Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Technologies

Diploma thesis

Use of monitoring and graphing tools for network router

Author: Karel Maršíček

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Department of Information Technologies

Faculty of Economics and Management

DIPLOMA THESIS ASSIGNMENT

Maršíček Karel

Informatics

Thesis title

Use of monitoring and graphing tools for network router

Objectives of thesis

Diploma thesis is thematically focused on the issues of monitoring the use of system resources of computers, specifically network routers. The main aim of the work is to create scripts for gathering, saving and presentation of the data. Scripts to control RRDtool will be created in the thesis. Particular aims are to briefly describe operating system GNU/Linux, issues of gathering the data and presenting them in web browser.

Methodology

The methodology of the solved issue is based on description of tools needed for gathering the data and graphing them for such metrics as CPU load, memory usage, etc. In the theoretical part the reader is introduced to the theoretical basis of GNU/Linux and tools needed. In the practical part there is code and its description for issue that is solved. In the practical part is also prepared deb package for GNU/Linux distribution Debian, that solves the dependencies, creates Round Robin Archives and prepares system to deploy monitoring.

Schedule for processing

Timeline of work

1) Preparation and studying of literary resources. 6/2013
2) Creation of the theoretical part of the paper 7/11 2013
3) Creation of the practical part of the paper 8 – 11/2013
4) Completing the paper and correcting the mistakes 12/2013 - 2/2014
5) Submitting the final version of the paper and the abstracts 3/2014

The proposed extent of the thesis

60 - 80 pages

Keywords

Linux, GNU/Linux, RRDtool, Apache, PHP, Debian, GNU

Recommended information sources

BLUM, Richard a Christine BRESNAHAN. Linux Command Line and Shell Scripting Bible, Second Edition. Indianapolis: Wiley; 2 edition, 2011. ISBN 978-1118004425.

Linux Pocket Guide, 2nd Edition. O'Reilly Media, 2012. ISBN 978-1-449-31669-3.

SOBELL, Mark G. . Practical Guide to Linux Commands, Editors, and Shell Programming, A (2nd Edition). Boston: Prentice Hall, 2009. ISBN 978-0131367364.

Linux Networking Cookbook. Beijing• Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo: O'Reilly Media. ISBN 978-0-596-10248-7.

NEMETH, Evi , Garth SNYDER a Trent HEIN. Linux -- Kompletní příručka administrátora, 2. aktualizované vydání. Brno: Computer Press, a.s. ISBN 978-80-251-2410-9.

NEGUS , Chris. Linux Bible 2010 Edition: Boot Up to Ubuntu, Fedora, KNOPPIX, Debian, openSUSE, and 13 Other Distributions. Indianapolis: Wiley, 2009. ISBN 978-0470485057.

The Diploma Thesis Supervisor

Vasilenko Alexandr, Ing.

Last date for the submission

March 2014

Electronic approval: March 21. 2014

doc. Ing. Zdeněk Havlíček, CSc.
Head of the Department

Electronic apporoval: March 21. 2014

Ing. Martin Pelikán, Ph.D.
Dean

Declaration

I declare that I have worked on my diploma thesis titled "Use of monitoring and graphing tools for network router" by myself under guidance of supervisor of diploma thesis and I have used only the sources mentioned in references.

In Prague, March 31, 2014 --------------------------------------

Karel Maršíček

Acknowledgement

I would like to thank my supervisor, Ing. Alexander Vasilenko for his advice and support during my work on this thesis.

Use of monitoring and graphing tools for network router

Použití nástrojů pro monitorování a tvorbu grafů pro síťový směrovač

Summary:

In the theoretical part of the work the reader is introduced to the sphere of computer network monitoring and monitoring of system resources. There are chapters presenting existing solutions such as Cacti, Nagios, Zabbix, Icinga and so on. Some tools are also described, such as Cron for time scheduling of tasks in Unix like systems, Apache web server is also presented and so on. There is also a chapter describing RRDtool and its functionality. In the practical part scripts that gather, save and present the data in the web browser are developed. In the thesis we also prepare package for GNU/Linux Debian that helps to administrator to deploy prepared scripts.

Souhrn:

V teoretické části práce je čtenář uveden do oblasti monitorování počítačových sítí a systémových zdrojů. V práci jsou kapitoly popisující existující řešení, jako například Cacti, Nagios, Zabbix, Icinga a další. V práci jsou také popsané nástroje jako Cron, pro plánování automatizovaných úloh v Unix like systémech, také je prezentovaný webový server Apache a další. V práci je také kapitola popisující nástroj RRDtool a jeho funkcionalitu. V praktické části jsou vyvinuty skripty pro shromažďování, ukládání a prezentaci dat ve webovém prohlířeči. V práci je také připraven balíček pro GNU/Linux Debian, který pomáhá administrátorovi s nasazením připravených skriptů.

