APPENDIX B

Network Administration

This appendix reviews some of the most popular tools for network administration and debugging. These tools can help a lot in finding solutions to common problems and in developing, debugging, benchmarking, analyzing, troubleshooting, and researching network projects. Most of these tools have very good documentation resources, either with man pages or with wiki pages, and a lot of other information resources about them are on the Internet. Many of them have active mailing lists (for users and developers) and a bug reporting system. Some of the most commonly used tools are described here by specifying their purpose and relevant links, accompanied by several examples. The tools mentioned in this appendix appear in alphabetical order.

arp

This command is for ARP table management. Example of usage:

You can display the ARP table by running `arp` from the command-line. `arp -n` will display the ARP table without name resolution.

You can add static entries to the ARP table by:

```
arp -s 192.168.2.10 00:e0:4c:11:22:33
```

The `arp` utility belongs to the `net-tools` package. Website: http://net-tools.sourceforge.net.

arping

A utility to send ARP requests. The `-D` flag is for Duplicate Address Detection (DAD). The `arping` utility belongs to the `iputils` package. Website: http://www.skbuff.net/iputils/.

arptables

A userspace tool for configuring rules for a Linux-based ARP rules firewall. Website: http://ebtables.sourceforge.net/.

arpwatch

A userspace tool for monitoring ARP traffic. Website: http://ee.lbl.gov/.

ApacheBench (ab)

A command-line utility for measuring the performance of HTTP web servers. The ApacheBench tool is part of the Apache open source project. In many distributions (for example, Ubuntu) it is part of the `apache2-utils` package. Example of usage:

```
ab -n 100  http://www.google.com/
```

The `-n` option is the number of requests to perform for the benchmarking session.

brctl

A command-line utility for administration of Ethernet bridges, enabling the setup of a bridge configuration. The `brctl` utility belongs to the `bridge-utils` package. Examples for usage:

- `brctl addbr mybr`: Add a bridge named `mybr`.
- `brctl delbr mybr`: Delete the bridge named `mybr`.
- `brctl addif mybr eth1`: Add the `eth1` interface to the bridge.
- `brctl delif mybr eth1`: Delete the `eth1` interface from the bridge.
- `brctl show`: Show information about the bridge and its attached ports.

The maintainer of the `bridge-utils` package is Stephen Hemminger. Fetching the `git` repository can be done by:

```
git clone git://git.kernel.org/pub/scm/linux/kernel/git/shemminger/bridge-utils.git
```

Website: http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge.

conntrack-tools

A set of userspace tools for management of netfilter connection tracking. It consists of a userspace daemon, `conntrackd`, and a command-line tool, `conntrack`. Website: http://conntrack-tools.netfilter.org/.

crtools

A utility for checkpoint/restore of a process. Website: http://criu.org/Installation.

ebtables

A userspace tool for configuring rules for a Linux-based bridging firewall. Website: http://ebtables.sourceforge.net/.

ether-wake

A utility to send Wake-On-LAN Magic Packets. The `ether-wake` utility belongs to the `net-tools` package.

ethtool

The `ethtool` utility provides a way to query or control network driver and hardware settings, get statistics, get diagnostic information, and more. With `ethtool` you can control parameters of Ethernet devices, such as speed, duplex, auto-negotiation and flow control. Many features of `ethtool` require support in the network driver code.

Examples:

- Output of `ethtool eth0`:

```
Settings for eth0:
        Supported ports: [ TP MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
        Supported pause frame use: No
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
        Advertised pause frame use: Symmetric Receive-only
        Advertised auto-negotiation: Yes
        Speed: 10Mb/s
        Duplex: Half
        Port: MII
        PHYAD: 0
        Transceiver: internal
        Auto-negotiation: on
        Supports Wake-on: pumbg
        Wake-on: g
        Current message level: 0x00000033 (51)
                               drv probe ifdown ifup
        Link detected: no
```

- Getting offload parameters is done by: `ethtool -k eth1`.
- Setting offload parameters is done by: `ethtool -K eth1 offLoadParamater`.
- Querying the network device for associated driver information is done by: `ethtool -i eth1`.
- Showing statistics is done by: `ethtool -S eth1` (note that not all the network device drivers implement this feature).
- Show permanent hardware (MAC) address: `ethtool -P eth0`.

