

MRTG on Raspberry Pi

Christian Külker

2022-06-06

Contents

1	Introduction	2
2	Dependencies	2
2.1	Web Server	2
2.2	SNMP Server	4
3	Installation of MRTG	5
4	MRTG Configuration	5
4.1	Adding Entries/ Graphs	5
4.2	Example Configuration For CPU Idle Time	9
4.3	Configuring MRTG Web output	9
5	Updating The Data For The Web Page	10
6	Access The New Installed MRTG	10
7	Testing And Debugging	10
7.1	SNMP And SNMPD	10
7.2	Run MRTG Manually	11
7.3	Running MRTG With Different Log Information	12
7.4	Debugging SNMP	12
8	Packages	12
9	Known Problems and Caveats	12
10	Critique	13
11	Links	14

12 History	14
13 Disclaimer of Warranty	14
14 Limitation of Liability	14

1 Introduction

This article is about a software called “Multi Router Traffic Grapher” (MRTG) that can monitor the load of network interfaces and other values on one or more computers. While this article tries to be as explicative as possible due to the nature of the subject it is targeting experienced Linux users or system administrators.

This article describes a simple installation on one server. In fact a small Raspberry Pi router. It should be very similar with any Debian based distribution on other hardware too.

One advantage of using [MRTG](#) is that it explicitly summarises the usage of network devices on daily, weekly and monthly basis in terms of **speed**. This is useful, if you are connected with a limited internet connection and you need to know how fast you are.

2 Dependencies

The [MRTG](#) is usually a multi server installation. One server can be a web server that collects and displays aggregated log data. Other servers or switches can be queried via [SNMP](#). For this article everything is installed on one machine.

2.1 Web Server

One dependency for [MRTG](#) is the web server. That can basically any web server. Out of curiosity I tested **Nginx** (speak: engine x) and it works. However in the past, when [MRTG](#) was created, Apache was more common. For this reason many guides on the web have some tricks when it comes with the usage of Apache. You might consider [Apache2](#), if you need certain features.

2.1.1 Configuring of Nginx for MRTG

The installation is straight forward:

```
aptitude update
aptitude install nginx
```

Copy the following into the file `/etc/nginx/sites-available/mrtg`

```
1 server {
2     listen 8080 default_server;
3     listen [::]:8080 default_server;
4
5     root /var/www/mrtg;
6     index index.html;
7     server_name _;
8     location /mrtg/ {
9         try_files $uri $uri/ =404;
10    }
11 }
```

Then make a link and restart **Nginx**

```
cd /etc/nginx/sites-enabled
ln -s /etc/nginx/sites-available/mrtg .
service nginx restart
```

2.1.2 Configuration of Apache2 for MRTG

The configuration can be done in many ways. [MRTG](#) can be configured as main host, virtual host with and without SSL. The following configuration is using the Debian default `site configuration` and do not set up [MRTG](#) as a site, but as an `configuration snippet` for [Apache2](#).

Copy the following content into the file `/etc/apache2/conf-available/mrtg.conf`. The values for `AllowOverride` and `Options` need to be made specific for the server. This is just an example.

```
1 <IfModule mod_alias.c>
2     Alias /mrtg /var/www/mrtg
3 </IfModule>
4
5 <Directory /var/www/mrtg/>
6     Options SymLinksIfOwnerMatch
7     AllowOverride None
8 </Directory>
9
10 ErrorLog /var/log/apache2/mrtg-error.log
11 CustomLog /var/log/apache2/mrtg-access.log combined
```

This kind of configuration can be enabled by `a2enconf mrtg`.

2.2 SNMP Server

Install the [SNMP](#) server and tools.

```
aptitude install snmp snmpd
```

Check if the server is running with one (!) of the following commands.

```
ps ax|grep snmp|grep -v grep
service snmpd status
```

The first command should give something like:

```
1 29529 ?          S      0:00 /usr/sbin/snmpd -Lsd -Lf /dev/null -u snmp \
2 -g snmp -I -smux mteTrigger mteTriggerConf -p /run/snmpd.pid
```

Test the server. The following command should give some output

```
snmpwalk -v1 -cpublic localhost

iso.3.6.1.2.1.1.1.0 = STRING: "Linux pi 4.4.9-v7+ #884 SMP Fri \
May 6 17:28:59 BST 2016 armv7l"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.8072.3.2.10
iso.3.6.1.2.1.1.3.0 = Timeticks: (38719) 0:06:27.19
iso.3.6.1.2.1.1.4.0 = STRING: "Me <me@example.org>"
iso.3.6.1.2.1.1.5.0 = STRING: "pi"
iso.3.6.1.2.1.1.6.0 = STRING: "Sitting on the Dock of the Bay"
iso.3.6.1.2.1.1.7.0 = INTEGER: 72
...
```

As it can be seen from this output some values in `/etc/snmp/snmpd.conf` are wrong. The e-mail address and the location “Sitting on the Dock of the Bay”. However for a basic function a change it is not needed. After your complete installation works, make a backup of that configuration file and update missing/ wrong entries.