Keywords:

Linux, GNU/Linux, RRDtool, Apache, PHP, Debian, GNU

Klíčová slova:

Linux, GNU/Linux, RRDtool, Apache, PHP, Debian, GNU

Table of Contents

1 Introduction

Nowadays monitoring of system resources of computers and such devices as routers plays a very important role. Well deployed alerts can contribute to mitigate amount of outages. After such alert network administrators should react to the problems as soon as possible to decrease impact on the end users. Monitoring is beneficial for not only alerts, with use of that administrator can plan upgrades and changes that are needed. For the purposes of monitoring it is possible to choose from wide spectrum of solutions, from which some have steep learning curve and could be hard to install and configure for new administrators.

In this work we have chosen to use RRDtool. RRDtool is used to store and present time-series data, such as processor load, room temperature etc. This diploma thesis focuses on preparation of scripts that use RRDtool to collect the data into Round Robin Archives and also graph observed data. For prepared scripts is assumed usage on GNU/Linux machine dedicated as router. In the work is also prepared package for Debian package management system, that should help administrators to deploy prepared scripts. Scripts are intended to run on GNU/Linux Debian distribution, that is why in the theoretical part we describe basics of GNU/Linux operating system and tools needed for deployment of our scripts.

2 Objectives of thesis and methodology

2.1 Objectives of thesis

Diploma thesis is thematically focused on the issues of monitoring the use of system resources of computers, specifically network routers. The main aim of the work is to create scripts for gathering, saving and presentation of the data. Scripts to control RRDtool will be created in the thesis. Particular aims are to briefly describe operating system GNU/Linux, issues of gathering the data and presenting them in web browser.

2.2 Methodology

The methodology of the solved issue is based on description of tools needed for gathering the data and graphing them for such metrics as CPU load, memory usage, etc. In the theoretical part the reader is introduced to the theoretical basis of GNU/Linux and tools needed. In the practical part there is code and its description for issue that is solved. In the practical part is also prepared deb package for GNU/Linux distribution Debian that solves the dependencies, creates Round Robin Archives and prepares system to deploy monitoring.

3 Theoretical Foundations

3.1 Linux kernel and its functions

Linux kernel has over 13 million lines of code and, therefore, it is one of the largest open source projects all over the world. Below we look at the kernel and the use of it.

A kernel is the core component in the operating system. It performs functions of a bridge between applications and the data processing performed at the hardware level by using interprocess communication and system calls.

With loading of an operating system into the memory, the kernel loads first and stays in memory up to the moment when the operating system is shut down. The kernel enables low-level tasks like memory management, disk management, task management. [1,2]

3.1.1 Types of Kernels

There exist many ways to build a kernel and architectural considerations when building one from the entire beginning. Generally, we can divide kernels into three types: microkernel, monolithic, and hybrid. Linux is a monolithic kernel while OS X (XNU) and Windows 7 use hybrid kernels. [1]

3.2 History

3.2.1 UNIX

If we want to understand more the popularity and system GNU/Linux itself, we need to get familiar with the history starting from about 30 years ago.

At those times, computers were as big as houses or even bigger. Although the size was a big issue, there were even worse problems, because at that time every computer had a different operating system and software running on one system usually did not work on the other one. When someone was able to work with one system it did not mean that they would be able to work with a different one. Those times were difficult for administrators and also users.

At those times computers were also very expensive and the total cost per unit of computing power was enormous. At those times the world was not very advanced technologically, so the problem with size lasted for another decade. In 1969 in Bell laboratories work on solution of mentioned compatibility problems started.

The new feature allowing code recyclation was very important. Before that all available computer systems were specifically developed for each system. On the other hand, there was need to write special code, today known as kernel for UNIX. The operating system and other programs and features were written in C programming language. The programming language C was developed for creation of UNIX system. This allowed much easier development of operating system and, moreover, thanks to that it could run on various types of hardware.

The adaptation of software vendors was fast, because it was made possible for them to sell about ten times more software without effort. Development of UNIX continued and more things became possible. More hardware vendors started to support UNIX in their products.