The development of `ethtool` is done by sending patches to the `netdev` mailing list. The maintainer of `ethtool` as of this writing is Ben Hutchings. The `ethtool` project is developed over a `git` repository. It can be downloaded by: `git clone git://git.kernel.org/pub/scm/network/ethtool/ethtool.git`.

Website: www.kernel.org/pub/software/network/ethtool/.

git

A distributed version control system started by Linus Torvalds. Linux kernel development, as well as many Linux related projects, are managed by `git`. One can also use the `git send-email` command in order to send patches by mail. Website: http://git-scm.com/.

hciconfig

A command-line tool for configuring Bluetooth devices. With `hciconfig`, you can display information such as the Bluetooth interface type (BR/EDR or AMP), its Bluetooth address, its flags, and more. The `hciconfig` tool belongs to the `bluez` package. Example:

```
hciconfig
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 00:02:72:AA:FB:94  ACL MTU: 1021:7  SCO MTU: 64:1
        UP RUNNING PSCAN
        RX bytes:964 acl:0 sco:0 events:41 errors:0
        TX bytes:903 acl:0 sco:0 commands:41 errors:0
```

Website: http://www.bluez.org/.

hcidump

A command-line utility for dumping raw HCI data coming from and going to a Bluetooth device. The `hcidump` utility belongs to the `bluez-hcidump` package. Website: http://www.bluez.org/.

hcitool

A command-line utility for configuring Bluetooth connections and for sending some special commands to Bluetooth devices. For example, you can scan for nearby Bluetooth devices by: `hcitool scan`. The `hcitool` utility belongs to the `bluez-hcidump` package.

ifconifg

The `ifconfig` command allows you to configure various network interface parameters, including the IP address of the device, the MTU, the MAC address, the Tx queue length (`txqueuelen`), flags, and more. The `ifconfig` tool belongs to the `net-tools` package, which is older than the `iproute2` package (discussed later in this appendix). Here are three examples of usage:

- `ifconfig eth0 mtu 1300`: Change the MTU to 1300.
- `ifconfig eth0 txqueuelen 1100`: Change the Tx Queue length to 1100.
- `ifconfig eth0 -arp`: Disable the ARP protocol on `eth0`.

Website: http://net-tools.sourceforge.net.

ifenslave

A utility for attaching and detaching slave network devices to a bonding device. *Bonding* is putting multiple physical Ethernet devices into a single logical one, what is often termed as Link aggregation/Trunking/Link bundling. The source file is in `Documentation/networking/ifenslave.c`. You can attach `eth0`, for example, to a bonding device `bond0` by:

```
ifenslave bond0 eth0
```

The `ifenslave` utility belongs to the `iputils` package, maintained by Yoshifuji Hideaki. Website: www.skbuff.net/iputils/.

iperf

The iperf project is an open source project that provides a benchmarking tool to measure TCP and UDP bandwidth performance. It allows you to tune various parameters. The iperf tool reports bandwidth, delay jitter, and datagram loss. It was originally developed by the Distributed Applications Support Team (DAST) at the National Laboratory for Applied Network Research (NLANR) in C++. It works in a client-server model. A new implementation from scratch, iperf3, which is not backwards compatible with the original iperf, is available from https://code.google.com/p/iperf/. The iperf3 is said to have a simpler code base. The iperf3 tool can report also the average CPU utilization of the client and the server.

Using iperf

Following is a simple example of using iperf for measuring TCP performance. On one device (which has an IP address of 192.168.2.104), run the next command, which starts the server side (by default, it is a TCP socket on port 5001):

```
iperf -s
```

On a second device, run the iperf TCP client to connect to the iperf server:

```
iperf -c 192.168.2.104
```

On the client side you will see the following:

```
------------------------------------------------------------
Client connecting to 192.168.2.104, TCP port 5001
TCP window size: 22.9 KByte (default)
------------------------------------------------------------
[  3] local 192.168.2.200 port 35146 connected with 192.168.2.104 port 5001
```

The default time interval is 10 seconds. After 10 seconds, the client will be disconnected, and you will see a message like this on the terminal:

```
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.3 sec  7.62 MBytes  6.20 Mbits/sec
```

You can tune many parameters of iperf, like these:

- -u: For using a UDP socket.
- -t: For using a different time interval in seconds instead of the default of 10 seconds.
- -T: Sets a TTL for multicast (the default is 1).
- -B: Bind to a host, an interface, or a multicast address.