Especially under Debian the problem is a very limiting default configuration that do not give read access to most of the MIB tree. To change this, for example, the following `-V systemonly` has to be removed from `rocommunity public default` for [IPv4](#) in `/etc/snmp/snmpd.conf` and the `snmpd` service has to be restarted. If one uses [IPv6](#) a different line need to be changed.

before:

```
snmpwalk -v 2c -c public localhost |wc -l
47
```

after:

```
snmpwalk -v 2c -c public localhost |wc -l
8936
```

3 Installation of MRTG

The installation of [MRTG](#) is easy:

```
aptitude install mrtg
```

4 MRTG Configuration

The first step is to secure the default configuration. The reason to reuse the name `mrtg.cfg` is, that `crond` or a `systemd` timer is most likely already configured to use this file, so we do not need to modify `crond` or `systemd`. For Debian 10 Buster [MRTG](#) uses cron in `/etc/cron.d/mrtg` and updates the graphs every 5 minutes and write log entries to `/var/log/mrtg/mrtg.log`. You should check the log file for errors in case the configuration is updated.

```
cp -a /etc/mrtg.cfg /etc/mrtg.cfg.`date +%F`
```

The configuration of the web interface is a very crude and archaic process, compared to INI or YAML files. The configuration is done via `/etc/mrtg.cfg` and applied by a command since the `mrtg` command is executed via a scheduler: `crond` or `systemd`. The format of `mrtg.cfg` can be looked up via the [mrtg-reference](#).

4.1 Adding Entries/ Graphs

Most entries are added via [SNMP](#) OIDs to the file `mrtg.cfg`. There fore looking up the MIB and figuring out which value can be reached via `snmpget` is the first thing to do.

A graph entry can be generated by a set of configuration keywords belonging to one entity. Lets call the entity 'hobbit' for example, than it looks in an abstract way like this:

```
1 KeyWord1[hobbit]: Value1
2 KeyWord2[hobbit]: Value2
3 KeyWord3[hobbit]: Value3
4 KeyWord4[hobbit]: Value4
5 KeyWord5[hobbit]: Value5
```

One of the most important keywords is the **Target** keyword. The **value** of a keyword is called an **argument** in [MRTG](#), probably because a keyword is related to a command. For the **Target** keyword different commands can be executed, which manifests it self in different **argument** classes. So to understand you have to reverse the thinking. The kind of value to a keyword determines with command is executed. The following classes are defined for the keyword **target**:

- Basic
- SNMPv2c
- SNMPv3
- noHC
- Reversing
- Explicid OIDs
- MIB variables
- Snmpwalk
- SnmpGetNext
- Counted SNMP Walk
- Interface by IP
- Interface by Description
- Interface by Name
- Interface by Ethernet Address
- Interface by Type
- Extended positioning of ifIndex

As this is quite extensive, only the **Explicid OIDs** class is used. For this to work **two** OIDs have to be used. The reason is that per default [MRTG](#) plots 2 variables against time. Usually this will be network bytes in and out. So, to plot a single value the same OID has to be used twice.

The format is:

```
1 Target[NAME]: OID_1&OID_2:COMMUNITY@HOST
2
3 NAME: arbitrary word
4 OID_1: numerical SNMP OID
5 OID_2: numerical SNMP OID
6 COMMUNITY: SNMP community, for example 'public'
7 HOST: hostname of machine to query, for example 'localhost'
```

The format can be more flexible. For all places where `COMMUNITY@HOST` can be used the real format of this can be

```
1 COMMUNITY@HOST[:[port]][:[timeout]][:[retries]][:[backoff]][:[version]]][[name]]
```

Also **port** can be more flexible. See [mrtg-reference](#) for examples and explanation.

What has to be understood is that the value of **Target** is interpolated in a very [MRTG](#) specific way and some expressions are extrapolated. So addition, subtraction, division, multiplication, brackets and piping to custom commands works. It is a complete language by itself.

When you find a valid OID, for example `.1.3.6.1.4.1.2021.4.6` for the total amount of free main memory, you sometimes have to add a `0` to make [MRTG](#) and [SNMP](#) happy: `.1.3.6.1.4.1.2021.4.6.0`.