By the end of 80's, already many people had personal computers at home. In those times there were also versions of UNIX available for PC architecture, but they were very slow and not used much. In 1983 Richard Stallman developed GNU and UNIX compatible OS that was intended to be free. It took attention of developers from several different countries and they contributed with big amount of utilities and programs. After some time most of of the system components were prepared, but they were still missing kernel for that operating system. [3,4]

3.2.2 Linus and Linux

Linux kernel was developed by Linus Torvalds. In 1991, still a student at the University of Helsinki where he had been using a non-free Unix-like system called Minix, he started working on his own kernel. The first step was developing device drivers and hard-drive access, which resulted in Version 0.01. This kernel, which is called Linux, was later combined with the GNU system and a complete free operating system was produced.

From the beginning the goal of Linus was to have an operating system that would be free and compliant with UNIX. That is why he used POSIX standards.

Thanks to help of big amount of supporters new drivers became available for all kinds of new hardware. It was almost that with every new available piece, somebody bought it and created the driver or tested the device with Linux.

About two years after Linus posted information about his intention in mailing there were 12000 of Linux users. Project took attention of many people and its growth continued within the boundaries of POSIX standard. After several years of development Linux became mature operating system. It can be said that Linux is a full UNIX clone and it can be used on workstations and on high-end, middle-range servers and so on. Linux version 0.12 was released in January 1992. It had improved and stable kernel. Next release was version 0.95 and it became full-featured system. After some time Linux became an underground phenomenon. There was an increasing amount of programmers that were programming, debugging, and improving the source code.

Torvalds released Version 0.11 under a freeware license that he devised on his own, but then the Version 0.12 came under the well established GNU General Public License. During the next few years new free software was developed for Linux.[3,4]

3.2.3 Current application of Linux systems

Nowadays there are many distributions intended to be used on desktops, but it should be admitted that Linux has a very low market share in that area.
From the server side, GNU/Linux is popular and has a reputation as a reliable and stable platform, that is used for running databases and other services for companies such as Amazon, German army, US Post Office the and so on. We can mention Internet service providers, that became keen on Linux as firewall, proxy and also web servers. Another example of usage are clusters of Linux machines that were used in the creation of movies such as "Shrek", "Titanic" and others. We can also read that most supercomputers from top500 use Linux. Also nowadays Android operating system is based on Linux kernel and many embedded devices such as networking equipment or consumer electronics use Linux. [3]

3.3 GNU/Linux Debian

Debian GNU/Linux was created by the Debian Project. The Debian Project was founded in 1993 by Ian Murdock. This association has made a common cause to create a coherent, free and complete operating system. Debian is not running only with kernel Linux, there is also release with FreeBSD and Hurd.

Commitment of Debian to free software distribution and its openness has brought it a lot of contributors in the technical community. Among all GNU/Linux system, Debian has been used the most as the basis for other Linux distributions, such as KNOPPIX, Ubuntu, Linspire , Xandros and so on.

Debian has become successful in spite of the lack of formal enterprise initiatives, large corporate sponsors, official certification and training programs. Keen Debian's users consider that Debian is one of the most stable and reliable GNU/Linux distribution. Debian contains more than 37500 packages. It is indeed well tested and new versions are not involved into stable version unless they are extraordinarly stable. [5,6]

3.4 Debian Packages

There are two basic forms of packages for Debian. Those are binary and source forms. The binary packages contain the files that can be extracted to the system with use of package management tools. The source packages contain the source code and instructions for the building that Debian build tools use to build the binary package. There are associated data files to the programs. Control data are contained in the package and it enables the package management tools to support advanced features.

There is main control file containing version and interrelationship data, such as dependencies of the package. Version is compared with already installed software to find out if there should be an upgrade. Dependencies contain the information which package must or cannot be installed.

Interrelationship fields include Depends, Replaces, Provides, Conflicts, Recommends, Enhances and Suggests. For further information go to http://debian.org/doc/debian-policy/ch- controlfields.html.

The package management tools can be instructed by optional preinst, prerm, postinst and postrm files to perform functions before or after package is installed or removed. For instance, most packages that contain daemons (like ftpd for FTP) include a postinst script that starts the daemon automatically.
A conffiles file can determine specific files as configuration files inside the package, that marks them to not to be overwritten during automatically during the upgrades. All files in directory /etc are by default configuration files.

There are two special package types such as virtual and meta. We will look at them in this paragraph. The Meta packages are basically binary packages, but do not contain any files. Nevertheless, they have dependencies with several of other packages. When we install a Meta package, it automatically leads to installation of all packages that they depend on. It can be used as an easy method in order to install a set of related packages.