See man iperf. Website: http://iperf.sourceforge.net/.

iproute2

The iproute2 package provides many tools for interaction between the userspace and the kernel networking subsystem. The most well-known is the ip command. It is based on netlink sockets (discussed in Chapter 2). With the

ip command, you can perform various operations in a wide range of networking areas, and it has numerous options; see man 8 ip. Here are several examples of using the ip command for various tasks:

- Configuration of a network device with ip addr:
  - ip addr add 192.168.0.10/24 dev eth0: Sets an IP address on eth0.
  - ip addr show: Displays the addresses of all network interfaces (both IPv4 and IPv6).

  See man ip address.
- Configuration of a network device with ip link:
  - ip link add mybr type bridge: Creates a bridge named mybr.
  - ip link add name myteam type team: Creates a teaming device named myteam. (The teaming device driver aggregates multiple physical Ethernet devices into one logical one and is in fact the new bonding device. The teaming driver is discussed in Chapter 14.)
  - ip link set eth1 mtu 1450: Sets the MTU of eth1 to be 1450.

  See man ip link.
- Management of ARP tables (IPv4) and NDISC (IPv6) tables:
  - ip neigh show: Shows both the IPv4 neighbouring table (ARP table) and the IPv6 neighbouring table.
  - ip -6 neigh show: Shows only the IPv6 neighbouring table.
  - ip neigh flush dev eth0: Removes all entries from the neighboring tables associated with eth0.
  - ip neigh add 192.168.2.20 dev eth2 lladdr 00:11:22:33:44:55 nud permanent: Adds a permanent neighbour entry (parallel to adding static entries in an ARP table).
  - ip neigh change 192.168.2.20 dev eth2 lladdr 55:44:33:22:11:00 nud permanent: Updates a neighbour entry.

  See man ip neighbour.
- Management of the parameters for the neighbour tables:
  - ip ntable show: Displays the neighbour tables parameters.
  - ip ntable change name arp_cache locktime 1200 dev eth0: Changes the locktime parameter for the IPv4 neighbouring table associated with eth0.

  See man ip ntable.
- Network namespaces management:
  - ip netns add myNamespace: Adds a network namespace named myNamespace.
  - ip netns del myNamespace: Deletes the network namespace named myNamespace.
  - ip netns list: Shows all network namespaces on the host.
  - ip netns monitor: Displays a line of screen for each network namespace that is added or removed by the ip netns command.

See `man ip netns`.

- Configuration of multicast addresses:
  - `ip maddr show`: Shows all multicast addresses on the host (both IPv4 and IPv6).
  - `ip maddr add 00:10:02:03:04:05 dev eth1`: Adds a multicast address on `eth1`.

  See `man ip maddress`.
- Monitor netlink messages. For example:
  - `ip monitor route` displays on the screen messages about various network events like adding or deleting a route.

  See `man ip monitor`.
- Management of routing tables:
  - `ip route show`: Shows the routing table.
  - `ip route flush dev eth1`: Removes routing entries associated with `eth1` from the routing table.
  - `ip route add default via 192.168.2.1`: Adds 192.168.2.1 as a default gateway.
  - `ip route get 192.168.2.10`: Gets the route to 192.168.2.10 and displays it.

  See `man ip route`.
- Management of rules in the RPDB (Routing Policy DataBase). For example:
  - `ip rule add tos 0x02 table 200`: Adds a rule that sets the routing subsystem to perform a lookup in routing table 252 for packets whose TOS value is 0x02 (TOS is a field in the IPv4 header).
  - `ip rule del tos 0x02 table 200`: Deletes a specified rule from the RPDB.
  - `ip rule show`: Displays the rules in the RPDB.

  See `man ip rule`.
- Management of TUN/TAP devices:
  - `ip tuntap add tun1 mode tun`: Creates a TUN device named `tun1`.
  - `ip tuntap del tun1 mode tun`: Deletes a TUN device named `tun1`.
  - `ip tuntap add tap1 mode tap`: Creates a TAP device named `tap1`.
  - `ip tuntap del tap1 mode tap`: Deletes a TAP device named `tap1`.
- Management of IPsec policies:
  - `ip xfrm policy show`: Shows IPsec policies.
  - `ip xfrm state show`: Shows IPsec states.

  See `man ip xfrm`.

