Network management, measurements and analysis

Michalis Katsarakis  
katsarakis@csd.uoc.gr

Tutorial: HY-439  
http://www.csd.uoc.gr/~hy439/

02 December 2013
Outline

• **Network management**
  – The SNMP protocol
  – `snmpget`, `snmpwalk`, polling an Access Point

• **Wireless network interface commands**
  – `ifconfig`, `iwconfig`, `iwlist`
  – `airmon-ng`
  – `Tcpdump`
  – Wireshark
Network Management

Network management includes the deployment, integration and coordination of the hardware, software and human elements to monitor, test, poll, configure, analyze, evaluate and control the network and element resources to meet the real-time, operational performance and Quality of Service requirements at a reasonable cost.

T. Saydam and T. Magedanz
Network Management

Benefits from network management

– **Detect failure** of an interface card at a host
– **Monitor traffic** to aid in resource deployment
– **Detect rapid changes** in routing tables
– **Intrusion Detection**
Architecture of a network management system

- **Managing entity**: the central “area” of activity. Controls the collection, processing, analysis and display of network management information.

- **Managed device**: a piece of network equipment that resides on a managed network

- Network management **protocol**: The protocol that runs between the managing entity and the managed devices.
  - In our case SNMP
Architecture of a network management system
Simple Network Management Protocol

- Simple Network Management Protocol (SNMP) is an Internet-standard protocol for managing devices on IP networks.

- Each managed system executes, at all times, a software component called an agent which reports information via SNMP to the manager.

- SNMP agents expose management data on the managed systems as variables.
Simple Network Management Protocol

Management Information Base (MIB)

- Information “database” holding managed objects whose values collectively reflect the current “state” of the network.

A MIB Object might be:

- The number of IP datagrams discarded at the router.
- The number of carrier sense errors in an Ethernet Interface.
- Descriptive information such as the server software running on a DNS server.
- Protocol specific information.
- Information whether a particular device is functioning correctly or not.
**Simple Network Management Protocol**

Management Information Base (MIB)

![OID Tree Example](image)
Simple Network Management Protocol

7 types of SNMP Messages known as Protocol Data Units (PDU):

1. GetRequest
   A manager-to-agent request to retrieve the value of a variable.

2. SetRequest
   A manager-to-agent request to change the value of a variable.

3. GetNextRequest
   A manager-to-agent request to discover available variables and their values

4. GetBulkRequest
   A manager-to-agent request for multiple iterations of GetNextRequest.

5. Response
   Returns variable bindings and acknowledgement from manager-to-agent requests mentioned above.

6. Trap
   Asynchronous notification from agent-to-manager.

7. InformRequest
   Acknowledged asynchronous notification.
Simple **Network Management** Protocol

snmpget:
  communicates with a network entity
  using SNMP GET requests.

snmpwalk:
  Retrieves a subtree of management values
  using SNMP GETNEXT requests.

Retrieve the interfaces sub-tree:
  
  
  # snmpwalk -v1 -c public [IP addr] 1.3.6.1.2.1.2

MIB discovery:
  http://cric.grenoble.cnrs.fr/Administrateurs/Outils/MIBS/?oid=1.3.6.1.2.1.2
while (true) {
    snmpget -v1 -c public [IP addr] 1.3.6.1.2.1.2.2.1.10.5 // ifInOctets, 32 bit unsigned int
    snmpget -v1 -c public [IP addr] 1.3.6.1.2.1.2.2.1.16.5 // ifOutOctets, 32 bit unsigned int
    sleep 1 minute
}
Outline

• Network management
  – The SNMP protocol
  – snmpget, snmpwalk, polling an Access Point

• **Wireless network interface commands**
  – ifconfig, iwconfig, iwlist
  – airmon-ng
  – tcpdump
  – Wireshark
A network interface controller (NIC) (also known as a network interface card, network adapter, LAN adapter and by similar terms) is a computer hardware component that connects a computer to a computer network.
interface configuration

Name
ifconfig - configure a network interface

Synopsis
ifconfig [interface]
ifconfig interface [aftype] options | address ...

