WLAN parameters
- The `ifup` and `ifdown` commands can bring up and down interfaces with the configurations specified in the file marked as `auto`

Using /etc/network/interfaces III

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
   address 10.0.1.7
   netmask 255.255.255.0
   gateway 10.0.1.254

allow-hotplug eth1
iface eth1 inet dhcp
```

Using NetworkManager

- Debian also allows GUI configuration of interfaces
- Best for user-administered client machines
- Interfaces which are not specified in `/etc/network/interfaces` are managed by NetworkManager
- Two components run:
  - A service (daemon) which runs in the background
  - A user-interface allowing status and configuration
- User interface may be GUI-based (accessible through the desktop) or command-line based
- Most common settings are accessible (including IPv6, and more advanced settings such as static routes)

9.2 Troubleshooting

Troubleshooting

- When the network is not operating correctly, the normal order of things to check is:
  1. Local configuration
  2. Connectivity to local network(s)
  3. Connectivity to remote network(s) via gateways
  4. Correct operation of DNS
  5. Application-layer checks (including proxy servers)
- Tools outlined previously can be used to help with this (eg `ifconfig`, `route`, `ip`)
- `arp` (or `ip neigh`) can show L2/L3 mappings (ARP table)
- Other tools (on following slides) can provide further information

Checking IP connectivity with `ping`

- `ping` command sends ICMP echo requests to destination, checks that ICMP echo replies are returned (with round trip times)
- Different versions of Unix may show slightly different behaviour
- On Linux, the default is to continue sending ICMP requests until stopped with Ctrl-C
- Various flags can control behaviour:
  - `-c` — specify how many packets to send
  - `-I` — specify source interface
  - `-W` — specify timeout
  - `-s` — specify packet size
  - `-t` — specify packet TTL

The `traceroute` Command

- If a remote routing problem is suspected (cannot reach some remote networks), a traceroute can locate the problem
- Sends ICMP echo requests with increasing TTLs (starting at 1)
- Packets with TTL of 1 are returned from first hop router
- Packets with TTL of 2 are returned from second hop router
- (and so on. . .)
- `-n` turns off reverse DNS lookup (for speed)
- `-q` specifies number of packets per hop (default is 3)

DNS Troubleshooting with `host`

- If DNS is not functioning ('Host not found') first check
  - Correct DNS servers specified in `/etc/resolv.conf` (or in NetworkManager)
  - DNS servers are reachable with `ping`
- Then `host` command can be used to send DNS queries:
  - Default is for any records from default servers
  - Servers can be specified as second parameter
  - `-t` allows specific types of query
- Examples:

```
host www.example.org
host www.example.org 10.0.1.10
host -t MX example.org
```

The `netstat` Command

- `netstat` with various flags can also show information about interfaces, ARP tables and routing tables
- However, its most useful purpose is to show current connections and listening servers
- Most communication between machines takes place over a TCP or UDP ‘socket’ (source/destination IP, source/destination port number)
- `netstat` can summarise information on these sockets
- `-t` shows TCP connections (used for most Internet applications)
- `-u` shows UDP information (used for (eg) DNS and streaming)
- `-l` shows listening ports (ie servers running on local machine)

9.3 Network Services

Network Services

- Unix machines are capable of providing a wide ranging of network services.
- Here we introduce Secure Shell (usually used to provide secure remote access to machines)
  - We'll put this into practice in the lab
- We'll also look at providing network services to groups of Unix machines:
  - Printing (using Common Unix Printing System — CUPS)
  - File sharing (using Network File System — NFS)
  - Sharing user accounts (using Network Information System — NIS)

Secure Shell — Why?

- Originally remote access to Unix machines used the telnet protocol
  - This was inflexible (only provided access to command line)
  - Also insecure (passwords and data passed in the clear)
- The ‘r’ commands (`rlogin`, `rsh` and `rcp`) replaced telnet
  - More flexible, allowing single commands to be remotely executed, and file copying
  - Passing passwords could be avoided, but still insecure
- Secure Shell (ssh) is the modern secure equivalent
  - Communications are encrypted.
  - Password is not sent to the server.
  - Provides remote commands, port forwarding, etc.

Secure Shell — Use

- Require a ssh server on the remote machine, eg in Debian the `openssh-server` package
- Software configured from `/etc/ssh/` directory
- `ssh` command allows connection to remote machine
- Hosts keys are stored and checked at each connection
- Username specified using `-l` flag or using `@` symbol
- A single command can be executed by providing the command line as an argument
- Examples:

```
ssh server.example.org
ssh alice@server.example.org
ssh bob@example.org "ls -l"
ssh server "tar cf - /etc" | cat >etc.tar
```

Secure Shell — More Advanced Uses

- `scp` can copy files over a ssh connection
- Examples:

```
scp myfile bob@server.example.org:
scp myfile server.example.org:newname
scp files* server.example.org:
scp server.example.org:myfile .
```

- A number of additional abilities of the `ssh` command:
  - `-L` and `-R` support local and remote port forwarding (security implications)
  - `-X` forwards X11 protocol (allowing local display of remote applications)
  - `-w` creates a tunnel
- `ssh-keygen` and `ssh-agent` provide key management abilities

Printing

- Printing in Unix has gone through a number of generations
- BSD printing supported network printing; complex config
- LPRng improved over some disadvantages of BSD system
- Current system is Common Unix Printing System (CUPS)
  - Developed originally by Apple
  - Supports Internet Printing Protocol (IPP)
  - Makes use of a wide range of printer profiles and PPD (PostScript Printer Description) files
- Administered easily via a web browser pointing at `http://localhost:631/` — or even over network
- Can also be administered using `lpadmin` command — or by configuration files in `/etc/cups/`
- Printers can be shared over the network using IPP

Printing — User Commands

- BSD-compatible commands. . .
- `lpr` — used to submit a print job
  - `-P` specifies printer name (if not default)
  - `-#` specifies number of copies
- `lpq` — shows printer queue
- `lprm` — removes a print job (by ID)
- Also System-V equivalents: `lp`, `lpstat` & `cancel`
- Many GUI programs allow direct submission without use of commands

Network File System — NFS

- A distributed filesystem developed by Sun Microsystems
- Based on Remote Procedure Call (RPC) — requires RPC portmapper to be running
- Server 'exports' a filesystem, which can then be mounted by a client in the normal ways:
    - By using the `mount` command
    - By placing in `/etc/fstab`
- Filesystem type is `nfs`
- Filesystem 'source' is `servername:/exportname`
- For correct operation within a network require:
    - Careful use of export operations to prevent `root` access
    - Mapping of UIDs between server/client (eg using NIS)

Running an NFS Server

- Requires running / turning on NFS server (eg installation of `nfs-kernel-server` package in Debian)
- Filesystems exported listed in `/etc/exports`
  - Name of filesystem
  - One or more client specifications with options in parenthesis *immediately* following
  - Example: `/home 10.0.0.0/24(rw) 10.0.1.0/24(ro)`
  - Documented in `exports(5)`
- `exportfs` command lists/refreshes exported filesystems
- Some options are particularly important for security:
  - `root_squash`: NFS client acessed/created files by *root* are not created as *root* on the NFS server.
- Important to ensure UIDs are correctly mapped (for security)

Network Information System — NIS

- For easy operation of NFS, UIDs should match on server and all clients
- Could update all `/etc/passwd` (etc) files when users are added or removed
- But easier to share this information over network
- NIS provides a mechanism to do this
- Originally called Yellow Pages — many commands and files still have `yp` prefix
- One or more NIS server share ‘maps’ (which are generally files in `/etc` — specifically `passwd`, `shadow`, `group` and `gshadow`, but may include others - eg `hosts`)
- Clients can ‘bind’ to these servers, and can then access the maps

Summary

- Network settings may be configured by using various commands and files or the GUI:
  - `ifconfig`, `route`, `ip` (Linux)
  - `/etc/resolv.conf`, `/etc/network/interfaces`
  - NetworkManager and associated applications (Linux)
- Troubleshooting commands include:
  - `ping` and `traceroute` for IP connectivity
  - `host` for name resolution with DNS servers
  - `netstat` for monitoring TCP and UDP traffic/servers
- Unix can host a wide range of network services, including:
  - Secure Shell for command line access (and more)
  - Printing, via CUPS
  - Sharing user files and accounts with NFS/NIS