The `ss` tool is used to dump socket statistics. For example, running

```
ss -t -a
```

will show all TCP sockets:

```
State       Recv-Q Send-Q          Local Address:Port              Peer Address:Port
LISTEN      0      32                          *:ftp                          *:*
LISTEN      0      128                         *:ssh                          *:*
LISTEN      0      128             127.0.0.1:ipp                          *:*
ESTAB       0      0               192.168.2.200:ssh             192.168.2.104:52089
ESTAB       0      52              192.168.2.200:ssh             192.168.2.104:51352
ESTAB       0      0               192.168.2.200:ssh             192.168.2.104:51523
ESTAB       0      0               192.168.2.200:59532          107.21.231.190:http
LISTEN      0      128                        :::ssh                         :::*
LISTEN      0      128                       ::1:ipp                         :::*
CLOSE-WAIT  1      0                         ::1:48723                      ::1:ipp
```

There are other tools of `iproute2`:

- `bridge`: Shows/manipulates bridge addresses and devices. For example:
  - `bridge fdb show`: Displays forwarding entries.

  See `man bridge`.
- `genl`: Gets information (like id, header size, max attributes, and more) about registered generic netlink families. For example, running `genl ctrl list` can have this as a result:

```
Name: nlctrl
        ID: 0x10  Version: 0x2  header size: 0  max attribs: 7
        commands supported:
                #1:  ID-0x3
                Capabilities (0xe):
                  can doit; can dumpit; has policy

        multicast groups:
                #1:  ID-0x10  name: notify
```

- `lnstat`: Displays Linux network statistics.
- `rtmon`: Monitors Rtnetlink sockets.
- `tc`: Shows/manipulates traffic control settings. For example:
  - `tc qdisc show`: Running this command shows which queueing discipline (`qdisc`) entries are installed, for example:

```
qdisc pfifo_fast 0: dev eth1 root refcnt 2 bands 3 priomap  1 2 . . .
```

- This shows that the `pfifo_fast qdisc` is associated with the `eth1` network device. The `pfifo_fast qdisc`, which is a classless queueing discipline, is the default `qdisc` in Linux.
  - `tc -s qdisc show dev eth1`: Shows statistics of the `qdisc` associated to `eth1`.

  See `man tc`.

  See: Linux Advanced Routing & Traffic Control HOWTO: www.lartc.org/howto/.

The development of `iproute2` is done by sending patches to the `netdev` mailing list. The maintainer of `ethtool` as of this writing is Stephen Hemminger. The `iproute2` is developed over a `git` repository, which can be downloaded by: `git clone git://git.kernel.org/pub/scm/linux/kernel/git/shemminger/iproute2.git`.

iptables and iptables6

The `iptables` and `iptables6` are administration tools for packet filtering and NAT management for IPv4 and IPv6, respectively. With `iptables/iptables6`, you can define lists of rules. Each such rule tells what should be done with the packet (for example, discard it or accept it). Each rule specifies some matching condition for a packet, for example, that it will be a UDP packet. Following are some examples for using the `iptables` command:

- `iptables -A INPUT -p tcp --dport=80 -j LOG --log-level 1`: The meaning of this rule is that incoming TCP packets with destination port 80 will be dumped to the syslog.
- `iptables –L`: Lists all rules in the filter table. (There is no table mentioned in the command, so it accesses the Filter table, which is the default table.)
- `iptables –t nat –L`: Lists all rules in the NAT table.
- `iptables –F`: Flushes the selected table.
- `iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE`: Sets a MASQUERADE rule.

Website: www.netfilter.org/.

ipvsadm

A tool for Linux Virtual Server administration. Website: www.linuxvirtualserver.org/software/ipvs.html.

iw

Shows/manipulates wireless devices and their configuration. The `iw` package is based on generic netlink sockets (see Chapter 2). For example, you can perform these operations:

- `iw dev wlan0 scan`: Scans for nearby wireless devices.
- `iw wlan0 station dump`: Displays statistics about a station.
- `iw list`: Gets information about a wireless device (such as band information and 802.11n information).
- `iw dev wlan0 get power_save` – get power save mode.
- `iw dev wlan0 set type ibss`: Changes the wireless interface mode to be ibss (Ad-Hoc).
- `iw dev wlan0 set type mesh`: Changes the wireless interface mode to be mesh mode.
- `iw dev wlan0 set type monitor`: Changes the wireless interface mode to be monitor mode.
- `iw dev wlan0 set type managed`: Changes the wireless interface mode to be managed mode.