Description
Ifconfig is used to configure the kernel-resident network interfaces. It is used at boot time to set up interfaces as necessary. After that, it is usually only needed when debugging or when system tuning is needed.
View Network Settings of an Ethernet Adapter:

katsarakis@jagermeister:~ # ifconfig eth0

eth0   Link encap:Ethernet  HWaddr 00:c0:9f:98:27:05
inet addr:139.91.182.207  Bcast:139.91.182.255  Mask:255.255.255.0
inet6 addr: fe80::2c0:9fff:fe98:2705/64 Scope:Link
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:414053 errors:0 dropped:9 overruns:0 frame:0
TX packets:78523 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:246815934 (246.8 MB)  TX bytes:47642161 (47.6 MB)
Interrupt:16
interface configuration

Display Details of All interfaces Including Disabled Interfaces
   # ifconfig –a

Disable an Interface
   # ifconfig eth0 down

Enable an Interface
   # ifconfig eth0 up

Assign ip-address, netmask and broadcast at the same time to interface eth0.
   # ifconfig eth0 192.168.2.2 netmask 255.255.255.0 broadcast 192.168.2.255
interface wireless configuration

NAME
iwconfig - configure a wireless network interface

SYNOPSIS
iwconfig [interface]
iwconfig interface [essid X] [nwid N] [mode M] [freq F] [channel C][sens S ]
[ap A ][nick NN ] [rate R] [rts RT] [frag FT] [txpower T]
[enc E] [key K] [power P] [retry R]
[commit] iwconfig --help iwconfig –version

DESCRIPTION
iwconfig is similar to ifconfig, but is dedicated to the wireless interfaces. It is used to set the parameters of the network interface which are specific to the wireless operation (for example: the frequency).
Register to the network with Extended Service Set Identification (ESSID) “eduroam”
   # iwconfig wlan0 essid “eduroam”

Associate with the Access Point (AP) with MAC address D4:D7:48:B0:87:C1
   # iwconfig wlan0 ap D4:D7:48:B0:87:C1

Set the operating frequency or channel in the interface.
   # iwconfig wlan0 freq 2422000000
   *A value below 1000 indicates a channel number, a value greater than 1000 is a frequency in Hz.*

Set the transmit power to 15 dBm
   # iwconfig wlan0 txpower 15
   *For cards that support multiple transmit powers.*
**Interface wireless list**

**NAME**

iwlist - Get more detailed wireless information from a wireless interface

**SYNOPSIS**

iwlist interface scanning
iwlist interface rate
iwlist interface power
iwlist interface retry
iwlist –help

iwlist interface frequency
iwlist interface key
iwlist interface txpower
iwlist interface event
iwlist –version

**DESCRIPTION**

Iwlist is used to display some additional information from a wireless network interface that is not displayed by iwconfig.
Trigger a scan: the interface will switch channels and listen for beacon frames
   # iwlist wlan0 scanning

wlan0   Scan completed :
Cell 01 - Address: 00:11:93:03:1E:32
   Channel:1
   Frequency:2.412 GHz (Channel 1)
   Quality=60/70  Signal level=-50 dBm
   Encryption key:off
   ESSID:"forth public access"
   Bit Rates: 1 Mb/s; 2 Mb/s; 5.5 Mb/s; 6 Mb/s; 9 Mb/s; 11 Mb/s; 12 Mb/s; 18 Mb/s
           24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
   Mode:Master

Cell 02 - Address: 00:11:93:03:1E:33
...
Select only the lines with Address or Signal level

# iwlist wlan0 scanning | grep 'Address\|Signal'

Cell 01 - Address: 00:11:93:03:1E:32
  Quality=51/70  Signal level=-59 dBm

Cell 02 - Address: 00:11:93:03:1E:31
  Quality=50/70  Signal level=-60 dBm

Cell 03 - Address: 00:11:93:03:1E:30
  Quality=49/70  Signal level=-61 dBm

Cell 04 - Address: 00:11:93:03:1E:34
  Quality=51/70  Signal level=-59 dBm

...
Name
airmon-ng - a bash script designed to turn wireless cards into monitor mode

