Chapter 3

MRTG

3.1 Overview of MRTG

MRTG is the Multi Router Traffic Grapher, a piece of free software released under the GNU General Public License.[1] It was written primarily by Tobias Oetiker and Dave Rand. MRTG produces Web pages that display graphs of bandwidth use on network links on daily, weekly, monthly, and yearly scales. This can be an invaluable tool for diagnosing network problems because it not only indicates the current status of the network but also lets you visually compare this with the history of network utilization.

MRTG relies on SNMP version one, and optionally SNMP version two, to obtain data from routers or other network hardware. Using the variables described in Chapter 1, MRTG sends SNMP requests every five minutes and stores the responses in a specialized data format. This format allows MRTG to present the daily, weekly, monthly, and yearly graphs without the data files forever growing larger. It does this by summarizing the older data as necessary. The graphs themselves are created in Portable Network Graphics (PNG) format and can be included in Web pages or used in other applications.

3.2 What MRTG Can Help You Do

In the middle of a crisis, or when you are debugging an immediate network problem, MRTG will allow you to view the traffic patterns of many networks at once and quickly determine if one or more is experi-

[1] The GNU General Public License can be found linked under "licenses" on `http://www.gnu.org/`.

encing an abnormal traffic load. The fact that the graphs display the history of the network is key. In practice, it can be difficult to tell from immediate bandwidth and packet-per-second counts alone whether a network is operating normally. If a 100Mb/s link is carrying 85Mb/s of traffic, is this heavy but normal use or is the network straining under an attack? By having the history of the network available, you can look for sudden changes that might account for an operational problem. A denial-of-service attack that attempts to exhaust the available bandwidth on a network nearly always presents as a sudden, sustained increase in traffic levels; the attackers do not have much to gain by slowly ramping up the attack over a period of time.

When you are not tending to an immediate problem, MRTG is useful for studying trends in traffic on your network. It will help you understand how traffic is distributed across your network, plan capacity needs for the future, and so on.

A sample MRTG graph of a day's worth of network traffic is depicted in Figure 3.1. Note that time progresses to the left, not to the right. This is the default configuration and it is indicated at the bottom of the graph both by the small arrow at the left and by the direction of the time scale. Some MRTG configurations choose to increase time to the right, so be sure examine the graph first. The data at the top of the graph represents the amount of traffic sent into an interface, while the data at the bottom represents the amount of traffic sent out from an interface. You can see that over the past day, this router interface

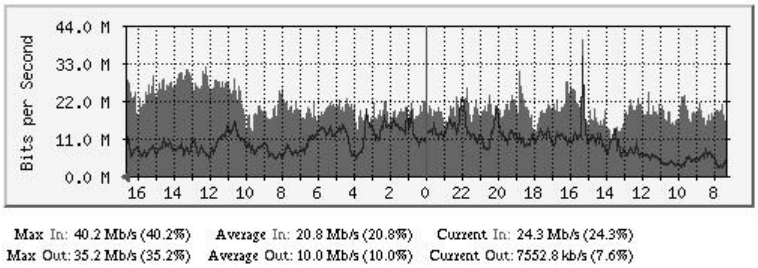

Figure 3.1. Sample Daily MRTG Graph.

typically received about 20Mb/s of traffic and sent about 10Mb/s. You will also notice that just after 3:00 p.m. yesterday, there was a short spike in traffic out of the interface.

While MRTG is most often used to collect data from router interfaces, it can also collect traffic data from switches or servers. In this way, you can monitor the bandwidth use of a particular machine. In fact, MRTG can be configured to collect any statistical data that a device makes available via SNMP.

3.3 Installing MRTG

MRTG is available at `http://www.mrtg.org/`. It relies on a few pieces of software not included in the distribution. In particular, it requires:

- Perl 5.005 or greater
- The GD library
- The PNG library
- The zlib library

You will not need external SNMP software because MRTG comes with its own SNMP implementation. Begin by unpacking the source in a convenient location:

```
Linux% gunzip -c mrtg-2.9.25.tar.gz | tar xvf -
Linux% cd mrtg-2.9.25
```

Install documentation is available from the `doc/` directory in the distribution, but on a modern Linux system, MRTG will build without any special instructions:

```
Linux% ./configure
Linux% make
```

Then as root you may log in and execute the following command:

```
Linux# make install
```

if you would like to install MRTG in the default location, `/usr/local/mrtg-2/`. If you are building MRTG for other platforms, you may find that a couple of necessary components are not already installed on your system. On Solaris, for example, you will first have to build the PNG and GD libraries before you can successfully build MRTG.