See `man iw`.
Gitweb: http://git.kernel.org/cgit/linux/kernel/git/jberg/iw.git.
Website: http://wireless.kernel.org/en/users/Documentation/iw.

iwconfig

The old tool for administering wireless devices. The `iwconfig` belongs to the `wireless-tools` package and is based on IOCTLs. Website: www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html.

libreswan Project

An IPsec software solution which forked from openswan version 2.6.38. Website: http://libreswan.org/.

l2ping

A command-line utility for sending L2CAP echo requests and receiving answers over a Bluetooth device. The l2ping utility belongs to the bluez package. Website: www.bluez.org/.

lowpan-tools

A set of utilities to manage the Linux LoWPAN stack. Website: http://sourceforge.net/projects/linux-zigbee/files/linux-zigbee-sources/0.3/.

lshw

A utility that displays information about the hardware configuration of the machine. Website: http://ezix.org/project/wiki/HardwareLiSter.

lscpu

A utility for displaying information about the CPUs on the system. It is based on information from /proc/cpuinfo and sysfs. The lscpu belongs to the util-linux package.

lspci

A utility for displaying information about PCI buses in the system and devices connected to them. Sometimes you need to get some information about a PCI network device with the lspci command. The lspci utility belongs to the pciutils package. Website: http://mj.ucw.cz/sw/pciutils/.

mrouted

A multicast routing daemon, implementing the IPv4 Distance Vector Multicast Routing Protocol (DVMRP), which is specified in RFC 1075 from 1988. Website: http://troglobit.com/mrouted.html.

nc

A command-line utility that reads and writes data across networks. The nc belongs to the nmap-ncat package. Website: http://nmap.org/.

ngrep

A command-line tool, based on the well-known `grep` command, that allows you to specify extended expressions to match against data payloads of packets. It recognizes TCP, UDP, and ICMP across Ethernet, PPP, SLIP, FDDI, and null interfaces. Website: http://ngrep.sourceforge.net/.

netperf

Netperf is a networking benchmarking tool. Website: www.netperf.org/netperf/.

netsniff-ng

`netsniff-ng` is an open source project networking toolkit that, among other things, can help in analyzing network traffic, performing stress tests, generating packets at a very high speed, and more. It uses the PF_PACKET zero-copy RINGs (TX and RX). Among the tools it provides are the following:

- `netsniff-ng` is a fast zero-copy analyzer, `pcap` capturing and replaying tool. The `netsniff-ng` tool is Linux-specific and does not support other operating systems, unlike many of the tools mentioned in this appendix. Example: Running `netsniff-ng --in eth1 --out dump.pcap -s -b 0` creates a `pcap` file that can be read by `wireshark` or by `tcpdump`. The `-s` flag is for silence, and the `-b 0` is for binding to CPU 0. See `man netsniff-ng`.
- `trafgen` is a zero-copy high performance network packet traffic generator utility.
- `ifpps` is a small utility that periodically provides top-like networking and system statistics from the kernel. `ifpps` gathers its data directly from `procfs` files.
- `bpfc` is a small Berkeley Packet Filter assembler and compiler.

Fetching the `git` repository: `git clone git://github.com/borkmann/netsniff-ng.git`.
Website: http://netsniff-ng.org/.

netstat

The `netstat` tool enables you to print multicast memberships, routing tables, network connections, interface statistics, state of sockets, and more. The `netstat` tool belongs to the `net-tools` package. Useful flags:

- `netstat -s`: Displays summary statistics for each protocol.
- `netstat -g`: Displays multicast group membership information for IPv4 and IPv6.
- `netstat -r`: Shows the kernel IP routing table.
- `netstat -nl`: Shows the listening sockets (the `-n` flag is for showing numerical addresses instead of trying to determine symbolic host, port, or user names).
- `netstat -aw`: Shows all raw sockets.
- `netstat -ax`: Shows all Unix sockets.
- `netstat -at`: Shows all TCP sockets.
- `netstat -au`: Shows all UDP sockets.