The virtual packages in fact do not exist, but those are interrelationship fields for package system. They can be used when there are several possible packages that could fulfill specific depends field requirements. For instance, smail and sendmail both have functionality of mail transfer agent and there can be installed one of them, when mail transfer agent is needed as dependency. But it has to be observed whether conflict does not appear among the packages.[7,39]

3.5 Debian Releases

The Linux distribution is a collection of specific package versions. After period of development distribution is ready to release and can become a release. For Debian new distributions are released approximately every two years. In case of Debian terms release and distribution these terms can be used instead of each other and usually when stable release is reached.

Distributions gets a code names (some had woody, sarge etc from characters in the Toy Story movie). There are refered with release tags in order to identify the state of development in release cycle. Those active releases are unstable, testing and stable. At the desktop computer stable release could be considered as older, because newest versions of desktop environment and other packages. But on the other hand, packages in stable release are usually very well tested and stable and reliable for use on servers etc. Below we look deeper at particular development phases.

New versions and brand new packages are uploaded into Debian archive then are imported to unstable release. This release always contains the newest versions of packages, but they have not been tested thoroughly whether their installation will not cause unexpected behavior of the system.

When a package that was assigned to unstable area and no significant bugs were found it can be imported into the testing release. Testing release stays open to changes and after some time it gets frozen in preparation. In frozen state only the changes fixing significant bugs are imported.

When all critical bugs for release have been dealt with in frozen distribution, then the manager of the release can declare the release to be ready to replace stable distribution. When it happens the previous stable version becomes obsolete, but for some period of time still stays in archive. As time flows, new versions come and the process begins again. [8]

3.6 Licenses used

The license of Linux is covered by GPL, that stands for GNU General Public License. It is sometimes called a "copyleft" license. Richard Stallman created the GPL so that the GNU software is not made proprietary.

Copyleft’s purpose is to keep software free, as the opposite of copyright, which is a means of privatizing software. A number of provisions are made by it for the distribution and modification of "free software", where "free" means not only cost, but also freedom.

Linux was at first released under a licence that had more restrictions than GPL. It allowed the software to be distributed freely and also modified, but it was preventing money operations while the distribution and usage. Unlike that, under GPL it is allowed to sell it and also make profit from it, but it does not enable them to distribute the software.

It should be clarified that when we mention Free software, that GPL covers, it is still not public domain. It means that the software is copyrighted and is not literally owned by public. The author of the software is defined and there are standard international laws applied.

It is stated in GPL, that programmers are allowed to copy freely and also distribute for instance DVD copied of a program’s source codes, fulfilling the condition that each copy displays a notice of copyright, intact GPL notices, disclaimer of warranty and also a copy of GPL. In the second term it is dealing with modifications and distribution of the software and requires the same conditions as in the first term and also notification about the changes in the software.

Nevertheless, some restrictions for selling the software are defined by the GPL. Firstly, they cannot restrict the rights of purchasing users that buy DVD-ROM of GPL software. Buyer can resell or offer the DVD free of charge. Secondly, it must be visible to a user that the software is really under the GPL license. Thirdly, the source code has to be provided with the copy or information, where it can be downloaded. The above-mentioned enables anybody who buys GPL software to modify it. [9,10]

3.7 Cron

Cron is a system daemon for executing desired tasks in the system background at times that were designated.

A crontab is a simple text file that contains a list of commands that are supposed to be run at specified times. It may be edited with the use of a command-line editor. The cron daemon controls these commands and their run times and executes them in the system background. Each user has a crontab file that specifies the actions and times at which they are to be executed. These tasks will run no matter if the user is actually logged into the system. There is also a root crontab for tasks that require administrative privileges. This system crontab allows scheduling of system wide tasks, such as system database updates and log rotations. [11,12]

3.7.1 Use of Cron

We can edit crontab file when we type in terminal

```
crontab -e
```

But it is important to realize that there is different file for different users, so it is reasonable to edit root's file. [11]

3.7.2 Explanation of schedulling

Lets look at the following line.

```
* * * * * /bin/execute/this/script.sh
```

We can see five stars. Starting from the left they represent minute, hour, day of month, month, day of week. Star means every, so in this case every minute of every hour of every day.

We can choose minute with range between 0 to 59, hour from 0 to 23, also day of month from 1 to 31 and day of week 0 to 6, where 0 is sunday.

So for example:

```
0 2 * * 5 /bin/execute/this/script.sh
```

It runs every Friday at 2 am.

Next example:

```
01,31 * 1-18 * * /bin/execute/this/script.sh
```

It runs at 01 and 31 past every hour on the 1st through 18th of each month. [11,12]

3.8 Apache

The Apache web server, more popularly called just Apache, is an open-source web server platform that lies in the basis of the majority of the websites we see today on the World Wide Web. If we look back at the time when it was developed in 1995 and gradually adopted as a preferred server platform on the web, we can say that Apache was the main driving force behind today's web expansion. As a web server 'pioneer', Apache has become a standard for developing other successful web server platforms.