3.3.1 Building the PNG Library

The source for PNG is available at `http://www.libpng.org/pub/png/libpng.html`. Download the latest version and unpack it:

```
Solaris% gunzip -c libpng-1.2.5.tar.gz | tar xvf -
Solaris% cd libpng-1.2.5
```

Then examine the `INSTALL` file. It contains a list of makefiles designed for use with different systems. For example:

```
makefile.linux    =>  Linux/ELF makefile (gcc, creates \
   libpng12.so.0.1.2.5)
makefile.hpux     =>  HPUX (10.20 and 11.00) makefile
makefile.macosx   =>  MACOS X Makefile
```

Find the one that is the closest match to your system and make a note of the name. In this example, we use `makefile.solaris`. Copy the makefile file you wish to use to the current directory and try building the package:

```
Solaris% cp scripts/makefile.solaris makefile
Solaris% make
```

If all goes well, there will be a file called `libpng.a` when you are finished. If the build complains about not being able to find the zlib library, you will need to retrieve it from `http://www.gzip.org/zlib/` and build it before continuing. Once you have `libpng.a`, install it on your system from a root account with:

```
Solaris# make install
```

3.3.2 Building the GD Library

The GD library is available at `http://www.boutell.com/gd/`. As above, download the latest version and unpack it:

```
Solaris% gunzip -c gd-2.0.9.tar.gz | tar xvf -
Solaris% cd gd-2.0.9
```

Before running the configure script, you must indicate where GD should find the PNG library by setting the `CFLAGS` and `LDFLAGS` environment variables. Assuming you installed PNG in the default location, `/usr/local/`, you will want:

```
Solaris$ CFLAGS=-I/usr/local/include export CFLAGS
Solaris$ LDFLAGS="-L/usr/local/lib -R/usr/local/lib" export \
   LDFLAGS
```

if you use the bash, Korn or Bourne[2] shell, but:

```
Solaris% setenv CFLAGS -I/usr/local/include
Solaris% setenv LDFLAGS "-L/usr/local/lib -R/usr/local/lib"
```

if you use the csh or tcsh shell.

Now install GD on your system with `make install` from a root account.

3.3.3 Building MRTG

Once the GD library is built with PNG, you can go on to build MRTG. Change back to the MRTG source directory and now run the configure script with `CFLAGS` and `LDFLAGS` still pointed at `/usr/local`:

```
Solaris$ CFLAGS=-I/usr/local/include export CFLAGS
Solaris$ LDFLAGS="-L/usr/local/lib -R/usr/local/lib" export \
   LDFLAGS
Solaris$ ./configure
Solaris$ make
```

And as root:

```
Solaris# make install
```

This will install all of the MRTG software in `/usr/local/mrtg-2`. You can specify an alternate location by running the configure script with the `--prefix` option:

```
Solaris$ ./configure --prefix=/var/tmp/my-mrtg
```

For the rest of this chapter, it is assumed that MRTG is installed in the default location.

3.4 Configuring MRTG

Once the MRTG software is installed, you will need to configure it to monitor your devices. This section describes the configuration process, which includes generating the config file, generating HTML index pages for the graphs, and setting up MRTG to gather data at regular intervals.

[2]The Bourne shell is `/bin/sh`.

3.4.1 Generating the Configuration File

MRTG has a versatile configuration language that makes it difficult to write your own configuration from scratch. Fortunately, the distribution comes with a handy program that will generate a configuration for you.

First you must decide where you would like MRTG to place its generated data files and Web pages. Typically, you will want this to be a directory on a Web server, and it may be publicly readable. You must also decide where to place the MRTG configuration file, which should *not* be publicly readable. It will contain SNMP community names for your devices, which you may wish to keep secret. We will place the MRTG pages and graphs in `/usr/local/apache/htdocs/mrtg/` on our Web server and the configuration file in `/usr/local/mrtg-2/cfg/mrtg.cfg`. The `/usr/local/mrtg-2/cfg/` directory does not exist yet, so we create it, using the account MRTG will run from, and we make sure others won't have access to the directory:

```
Solaris# mkdir /usr/local/mrtg-2/cfg
Solaris# chmod 700 /usr/local/mrtg-2/cfg
```

The `/usr/local/apache/htdocs/mrtg/` directory should also be created, but with permissions appropriate for your Web server to be able to read it.