A target for the value would look like this:

```
1 Target[hobbit]: .1.3.6.1.4.1.2021.4.6.0&.1.3.6.1.4.1.2021.4.6.0:
   public@localhost
```

The full example for the measurement of free main memory on a Debian 10 Buster Linux Raspberry Pi 4 with 4 GB main memory looks like this:

Global Configuration:

```
1 WorkDir: /var/www/mrtg
2 WriteExpires: Yes
3 EnableIPv6: no
```

Free Main Memory:

```
1 Title[localhost-free]: Localhost free main memory
2 PageTop[localhost-free]: <H1>Localhost - Memory Free</H1>
3 Target[localhost-free]: .1.3.6.1.4.1.2021.4.6.0&.1.3.6.1.4.1.2021.4.6.0:\
4 public@localhost
5 MaxBytes[localhost-free]: 3918772
6 YLegend[localhost-free]: memory
7 ShortLegend[localhost-free]: Bytes
8 LegendI[localhost-free]: bytes
9 Legend1[localhost-free]: bytes free
10 Options[localhost-free]: integer, gauge, nopercent, growright, unknaszero,
   noo
```

The value for **MaxBytes** can be queried with [SNMP](#): (A working configuration for `snmpd` is assumed here - if this does not give a value, either the `snmpd` configuration need set up correctly or it is a different hardware)

```
snmpget -v 2c localhost -c public .1.3.6.1.4.1.2021.4.5.0
iso.3.6.1.4.1.2021.4.5.0 = INTEGER: 3918772
```

Then the configuration needs to be activated

```
LANG=C indexmaker /etc/mrtg.cfg > /var/www/mrtg/index.html
```

Then the configuration needs to be executed twice, as first run throws errors due to an empty database.

```
LANG=C /usr/bin/mrtg /etc/mrtg.cfg
LANG=C /usr/bin/mrtg /etc/mrtg.cfg
```

4.1.1 Semi Automatic Configuration

A semi automatic configuration for network interfaces is possible with the `cfgmaker` script.

Since one interface, the `wlan0`, do not deliver a speed value, to be able to use it with `MRTG`, it is necessary to set a value with the `-zero-speed=` parameter

```
LANG=C cfgmaker --zero-speed=100000000 public@127.0.0.1 > /etc/mrtg.cfg
```

`MRTG` has a limited capability to scan hardware and create a configuration for it.

```
LANG=C cfgmaker public@127.0.0.1 --ifref=descr --output /etc/mrtg.cfg
```

This basically generates 3 interface graphs for `lo`, `eth0` and `wlan` on the Raspberry Pi 4. Some commented out. The `eth0` section looks like:

```
1  #### Interface 2 >> Descr: 'eth0' | Name: 'eth0' | Ip: '192.168.168.35' | \
2  Eth: 'dc-a6-32-78-c1-d5' ###
3
4  Target[127.0.0.1_2]: 2:public@127.0.0.1:
5  SetEnv[127.0.0.1_2]: MRTG_INT_IP="192.168.168.35" MRTG_INT_DESCR="eth0"
6  MaxBytes[127.0.0.1_2]: 125000000
7  Title[127.0.0.1_2]: Traffic Analysis for 2 -- monitor
8  PageTop[127.0.0.1_2]: <h1>Traffic Analysis for 2 -- monitor</h1>
9      <div id="sysdetails">
10         <table>
11             <tr>
12                 <td>System:</td>
13                 <td>monitor in monitor.c8i.org</td>
14             </tr>
15             <tr>
16                 <td>Maintainer:</td>
17                 <td>c &lt;c@c8i.org&gt;</td>
18             </tr>
19             <tr>
20                 <td>Description:</td>
21                 <td>eth0 </td>
```



```

22         </tr>
23         <tr>
24             <td>ifType:</td>
25             <td>ethernetCsmacd (6)</td>
26         </tr>
27         <tr>
28             <td>ifName:</td>
29             <td>eth0</td>
30         </tr>
31         <tr>
32             <td>Max Speed:</td>
33             <td>125.0 MBytes/s</td>
34         </tr>
35         <tr>
36             <td>Ip:</td>
37             <td>192.168.168.35 (monitor.c8i.org)</td>
38         </tr>
39     </table>
40 </div>

```

Even though the scanning understands that this is the `eth0` interface, the name of the title is just the number '2'. Which might work well for switches, but not for hosts. Clicking on the graph however show the information.