The Apache web server was developed by Apache Software Foundation open source community. The efforts of many supporters all over the world ensure its good maintenance and regular updates with new useful features and functionalities up to the latest quality and security requirements in HTTP service delivery. However, because of the fact that the source code is freely available, anybody can adapt the server for specific needs, and there is a large public library of Apache add-ons.

There are two theories about the origin of the name Apache, that are significantly different. One of them, the more popular one, says that its name is a tribute to the Native American Apache Indian tribe, which is well known for its endurance and war skill. According to the other story, the name Apache was developed from existing NCSA code plus various patches, therefore the name a patchy server, or Apache server. [13,14]

3.9 PHP

PHP is a programming language that is used for building dynamic, interactive Web sites. PHP programs usually run on a Web server and serve Web pages to visitors on their request. To the key features of PHP belongs the possibility to embed PHP code within HTML Web pages, which makes it very easy to quickly create dynamic content.[15]

PHP originally stood for Personal Home Page, and now it has changed for PHP: Hypertext Preprocessor, which explains its core purpose: to process information and produce hypertext (HTML) as a result.

PHP is a server - side scripting language. It means that PHP scripts, or programs, usually run on a Web server. As an example of a client – side scripting language we can give JavaScript, which commonly runs within a Web browser. Moreover, PHP is an interpreted language, which means that PHP script is processed by the PHP engine every time it is run.

From the point of view of security, there are many configuration options controlling its behaviour, because there are many different ways of utilizing PHP. A large choice of options ensures that you can use PHP for many different purposes, but it also means the possibility of an insecure setup because of combinations of these options and server configurations. [15,16]

3.10 Introduction to work with command line

3.10.1 Bash

Bash is the shell, or command language interpreter, that is important part of GNU/Linux system. Bash is a sh-compatible shell that incorporates useful features from the C shell (csh) and Korn shell (ksh). Most of tasks that users could do with computers can be done through both GUI (Graphical User Interface) and CLI (Command Line Interface), some things are more easily achieved from one or the other. For instance, changing file permissions of a folder and all its sub folders is more easily achieved using cli instead gui.

For every command (program) in command line several parameters can be used, which are usually well described in manual pages. In the practical part of thesis we use several commands, therefore, we describe them in the following part. [17,18]

3.11 Utilities for data collection

3.11.1 Df

Command „df“ (disk free sometimes also referred as disk filesystem) is used to get information about disk space usage on filesystem on GNU/Linux system. There are several options to use, for instance „-h“ will show information in human readable format, that means details in bytes, megabytes, etc. With parameter „-m“ we obtain information in megabytes. There are many more possible parameters that we can read about in manual page. [19]

3.11.2 Free

Command „free“ displays total amount of free and used memory in the system. It shows information about physical and swap memory and also buffers that are used by kernel. There are several options that we may use, for example “-b” display amount of memory in bytes. Parameter “m” displays memory in megabytes. Analogically “-g” in gigabytes. [20]

3.11.3 Iostat

Command „iostat“ reports CPU statistics and also input/output statistics for devices and disk partitions. The command is used to monitor system input/output device loading by observing the time the devices are active in relation to their average transfer rates. There are three types of reports generated by iostat, namely CPU Utilization, Device Utilization and Network Filesystem report. In the CPU Utilization Report there is %system, that stands for CPU utilization that occurred while executing at kernel (system) level. The %user stands for percentage while executing at user (application) level. Again further information could be found in manual page. [21]

3.12 Tools for data filtering

3.12.1 Grep

We use the grep command for searching text or searching the given file whether it contains lines with a match to the given words or strings. Grep shows the matching lines by default. It works very well with regular expressions, when it outputs lines that match our search. It is thought that grep is one of the most useful commands on Unix-like operating systems.