Synopsis
airmon-ng <start|stop> <interface> [channel]

Description
airmon-ng is a bash script designed to turn wireless cards into monitor mode. It autodetects which card you have and run the right commands.
NAME
tcpdump - dump traffic on a network

SYNOPSIS
tcpdump [ -AdDefILnNOpqRStuUvxX ] [ -c count ] [ -C file_size ] [ -F file ]
[ -i interface ] [ -m module ] [ -M secret ] [ -r file ] [ -s snaplen ]
[ -T type ] [ -w file ] [ -W filecount ] [ -E spi@ipaddr algo:secret,... ]
[ -y datalinktype ] [ -Z user ] [ expression ]

DESCRIPTION
Tcpdump prints out the headers of packets on a network interface that match the
boolean expression. It can also be run with the -w flag, which causes it to save the
packet data to a file for later analysis.
“The quieter you become, the more you are able to hear...”

Usually a Wi-Fi interface will, even in promiscuous mode, only capture the traffic on the BSS to which it is associated. In order to capture all traffic that the interface can receive and see the IEEE 802.11 headers, we must turn the interface in monitor mode.

Turn the interface `wlan0` in monitor mode and call it `mon0`.

```bash
# airmon-ng start wlan0
```

Capture, in **promiscuous mode**, all traffic received by `mon0`, and write it in files with

- max size $10^6$ Bytes
- file names: wifi-capture1.pcap, wifi-capture2.pcap, ...

```bash
# tcpdump -i mon0 -C 10 -w wifi-capture.pcap
```
Wireshark

Description

**Wireshark** is a GUI network protocol analyzer. It lets you interactively browse packet data from a live network or from a previously saved capture file. **Wireshark**'s native capture file format is **libpcap** format, which is also the format used by **tcpdump** and various other tools.

With Wireshark we can open a .pcap file created with tcpdump, apply filters to inspect specific packets and export network traffic statistics.

Some 802.11-related Wireshark filters can be found here:
Wireshark

- General traffic statistics
  - Traffic volume
  - Burstiness
  - Traffic volume by types
- End to end statistics
  - Connection throughput
  - Round trip delay
  - Loss rate

Traffic categorization
Wireshark

- General traffic statistics
  - Traffic volume
  - Burstiness
  - Traffic volume by types
- End to end statistics
  - Connection throughput
  - Round trip delay
  - Loss rate

*asymmetric packet sizes*, i.e., APs with large sent and small receive packets, and APs with small sent and large receive packets
Wireshark: beacon frame

Wireless interface commands
Wireshark: association request

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>7665</td>
<td>01:14:52</td>
<td>SonyEric_17:dd:07</td>
<td>Cisco_03:1e:33</td>
<td>802.11</td>
<td>91</td>
<td>Association Request, SN=1, FN=0, Flags=........C, SSID=netlab-QoS-test</td>
</tr>
<tr>
<td>7667</td>
<td>01:14:53</td>
<td>Cisco_03:1e:33</td>
<td>SonyEric_17:dd:07</td>
<td>802.11</td>
<td>76</td>
<td>Association Response, SN=1361, FN=0, Flags=........C</td>
</tr>
</tbody>
</table>

Header revision: 0
Header pad: 0
Header length: 26

Present flags
MAC timestamp: 3224955212815
Flags: 0x10
Data Rate: 1.0 Mb/s
Channel frequency: 2412 [BG 1]
Channel type: 802.11b (802a0)
SSID Signal: -48 dBm
Antenna: 2
RX flags: 0x0080

IEEE 802.11 Association Request, Flags: .......C
Type/Subtype: Association Request (0x00)
Frame Control: 0x0000 (Normal)
Duration: 314
Destination address: Cisco_03:1e:33 (00:11:93:03:1e:33)
BSSID: Cisco_03:1e:33 (00:11:93:03:1e:33)
Fragment number: 0
Sequence number: 1
Wireshark: RTS