Now run the `cfgmaker` program to create the configuration file:

```
Solaris# /usr/local/mrtg-2/bin/cfgmaker \
   --global 'WorkDir: /usr/local/apache/htdocs/mrtg' \
   --global 'Options[_]: bits' \
   --global 'IconDir: icons' \
   --snmp-options=:::::2 \
   --subdirs=HOSTNAME \
   --ifref=ip \
   --ifdesc=alias \
   --output /usr/local/mrtg-2/cfg/mrtg.cfg \
     community@router1.example.com \
     community@router2.example.com \
     community@router3.example.com
```

It will spend a short while probing each device in order to build the configuration.

Each option on the command line controls a feature in the configuration. The `--global` options control global configuration features.

`WorkDir` is the directory where MRTG will place data files, and the `bits` option instructs MRTG to report bandwidth in multiples of bits per second instead of bytes per second. The global option `IconDir` specifies the name of a directory in the `WorkDir` directory where MRTG icons will be stored. Copy the icons from the MRTG distribution to this directory now:

```
Solaris# mkdir /usr/local/apache/htdocs/mrtg/icons
Solaris# cp /usr/local/mrtg-2/share/mrtg2/icons/* \
   /usr/local/apache/htdocs/mrtg/icons/
```

and make sure the directory and files are readable to your Web server.

The `snmp-options` variable controls several aspects of SNMP behavior. The only modification we make to the default is to use SNMPv2, which will allow large counter values to work appropriately. If this option is not enabled, you may see incorrect data reported for high-speed network links. You can override the SNMP options set with the `snmp-options` variable for any particular router by appending the options to the name of the router later on the `cfgmaker` command line.

The `subdirs` option controls the organization of MRTG data. By default, MRTG will store all data files in the one specified `WorkDir`. But as each interface will have several different data files associated with it, and each router may hold several interfaces, this can quickly become unwieldy. Setting `subdirs=HOSTNAME` will cause each router to have its own subdirectory under `WorkDir` where all data files for interfaces on that router are stored.

When MRTG stores interface data, it picks a unique filename for each interface. The default name is based on the SNMP index number of the interface. However, there is a serious downside to accepting this as the default. On many routers, adding or removing a board will change SNMP index numbering of other interfaces; interface fifteen yesterday might become interface twenty today. When this happens, MRTG won't know which interface moved where, and the data will become hopelessly confused. One way to avoid this problem is by using the `ifref=ip` option. This tells MRTG to name interface data files by the IP address of the interface rather than the SNMP index number. Under this system, you can add or remove boards, and MRTG will still be able to access the appropriate data. You can even move a network to a new interface and have no problem. The section on maintaining

MRTG goes into greater detail on keeping MRTG consistent with your network configuration.

The `ifdesc=alias` option instructs MRTG to use the interface description when labeling graphs instead of using the default, the SNMP index number. This description will correspond to the string set using the `description` interface command on a Cisco router. If you don't typically set meaningful interface descriptions, you can choose another option for labeling your graphs; several alternate options are listed in Figure 3.2.

The last option to the `cfgmaker` command, `output`, simply specifies the name of the file to which the configuration should be written. Following that, we list each router to be monitored in the form *community*@*router* where *community* is the community name needed to perform SNMP requests.

ifdesc	Label type
`nr`	Index number
`ip`	IP address
`eth`	Ethernet address
`descr`	Description (board name)
`name`	Abbreviated board name
`alias`	Config description

Figure 3.2. Settings for the `ifdesc` Option.

3.4.2 Other Configuration Options

Examine `/usr/local/mrtg-2/cfg/mrtg.cfg` and you will see the results of the config generation. Many options are automatically set, and there are many other options not in use. The MRTG distribution comes with a full reference for configuration options, linked as MRTG Configuration Reference from the Web page in the `doc/` directory. If you decide to use some of these other options, set them by modifying the `cfgmaker` command line above and running the program again. This way when you move router interfaces around in the future, you can run the `cfgmaker` command to detect the changes without losing any modifications you might have made by hand.

3.4.3 Generating Initial Data

Once the configuration file has been created, you can run the `mrtg` program. From the command line, try:

```
Solaris% /usr/local/mrtg-2/bin/mrtg /usr/local/mrtg-2/cfg/mrtg.cfg
```

This will contact your routers and gather the first set of data. Do not be alarmed if you see a long list of errors; this is normal for the first two times you run the program. MRTG is warning you about older data files that it tries to move, but those data files do not yet exist. If you run the program twice more, it should no longer report errors.

After running the `mrtg` command, you will be able to find PNG files and HTML files in `/usr/local/apache/htdocs/mrtg`. A typical Apache Web server installation will allow you to view the pages at `http://server.example.com/mrtg`, where `server.example.com` is replaced with the name of the machine you are installing MRTG on.