4.2 Example Configuration For CPU Idle Time

This example show the configuration of the CPU idle time for the Raspberry PI 4 and shows how simple mathematical terms are realized inside the target keyword.

```

1 Title[localhost-CPU]: Localhost CPU load
2 PageTop[localhost-CPU]: <H1>Localhost - CPU Idle Time (%)</H1>
3 Target[localhost-CPU]: 100 - .1.3.6.1.4.1.2021.11.11.0&\
4 .1.3.6.1.4.1.2021.11.11.0:public@localhost
5 MaxBytes[localhost-CPU]: 100
6 YLegend[localhost-CPU]: CPU %
7 ShortLegend[localhost-CPU]: %
8 LegendI[localhost-CPU]: CPU
9 Legend1[localhost-CPU]: CPU usage
10 Options[localhost-CPU]: integer, gauge, nopercent, growright, unknaszero,
    noo

```

4.3 Configuring MRTG Web output

One can of course generate an index page by hand, which is probably recommended. A quick and dirty approach is to use the `indexmaker` script. This creates a page with one graph per target and a link to the sub-page of the target.

Localhost - CPU Idle Time (%)

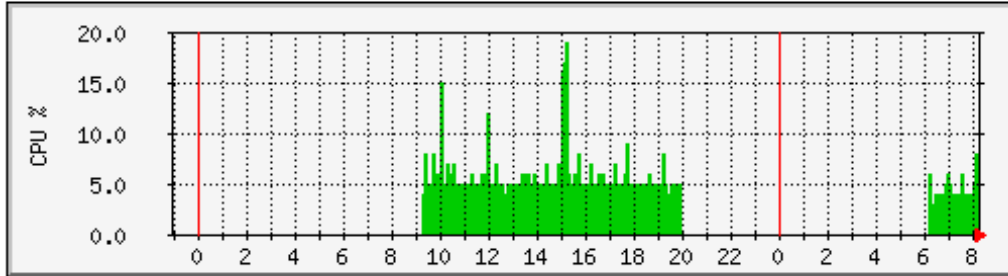


Figure 1: CPU Idle Time

```
mkdir -p /var/www/mrtg
LANG=C indexmaker /etc/mrtg.cfg > /var/www/mrtg/index.html
```

5 Updating The Data For The Web Page

Manually:

```
LANG=C mrtg
```

Of course this is already configured with `cron`.

6 Access The New Installed MRTG

Point your web browser to the IP of your [MRTG](#) machine. `http://<IP-TO-MRTG-HOST>/`
or `http://<IP-TO-MRTG-HOST>/PATH`.

7 Testing And Debugging

7.1 SNMP And SNMPD

For many functionality of [MRTG](#) a working and well configured `snmpd` is mandatory. Version 1 can be tested with:

```
snmpstatus -c public -v1 localhost
```

This would be considered an error for example:

```
1 Error in packet.
```

```
2 Reason: (noSuchName) There is no such variable name in this MIB.
3 Failed object: iso.3.6.1.2.1.4.10.0
4
5 Error in packet.
6 Reason: (noSuchName) There is no such variable name in this MIB.
7 Failed object: iso.3.6.1.2.1.4.3.0
8
9 [UDP: [127.0.0.1]:161->[0.0.0.0]:45109]=>[Linux monitor 5.4.79-v7l+ #1373
   SMP \
10 Mon Nov 23 13:27:40 GMT 2020 armv7l] Up: 0:00:54.02
11 Interfaces: 0, Recv/Trans packets: 0/0 | IP: 0/0
```

Version 2c can be tested with

```
snmpstatus -c public -v2c localhost
```

This would be considered a success:

```
[UDP: [127.0.0.1]:161->[0.0.0.0]:49633]=>[Linux monitor 5.4.79-v7l+ #1373
↳ SMP \
Mon Nov 23 13:27:40 GMT 2020 armv7l] Up: 0:03:16.30
Interfaces: 0, Recv/Trans packets: 0/0 | IP: 0/0
```

To query the kernel on Debian 10 Buster on Raspberry PI 4 do

```
snmpget -v2c -cp public localhost iso.3.6.1.2.1.1.1.0
iso.3.6.1.2.1.1.1.0 = STRING: "Linux monitor 5.4.79-v7l+ #1373 SMP Mon Nov
↳ \
23 13:27:40 GMT 2020 armv7l"
```

There are different approaches to query the memory, one is:

```
# UCD-SNMP-MIB::memTotalReal
snmpget -v2c -cp public localhost .1.3.6.1.4.1.2021.4.5.0
iso.3.6.1.4.1.2021.4.5.0 = INTEGER: 3918772
```

7.2 Run MRTG Manually

The cron job defined and runs [MRTG](#) as follows:

```
if [ -x /usr/bin/mrtg ] && [ -r /etc/mrtg.cfg ] && \
[ -d "$(grep '^[[:space:]]*[^#]*[[:space:]]*WorkDir' /etc/mrtg.cfg | \
awk '{ print $NF }')" ]; then mkdir -p /var/log/mrtg ; \
env LANG=C /usr/bin/mrtg /etc/mrtg.cfg 2>&1 | tee -a
↳ /var/log/mrtg/mrtg.log ; \
```

```
fi
```

This can be used to run [MRTG](#) immediately and there is no need to wait 5 minutes.