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>28733</td>
<td>219.101450</td>
<td>Cisco_51:e6:02 (TA)</td>
<td>Apple_03:ce:1a (RA)</td>
<td>802.11</td>
<td>48</td>
<td>Request-to-send, Flags=........C</td>
</tr>
<tr>
<td>26272</td>
<td>219.588536</td>
<td>Cisco_51:e6:02 (TA)</td>
<td>Apple_03:ce:1a (RA)</td>
<td>802.11</td>
<td>48</td>
<td>Request-to-send, Flags=........C</td>
</tr>
<tr>
<td>39365</td>
<td>390.167453</td>
<td>Cisco_51:e6:02 (TA)</td>
<td>Apple_03:ce:1a (RA)</td>
<td>802.11</td>
<td>48</td>
<td>Request-to-send, Flags=........C</td>
</tr>
<tr>
<td>42771</td>
<td>418.497385</td>
<td>Cisco_03:1e:32 (TA)</td>
<td>SonyEric_17:dd:07 (RA)</td>
<td>802.11</td>
<td>48</td>
<td>Request-to-send, Flags=........C</td>
</tr>
<tr>
<td>42840</td>
<td>418.538361</td>
<td>Cisco_03:1e:32 (TA)</td>
<td>SonyEric_17:dd:07 (RA)</td>
<td>802.11</td>
<td>48</td>
<td>Request-to-send, Flags=........C</td>
</tr>
<tr>
<td>60347</td>
<td>572.244064</td>
<td>Cisco_51:e6:02 (TA)</td>
<td>Apple_03:ce:1a (RA)</td>
<td>802.11</td>
<td>48</td>
<td>Request-to-send, Flags=........C</td>
</tr>
</tbody>
</table>

Wireshark 1.6.7

**Filter:** wlan.fc.type_subtype==27

---

**Frame 42771:** 46 bytes on wire (368 bits), 46 bytes captured (368 bits)

**Radiotap Header v0, Length 26**
- Header revision: 0
- Header pad: 0
- Header length: 26
- Present flags
- MAC timestamp: 3225296519531
- Flags: 0x10
- Data Rate: 12.0 Mb/s
- Channel frequency: 2412 [86]
- Channel type: 802.11g (pure-g) (0x80c)
- SSI Signal: -48 dBm
- Antenna: 1
- RX flags: 0x0060

**IEEE 802.11 Request-to-send, Flags: ........C**
- Type/Subtype: Request-to-send (0x1b)
- Frame Control: 0x00b4 (Normal)
- Duration: 816
- Transmitter address: Cisco_03:1e:32 (00:11:93:03:1e:32)
- Frame check sequence: 0x00c1a031 [correct]

```
0600 00 00 10 00 2f 48 00 00 6b 41 a7 f2 ee 02 00 00 .../H: kA......
0610 10 18 6c 00 c0 00 dd 01 00 00 00 30 30 03 8c 64 .l......l.0...d
0620 22 17 dd 07 00 11 93 03 1e 32 0c 01 00 a0 31        ".......
```

- Type and subtype combined (first ...)
- Packets: 123196 Displayed: 39 Marked: 0 Dropped: 22

---

**Profile:** Default
Wireshark: 802.11 ACK
Wireshark: guides for Project 1

1. Apply the appropriate Wireshark filter

Filter: tcp && http
Wireshark: guides for Project 1

2. Save captured data in text file
   - File --> Export (Packet Dissections) --> As “Plain Text” file
   - Packet Range: Select “All packets” & “Displayed”
   - Packet Format: Check only “packet summary line”.

![Image of Wireshark interface showing save as plain text options]
2. Save captured data in text file

- File --> Export (Packet Dissections) --> As “Plain Text” file
- Packet Range: Select “All packets” & “Displayed”
- Packet Format: Check only “packet summary line”.

![Image of Wireshark interface with captured data]
3. Load saved text file in matlab

```matlab
fid = fopen('capture1.txt');
C = textscan(fid, '%d %f %s %s %s %d %*[^
]', 'headerLines', 1);
fclose(fid);
```

Means: Ignore everything until “\n”.

Wireless interface commands