There are several very useful options, for example

```
grep -r "10.93.0.1" /etc/
```

Parameter „-r“is used to search recursively, in this case it reads all files under each directory for a string with given IP address. Another example is parameter „-c“, it reports number of lines that match the criterion. [22]

3.12.2 Cut

Command “cut” is used for text processing. While using it, we may extract portion of text from a text file by selecting only specified column or field. There are several very useful options to use. For instance, if there was a line in a file

```
395628:Research:Novak:Petr
with command cut -c1-6 input_data
```

we would receive “395628”. The command chooses columns 1 to 6. In case that file contained more lines, they would also be processed. [23,24]

3.13 Commands for text manipulation

3.13.1 Awk

The awk is scripting language for text processing with similar syntax as C programming language. It is included in most of GNU/Linux distributions as GNU awk, gawk for short. Awk breaks each input line into fields. A field is string of consecutive characters, that are delimited by whitespace by default, but there are also options to change that. Because of its character it is very good to handle structured text files, especially tables.

```
root@debian:/# echo one two three four five | awk '{print $3}'
three
```

With previous example we use echo to display the line of text and we give its output as input for awk in order to select third column. Awk is very powerful this was just simple example. Further information are available in manual page. [26, 27]

3.13.2 Sed

Sed is the name of a stream editor. Stream editors are used for performing basic text transformations on an input stream, which may be a file or input from a pipeline. In some ways sed is similar to an editor which allows scripted edits,but functions by making only one pass over the input(s), and, in such a way, is more efficient. Sed's distinctive feature is the ability to filter text in a pipeline. [28]

3.14 File permissions

There may be many users with login accounts. In order to maintain security and privacy users have restricted access only to particular files, not to all of them. Two aspects can be considered.

Who has permission?

For every file and directory there is an owner who is allowed to do anything with it. Usually the user who created the file is the owner, but it can be more complex Respectively, a group of predefined users may have permissions. Groups are defined and can be changed by root, which is system administrator account.

What kind of permission is granted?

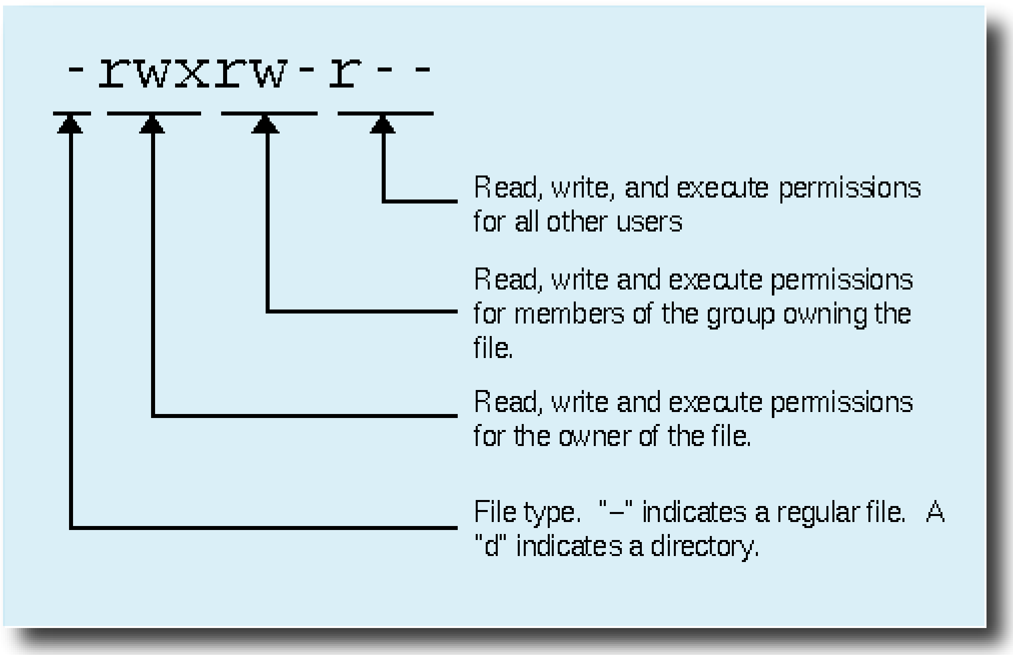
For file owners, groups, and everyone else there can be set a permission to read, write, and execute particular files, where with write we mean that file can be modified. Permissions also apply to directories, that users are allowed to read (access to the files within the directory), write (create and delete files within the particular directory), and execute (that user can list the content).

In order to see the ownership and permissions of a file, there should be run:

```
$ ls -l disk.html
-rw-r--r-- 1 root root 1524 Jan 21 22:09 disk.html
```

In order to see the ownership and permissions of a directory, there should be run:

```
$ ls -ld png/
drwxr-xr-x 2 www-data root 4096 Mar 3 09:37 png/
```


-rwxrw-r--
Read, write, and execute permissions for all other users
Read, write and execute permissions for members of the group owning the file.
Read, write and execute permissions for the owner of the file.
File type. "–" indicates a regular file. A "d" indicates a directory.