Website: http://net-tools.sourceforge.net.

nmap (Network Mapper)

Nmap is an open source security project that provides a network exploration and probing tool and a security/port scanner. It has features like port scanning (detecting the open ports on target hosts), OS detection, detecting MAC addresses, and more. For example,

```
nmap www.google.com
```

can give output such as:

```
Starting Nmap 6.00 (http://nmap.org ) at 2013-09-26 16:37 IDT
Nmap scan report for www.google.com (212.179.154.227)
Host is up (0.013s latency).
Other addresses for www.google.com (not scanned): 212.179.154.221 212.179.154.251 212.179.154.232
212.179.154.237 212.179.154.216 212.179.154.231 212.179.154.241 212.179.154.247 212.179.154.222
212.179.154.226 212.179.154.236 212.179.154.246 212.179.154.212 212.179.154.217 212.179.154.242
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
Nmap done: 1 IP address (1 host up) scanned in 5.24 seconds
```

The `nping` utility of `nmap` can be used to generate raw packets for ARP poisoning, networking stress tests, and Denial of Service attacks, as well as to test connectivity like the ordinary `ping` utility. You can use the `nping` utility for setting IP options in generated traffic. See `http://nmap.org/book/nping-man-ip-options.html`. Website: `http://nmap.org/`.

openswan

An open source project implementing an IPsec-based VPN solution. It is based on the FreeS/WAN project. Website: `www.openswan.org/projects/openswan`.

OpenVPN

An open source project implementing VPN based on SSL/TLS. Website: `www.openvpn.net/`.

packeth

An Ethernet-based packet generator tool for Ethernet. The tool has both GUI and CLI. Website: `http://packeth.sourceforge.net/packeth/Home.html`.

ping

The well-known utility for testing connectivity by sending ICMP ECHO request messages. Here are four useful options that are also mentioned in this book:

- `-Q tos`: Enables setting Quality Of Service bits in an ICMP packet. Mentioned in this appendix in the explanation about `tshark` filters.
- `-R`: Sets the Record Route IP option (discussed in Chapter 4).

- `-T`: Sets the timestamp IP option (discussed in Chapter 4).
- `-f`: Flood ping.
- See `man ping` for more command-line options.

The `ping` utility belongs to the `iputils` package. Website: .`www.skbuff.net/iputils/`.

pimd

An open source lightweight stand-alone Protocol Independent Multicast - Sparse Mode (PIM-SM) v2 multicast daemon. Maintained by Joachim Nilsson. See `http://troglobit.com/pimd.html`. `git` repository: `https://github.com/troglobit/pimd/`.

poptop

PPTP Server for Linux. Website: `http://poptop.sourceforge.net/dox/`.

ppp

An open source PPP daemon. `git` repository: `git://ozlabs.org/~paulus/ppp.git`. Website: `http://ppp.samba.org/download.html`.

pktgen

The `pktgen` kernel module (`net/core/pktgen.c`) can generate packets at very high speed. Monitoring and controlling is done via writing to `/proc/net/pktgen` entries. For "HOWTO for the linux packet generator" see `Documentation/networking/pktgen.txt`.

radvd

This is a Router Advertisement Daemon for IPv6. It is an open source project maintained by Reuben Hawkins. It can be used for IPv6 stateless autoconfiguration and for renumbering. Website: `www.litech.org/radvd/`. `git` repository: `https://github.com/reubenhwk/radvd`.

route

A command-line tool for routing tables management. It belongs to the `net-tools` package, which is based on IOCTLs and which is older than the `iproute2` package. Examples:

- `route -n`: Shows the routing table without name resolving.
- `route add default gateway 192.168.1.1`: Adds 192.168.1.1 as a default gateway.
- `route -C`: Displays the routing cache (keep in mind that the IPv4 routing cache was removed in kernel 3.6; see the "IPv4 Routing Cache" section in chapter 5).

See `man route`.