If you view one of the HTML files in a Web browser, you will see the empty graphs for a particular interface. Digging out these HTML files is an awkward way to access the graphs because they have odd names like `router1.example.com_10.175.0.1.html`.[3] Fortunately, MRTG also comes with a program that generates an index Web page filled with graphs, one per interface.

3.4.4 Generating Index Pages

Index pages are created with the `indexmaker` program in the MRTG distribution. A simple example would be:

```
Solaris% /usr/local/mrtg-2/bin/indexmaker \
   --output /usr/local/apache/htdocs/mrtg/all.html \
   --columns=1 \
   /usr/local/mrtg-2/cfg/mrtg.cfg
```

This creates a single HTML file called all.html that contains the daily graph for all interfaces on all routers. Open this file in a Web browser and see how it looks. The graphs will not yet contain any data. Click

[3] This assumes you chose to reference interfaces by IP address. If you chose to reference interfaces differently, the name will take a different form but will still be too cumbersome for easy access.

on a graph and you will see the detailed daily, weekly, monthly, and yearly graphs for that interface.

If you have many interfaces, you may find it takes a long time for the index page to load. If this is the case, you may wish to break the graphs up into several different Web pages, perhaps one page per router. You can do this with the `--filter` option. The following example will create the page router1.html with only interfaces from router1:

```
Solaris% /usr/local/mrtg-2/bin/indexmaker \
   --output /usr/local/apache/htdocs/mrtg/router1.html \
   --filter name=~router1 \
   --columns=1 \
   --title="Bandwidth stats for router1.example.com" \
   /usr/local/mrtg-2/cfg/mrtg.cfg
```

Since you will want to generate the index pages for each router a few times while experimenting with MRTG and later when maintaining it, you can use a simple shell script to create an index page for each router. Create a file called `indexer.sh`:

```
#!/bin/sh
for i in router1 router2 router3; do
   echo "Indexing $i"
   /usr/local/mrtg-2/bin/indexmaker \
        --output /usr/local/apache/htdocs/mrtg/$i.html \
        --filter name=~$i \
        --columns=1 \
        --title="Bandwidth stats for $i" \
        /usr/local/mrtg-2/cfg/mrtg.cfg
done
```

and then type

```
Solaris% chmod u+x indexer.sh
```

to make the program executable. Run the program and it will create the index page for each of the routers listed on the third line. You can now open the `router2.html` page in a Web browser. You should see a page of graphs, one for each interface on router2. If you click on the graph of an interface, you will be taken to the page with its weekly, monthly, and yearly graphs.

If you use separate pages for each router, you'll also want to create a small Web page that links to the index page for each router. This you

will have to do by hand. An example might be a `/usr/local/apache/htdocs/mrtg/index.html` that contains the HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
  <title>MRTG Graphs</title>
  <meta http-equiv="Content-Type" content="text/html;
     charset=iso-8859-1">
  <meta http-equiv="Refresh" content="300">
  <meta http-equiv="Cache-Control" content="no-cache">
  <meta http-equiv="Pragma" content="no-cache">
  <meta name="robots" content="noarchive">
</head>
<body bgcolor="#FFFFFF" text="#000000">
  <h1>MRTG Graphs</h1>
  <blockquote>
    <a href="router1.html">router1</a> <br>
    <a href="router2.html">router2</a> <br>
    <a href="router3.html">router2</a> <br>
  </blockquote>
</body>
</html>
```

This page can now act as your starting point for using MRTG. Choose a router to examine from this page, which links to an index page with a graph for each interface on the router. Click on that graph and you'll see detailed information about the interface.

3.4.5 Setting Up Regular Data Gathering

The only task left is to make sure that MRTG contacts the routers and collects data every five minutes. There are two ways to do this. The first and preferred option is to add an entry to the crontab on your server. This is done differently on different systems. On Linux and Solaris, run the `crontab -e` command, which will start an editor from which you may edit the crontab. On other systems, you are expected to edit the crontab manually. Either way, you should be logged in as the user whose account you wish to run MRTG from. Add a line to the crontab:

```
0,5,10,15,20,25,30,35,40,45,50,55 * * * * \
   /usr/local/mrtg-2/bin/mrtg /usr/local/mrtg-2/cfg/mrtg.cfg \
      --logging /var/log/mrtg.log
```

Under Linux only, you can you use a nifty abbreviation:

```
*/5 * * * * \
   /usr/local/mrtg-2/bin/mrtg /usr/local/mrtg-2/cfg/mrtg.cfg \
      --logging /var/log/mrtg.log
```

Note that each crontab entry must be entirely on a single line. The backslashes in the example should not be included in your crontab; they are used here only to indicate that the line continues without breaking. Save the file and quit the editor, and now the MRTG program will run every five minutes to gather data from routers and update the graphs.

The other method for running MRTG at periodic intervals is to configure it to run as a daemon. This will keep the `mrtg` program running in the background once you have started it. This is a reasonable way to run MRTG but has the disadvantage that if the program were to accidentally exit, it would stop collecting data until restarted by hand. If you decide to run MRTG as a daemon, you will need to add the text "`RunAsDaemon: Yes`" to the config, preferably by adding `--global 'RunAsDaemon: Yes'` as an option to the `cfgmaker` command. Be sure to add the `mrtg` command to your system startup scripts so that the daemon will be started when the machine reboots.

3.5 Using MRTG

Using MRTG is, as they say, as easy as falling off a log. Click on the graphs you want and examine the data. There are a couple of subtle points to be aware of, though.

3.5.1 Faulty Data

When looking at the graphs, consider whether the data makes sense before blindly trusting it. We had an MRTG graph indicate that the traffic on an important network had dropped to zero. In reality, no such thing had occurred; the router software encountered a bug that caused it to stop reporting traffic data correctly via SNMP. This became clear after we examined traffic statistics for other devices on the network.

On a separate occasion, we found that over the period of a few weeks, the bandwidth use on one of our external links was steadily dropping. This was suspicious given the time of year and the fact that

the demand for bandwidth seems to be consistently growing with time. It turned out that we had not configured MRTG to use SNMPv2 large counters, and we had hit a point on that link where there was so much traffic that it overloaded the capacity of the smaller counters.

If you are suspicious about the accuracy of MRTG data use tools such as the router command interface to obtain second and third opinions.

3.5.2 Missing Data

Traffic levels on a typical network are somewhat bursty, and as a result, the edge of the data in an MRTG graph is usually jagged. When MRTG cannot gather or store data from a router as scheduled, it fills in the same value it found for the previous interval. This tends to keep the data more in tune with reality than filling in a traffic rate of zero. If you notice a completely flat section of an MRTG graph, such as in Figure 3.3, consider that it is likely a period of time when MRTG could not retrieve or store data from the router. A perfectly constant traffic level is a rare exception. In this example, the file system where MRTG stores its data was unavailable between 8:30 p.m. and 1:00 a.m.

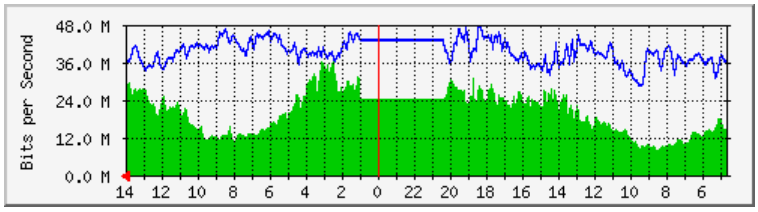

Figure 3.3. Graph with Missing Data.

3.6 Maintaining MRTG

MRTG requires more maintenance than most of the tools described in this book. Each time you move a network or router interface, you will have to make sure the change is reflected in the MRTG configuration. This is why it is to your advantage to save the `cfgmaker` command line and `indexer.sh` script described earlier. They will allow you to generate a new configuration and new index pages with a minimal amount

of effort. You may even choose to run them nightly from the crontab so that changes will be reflected automatically.

Moving networks and interfaces can wreak havoc with MRTG's sense of which data files belong to which network. Your setting of `ifref` on the `cfgmaker` command line will give you control over how MRTG references interfaces. If you set `ifref` to `ip`, MRTG will track an interface by its IP address. This is a good choice since you'll be able to move a network to a new interface and MRTG will still track the data. Other options for the the `ifref` variable are the same as those listed in Figure 3.2, except that the `alias` setting is not available.

In the event that you do make a change that causes MRTG to lose its sense of which data belongs to which network, you can attempt to remedy the situation by finding the appropriate `.log` file under `/usr/local/apache/htdocs/mrtg/router*` and renaming it to be the data file that MRTG expects for the new network.

3.7 References and Further Study

The MRTG Web site at `http://www.mrtg.org/` and the MRTG software distribution both have detailed documentation on using MRTG, including information on `cfgmaker`, `indexmaker`, the configuration language and all other components of MRTG.

The PNG distribution and further information about PNG can be found at `http://www.libpng.org/`. The GD library is at `http://www.boutell.com/gd/`.

`http://www.perl.org/` is one of the many sites devoted to information related to Perl, the language in which most of MRTG is written.