Ilustrace 1: Permisions [45]

[45, 46]

3.15 Reasons for network monitoring

Network monitoring can mean tracking Internet usage and data packets and it also ensures that the computers that need to run are up. Network monitoring is important nowadays. It is needed to mention that the role of networking monitoring is to ensure functionality of IT network infrastructure, that can help to save time and money. There are several reasons why companies should deploy network monitoring. [28]

3.15.1 Being informed about the situation

Due to network monitoring solutions administrators are informed about the computers, devices and resources on the network. Being able to detect problematic access points and saturation of network resources is important. Having access to real-time mapping and track user activity is also helpful. The possibility to automatically discover, track, map and document what's deployed across entire infrastructure gives the most up-to-date picture of the network at all times. Without these features, there would be need to wait, until someone complains about some not functional device in order to repair that. [28]

3.15.2 Planning of upgrades or changes

In case when we are running out of disk space at disk array or the bandwidth to a concrete subnet is constantly running close the limit, it could be time to implement a change. Network monitoring applications enable network administrator to keep track of this type of data and easily make needed changes. [28]

3.15.3 Quick diagnoses of the problem

There may be a situation when one of servers is unreachable from the intranet. Without network monitoring, it is often not easy to tell, what cause the problem. The problem may be in not responding server, the switch that server is connected to, or the router. The possibility to distinguish where the problem is saves time. The software can warn us about overloaded circuits or servers, network slowdowns, and other signs of potential trouble so that technicians can deal with the problems before they influence network users. [28, 19]

3.15.4 Showing the situation of the network to others

Graphical reports are very helpful in explaining the health of the network and presenting its activity. They are useful tools in providing an SLA (Service Level Agreement) or showing that a problematic device needs to be replaced. [28]

3.15.5 Knowing when to apply disaster recovery solution

In case when there is enough warning, the operation of important servers may be transferred to a backup system until the primary system can be repaired and put into use again. Without network monitoring, network administrator may not know there is a problem until it is too late. Performance thresholds can be set to trigger a warning through a text message or email and an organization's fault management process responds to the alarm. [28,29]

3.15.6 Keeping track of customer-facing resources

Many services on the network are applications running on a servers (HTTP, SMTP, FTP, etc). Network monitoring can intermittently these applications and make sure that customers will be able to connect to the servers and use the services as they are supposed to. This helps to check on Service Level Agreements. [28,19]

3.15.7 Being informed about network status from anywhere

Many network monitoring applications provide remote viewing and management remotely, some of them even from smart phones. That way, if an administrator is away and a problem appears, you can log into your Web interface and see what is wrong. [28]

3.15.8 Saving money

Above all, network monitoring helps decrease the total amount of downtime and time for investigating problems. As a result, it takes fewer man-hours and less money when problems occur. [28]

3.16 System monitoring and statistics

For purposes of maintenance of routers, servers etc. it is very helpful to have available statistics of network performance, used memory, processor load, etc.
There already exist very robust and complex systems, for instance, Cacti.
Cacti, nagios and other monitoring systems are described in the previous chapters.
But such systems have several disadvantages, they are very complex, hard to configure and some administrators are not satisfied with them. That is why we have decided to prepare our own solution based on RRDtool for creating graphs of mentioned performance statistics.

3.17 Introduction to RRD - Round Robin Database

Round Robin Database was developed by Tobias Oetiker, a system manager at the Swiss Federal Institute of Technology.
RRDtool (Round Robin Database tool) is a system which is used to store and display time-

series data, for example, server load average, network bandwidth, machine-room temperature etc.

RRD graphics are built from data which are extracted from a RRD database. RRDtool is also able to create and fill with data this database. RRDtool stores data, which makes it a back end tool. The RRDTool command set makes the creation of graphs possible, which makes it a front end tool too. Other databases just stores data and is not able to create graphs.

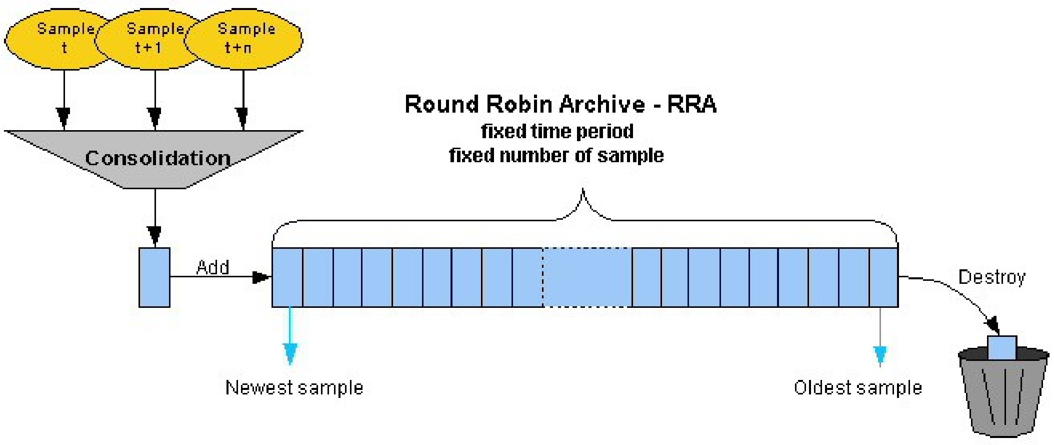
RRDTool databases are in the first place used for monitoring purposes and, therefore, their structure is very simple. The parameters which must be defined are variables that hold values and archives of those values.