RP-PPPoE

An open source PPP over Ethernet (PPPoE) client for Linux and Solaris systems. Website: www.roaringpenguin.com/products/pppoe.

sar

A command-line tool to collect and report statistics about system activity. It is part of the `sysstat` package. As an example, running the following command will display four times the CPU statistics with interval of 1 second and the average at the end:

```
sar 1 4
Linux 3.6.10-4.fc18.x86_64 (a)  10/22/2013      _x86_64_        (2 CPU)

07:47:10 PM     CPU     %user     %nice   %system   %iowait    %steal     %idle
07:47:11 PM     all      0.00      0.00      0.00      0.00      0.00    100.00
07:47:12 PM     all      0.00      0.00      0.00      0.00      0.00    100.00
07:47:13 PM     all      0.00      0.00      0.00      0.00      0.00    100.00
07:47:14 PM     all      0.00      0.00      0.50      0.00      0.00     99.50
Average:        all      0.00      0.00      0.13      0.00      0.00     99.87
```

Website: http://sebastien.godard.pagesperso-orange.fr/.

smcroute

A command-line tool for multicast routing manipulation. Website: www.cschill.de/smcroute/.

snort

An open source project that provides a network intrusion detection system (IDS) and a network intrusion prevention system (IPS). Website: www.snort.org/.

suricata

An open source project that provides an IDS/IPS and a network security monitoring engine. Website: http://suricata-ids.org/.

strongSwan

An open source project that implements IPsec solutions for Linux, Android, and other operating systems. Both IKEv1 and IKEv2 are implemented. The maintainer is Professor Andreas Steffen. Website: www.strongswan.org/.

sysctl

The `sysctl` utility displays kernel parameters (including network parameters) at runtime. It can also set kernel parameters. For example, `sysctl -a` shows all kernel parameters. The `sysctl` utility belongs to the `procps-ng` package.

taskset

A command-line utility for setting or retrieving a process's CPU affinity. The `taskset` utility is from the `util-linux` package.

tcpdump

`Tcpdump` is an open source command-line protocol analyzer, available from `www.tcpdump.org`. It is based on a C/C++ network traffic capture library called `libpcap`. Like `wireshark`, it can write its results to a file and read them from a file and it supports filtering. Unlike `wireshark`, it does not have a front end GUI. However, its output files can be read by `wireshark`. Example of sniffing with `tcpdump`:

```
tcpdump -i eth1
```

Website: `www.tcpdump.org`.

top

The `top` utility provides a real-time view of the system (parameters like memory usage, CPU usage, and more) and a system summary. This utility is part of the `procps-ng` package. Website: `https://gitorious.org/procps`.

tracepath

The `tracepath` command traces a path to a destination address, discovering the MTU along this path. For IPv6 destination addresses, you can use `tracepath6`. The `tracepath` utility belongs to the `iputils` package. Website: `www.skbuff.net/iputils/`.

traceroute

Print the path that packets traverse to some destination. The `traceroute` utility uses the IP protocol's Time To Live (TTL) field to cause hosts on the packet path to return an ICMP TIME EXCEEDED response. The `traceroute` utility is discussed in Chapter 3, which deals with the ICMP protocol. Website: `http://traceroute.sourceforge.net`.

tshark

The `tshark` utility provides a command-line packet analyzer. It is part of the `wireshark` package. It has many command-line options. For example, you can write the output to a file with the `-w` option. You can set various filters to the packet filtering with `tshark`, some of which can be complex filters (as you will soon see). Example of setting a filter for capturing only ICMPv4 packets:

```
tshark -R icmp
Capturing on eth1
17.609101 192.168.2.200 -> 81.218.16.241 ICMP 98 Echo (ping) request  id=0x0dc6, seq=1/256, ttl=64
17.617101 81.218.16.241 -> 192.168.2.200 ICMP 98 Echo (ping) reply    id=0x0dc6, seq=1/256, ttl=58
```

You can also set a filter on a value of a field in the IPv4 header. For example, the following command sets a filter on the DS field in the IPv4 header:

```
tshark -R "ip.dsfield==0x2"
```

If from a second terminal you send traffic with DS field as 0x2 in the IPv4 header (such traffic can be sent, for example, with `ping -Q 0x2 destinationAdderss`), it will be displayed onscreen by `tshark`.

Example for filtering by source MAC address:

```
tshark ether src host 00:e0:4c:11:22:33
```

Example for filtering for UDP packets whose ports are in the port range 6000–8000:

```
tshark -R udp portrange 6000-8000
```