7.3 Running MRTG With Different Log Information

```
LANG=C /usr/bin/mrtg -debug cfg /etc/mrtg.cfg
```

7.4 Debugging SNMP

This helps to debug `snmpget` and other [SNMP](#) operations as well as some script debugging.

```
LANG=C /usr/bin/mrtg -debug snpo /etc/mrtg.cfg
```

A script failure might look like

```
1 --snpo: run external sh /etc/mrtg/mrtg_ping 8.8.8.8
2 --snpo: External result:100 out:undef uptime:unknown name:unknown
```

However in this case the script do not return all values, so this is actually intended.

8 Packages

```
1 mrtg - multi router traffic grapher
2 mrtg-contrib - multi router traffic grapher (contributed files)
3 mrtg-ping-probe - Ping module for Multi Router Traffic Grapher
4 mrtg-rrd - Generating graphs for MRTG statistics (CGI)
5 mrtgutils - Utilities to generate statistics for mrtg
6 mrtgutils-sensors - Utilities to generate statistics for mrtg (from lm-sensors)
7 pcp-import-mrtg2pcp - Tool for importing data from MRTG into PCP archive logs
```

9 Known Problems and Caveats

- The standard web page (`index.html`) of [MRTG](#) will not **reload** when [MRTG](#) has updated the graphs manually. The page refresh is set to the time specified inside the configuration, usually 10 minutes, which make sense. However when debugging or adding new configuration [MRTG](#) is usually run manually and the page do not update. To mitigate, press browser reload button.

- **Devices are numbers:** For now the display of the index page enumerates devices, like 2 (`eth0`) and 3 (`wlan0`) that is not very intuitive. I tried to use some options to `cfgmaker` but without success. In case you know the answer I would be happy to include it here.
- **No back links:** If you click on a device graph of the index page you can enter the detailed report about each device on a dedicated page. There are no links back. One can modify the configuration and add a link for every target manually in the `PageTop` attribute.
- Nice about `MRTG` is that it already have a daily, weekly, monthly and yearly overview of network interfaces **speed** (that can be generated automatically). However it do not have a summary on how much KB or MB or GB (or KiB, MiB, GiB) the interface processed over the time (of a day, month, week or year) aka **accumulated traffic**. So if your intention is to keep the usage of your mobile plan under control then `MRTG` will not help.

10 Critique

Pros

- Simple architecture (configuration, cron, log file, web pages)
- Most data accessible via `SNMP` and scripts
- Easy testing of sensors
- Semi automatic setup of interface speed measurements
- Simple data model
- Moderate dependencies
- No obfuscating abstraction layers
- Reliable execution
- Low system resource usage

Cons

- First invocation of `MRTG` always produces errors (the handling of data files and data backup files is not clean)
- No back links in the web interface to the index page
- Mixing of HTML and configuration inside the configuration
- Difficult to configure new graphs from scratch
- The configuration is a complete new language
- 2 value constrains for single value graphs
- Only numeric values
- Data and markup language is not separated and even stored in log files inside the web root (e.g. `/var/www/mrtg/localhost-cpu.log`)

11 Links

- [satsignal](#)
- [mrtg-rbp-project](#)

12 History

Version	Date	Notes
0.1.6	2022-06-06	Change shell to bash
0.1.5	2021-05-18	Updates for Raspberry PI 4
0.1.4	2021-01-02	Updates for Debian 10 Buster
0.1.3	2020-06-06	Formatting for Quick-Guide
0.1.2	2016-06-25	Add caveats section
0.1.1	2016-06-21	Add Nginx configuration
0.1.0	2016-06-20	Initial release

13 Disclaimer of Warranty

THERE IS NO WARRANTY FOR THIS INFORMATION, DOCUMENTS AND PROGRAMS, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE INFORMATION, DOCUMENT OR THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE INFORMATION, DOCUMENTS AND PROGRAMS IS WITH YOU. SHOULD THE INFORMATION, DOCUMENTS OR PROGRAMS PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

14 Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE INFORMATION, DOCUMENTS OR PROGRAMS AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE INFORMATION, DOCUMENTS OR PROGRAMS (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE INFORMATION, DOCUMENTS OR PROGRAMS TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.