Example for setting a filter for capturing traffic where the source IP address is 192.168.2.200 and the port is 80 (it does not have to be TCP traffic only because here there is no filter set on some specified protocol):

```
tshark -i eth1 -f "src host 192.168.2.200 and port 80"
```

tunctl

`tunctl` is an older tool for creating TUN/TAP devices. It is available from http://tunctl.sourceforge.net. Note that you can also create or remove a TUN/TAP device with the `ip` command (see the `iproute2` section earlier in this appendix) and with the `openvpn` command-line tool of the `openvpn` package:

```
openvpn --mktun --dev tun1
openvpn --rmtun --dev tun1
```

udevadm

You can get the network device type by running `udevadm` on its `sysfs` entry. For example, if the device has this entry under `sysfs`:

```
/sys/devices/virtual/net/eth1.100
```

then you can find that its DEVTYPE is VLAN:

```
udevadm info -q all -p  /sys/devices/virtual/net/eth1.100/

P: /devices/virtual/net/eth1.100
E: COMMENT=net device ()
E: DEVPATH=/devices/virtual/net/eth1.100
E: DEVTYPE=vlan
E: IFINDEX=4
E: INTERFACE=eth1.100
E: MATCHADDR=00:e0:4c:53:44:58
E: MATCHDEVID=0x0
```

```
E: MATCHIFTYPE=1
E: SUBSYSTEM=net
E: UDEV_LOG=3
E: USEC_INITIALIZED=28392625695
```

udevadm belongs to the udev package. Website: www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html.

unshare

The unshare utility enables you to create a namespace and run a program within that namespace that is unshared from its parent. The unsare utility belongs to the util-linux package. For various command-line options of the unshare utility, see man unshare, Example of usage:

```
unshare -u /bin/bash
```

This will create a UTS namespace.

```
unshare --net /bin/bash
```

This will create a new network namespace, in which a bash process will be started. Gitweb: http://git.kernel.org/cgit/utils/util-linux/util-linux.git. Website: http://userweb.kernel.org/~kzak/util-linux/.

vconfig

The vconfig utility enables you to configure VLAN (802.1q) interface. Examples of usage:

- vconfig add eth2 100: Adds a VLAN interface. This will create a VLAN interface, eth2.100.
- vconfig rem eth2.100: Remove the eth2.100 VLAN interface.
- Note that you can also add and delete VLAN interfaces with the ip command, for example, like this:
  - ip link add link eth0 name eth0.100 type vlan id 100
- vconfig set_egress_map eth2.100 0 4: Map SKB priority of 0 to VLAN priority 4, so that outgoing packets which their SKB priority is 0 will be tagged with 4 as VLAN priority. The default VLAN priority is 0.
- vconfig set_ingress_map eth2.100 1 5: Map VLAN priority 5 to SKB priority of 1, so that incoming packets with VLAN priority of 5 will be queued with SKB priority of 1. The default SKB priority is 0.

See man vconfig.

Note that if VLAN support is compiled as a kernel module, then you must load the VLAN kernel module before trying to add the VLAN interface, by modprobe 8021q. Website: www.candelatech.com/~greear/vlan.html.

wpa_supplicant

Open source software that provides a wireless supplicant for Linux and other OSs. It supports WPA and WPA2. Website: http://hostap.epitest.fi/wpa_supplicant/.

wireshark

The `wireshark` project provides a free and open source analyzer ("sniffer"). It has two flavors: a front-end GTK+ based GUI and a command-line, the `tshark` utility (mentioned earlier in this appendix). It is available on many operating systems and evolves dynamically: when new features are added to existing protocols and new protocols are added, new parsers ("dissectors") are modified or added. Wireshark has many features:

- Enables defining a wide range of filters (ports, destination or source address, protocol identifier, fields in headers, and more).
- Enables sorting the result according to various parameters (protocol type, time, and so on).
- Saves the sniffer output to a file/read a sniffer output from a file.
- Reads/writes many different capture file formats: `tcpdump` (`libpcap`), Pcap NG, and more.
- Capture Filters and Display Filters.

Activating the `wireshark` or `thsark` sniffer puts the network interface to be in promiscuous mode to enable it to handle packets that are not destined to the local host. A lot of information is available in the man pages: `man wireshark` and `man tshark`. You can find more than 75 sniff samples of different protocols in `http://wiki.wireshark.org/SampleCaptures`. Wireshark users mailing list: `www.wireshark.org/mailman/listinfo/wireshark-users`. Website: `www.wireshark.org`. Wiki: `http://wiki.wireshark.org/`.

XORP

An Open Source project, implementing various routing protocols, like BGP, IGMP, OLSR, OSPF, PIM, and RIP. The name XORP is derived from eXtensible Open Router Platform. Website: `www.xorp.org/`.