In case of linear databases, new data appear at the bottom of the database table. Therefor its size continues to increase, while the size of an RRDTool database is determined at the time of its creation. If we imagine an RRDTool database as the perimeter of a circle, the data will be added along the perimeter. When new data reach the starting point, the already existing data are overwritten. In such a way, the size of an RRDTool database remains the same all the time. For this reason it is called Round Robin.

It is possible to configure RRDTool to calculate the rate of change from the previous to the current value and store the calculated value instead. The structure of RRDTool database is such that it needs data at time intervals which are predefined. In case when no new value is supplied during the interval, an UNKNOWN value for that interval is stored. Therefore, when using the RRDTool database, it is necessary to use scripts that run at regular intervals to ensure that a constant data flow will update the RRDTool database.[30, 31]

3.17.1 RRD create graph with consolidation functions.

It is possible to log data at 5 minute interval, but there may also be the need to know the development of the data during the last year. This can be done by simply storing the data at 5 minute interval, for one year. While this would take considerable space on the disc, it would also be time consuming to analyze the data if there is a need to create a graph covering the whole year. RRDTool offers a solution to this problem due to its feature of data consolidation. When setting up a Round Robin Database (RRD), it can be defined at which interval this consolidation will occur and what consolidation function (CF) should be used to build the consolidated values: average, minimum, maximum, or last. It is possible to define any number of different consolidation setups within one RRD. They are all maintained on the fly when new data is loaded into the RRD. [30,31]



Ilustrace 2: Round Robin Archive principle [30]

3.17.2 RRDtool store the consolidated values in Round Robin Archives

Abbreviation RRD means Round Robin Database. It is a database that collects the data against time. The Round Robin explains the fact that it is only possible to store a certain amount of data points. When the end of the database is reached, then it wraps back to the beginning again. As a result, RRD database files will never grow in size.

The principle of work is as follows: If we choose to store 1000 values at 5 minute interval, then RRDTool will allocate space for 1000 data values and a header area. In the header there will be stored a pointer that will tell to which of the values in the storage area we last wrote. We write the new values to the Round Robin Archive in a round robin way. In our case it automatically limits the history to the last 1000 values.

Usage of RRAs ensures that the RRD will not grow with time and that old data will automatically be overwritten. The consolidation function makes it possible to keep data for a very long time, at the same time gradually reducing the resolution of the data on the axis of time. By using different consolidation functions (CF) we can store exactly the type of information that is actually of interest to us.

Consolidation functions can be of 4 types:
AVERAGE - the arithmetic average of the collected values
LAST - the last collected value
MIN - the smallest collected value
MAX - the highest collected value [30]

3.18 Graph creation with RRDtool

The ability to create graphs is another important feature of RRDtool. The graph command uses fetch command internally to retrieve values from the database. With the retrieved values, it draws graphs as defined by the parameters written in the scripts.
One graph can show various Data Sources from one RRD database. There is also a possibility to show the values from several RRD databases into one RRD graph.
It is often necessary to do some calculations with the data before drawing the graph. For instance, memory usage values are usually specified in KBytes and network traffic is specified in Bytes. Graphs for presenting such values could be more understandable if values would be in MBytes and mbps. With RRDtool we may define such conversions.
There is five different shapes that we may use in a graph such as AREA, LINE1, LINE2, LINE3 and STACK.

- **AREA** is represented by a solid colored area with values as the boundary of this area.
- **LINE1/2/3** (increasing width) are just plain lines representing the values.
- **STACK** is also an area but it is ``stack''ed on AREA or LINE1/2/3.

It is important to note that variables are plotted in the same order as they are defined in graph command. In the Unix world commands for RRDtool are probably the ones where we deal with the longest line, because whole script is usually written as one line. Therefore, we must define STACK only after we define AREA/LINE. Formatted comments can also be put within the graph. Further information can be found in RRDtool man page.[30,31]

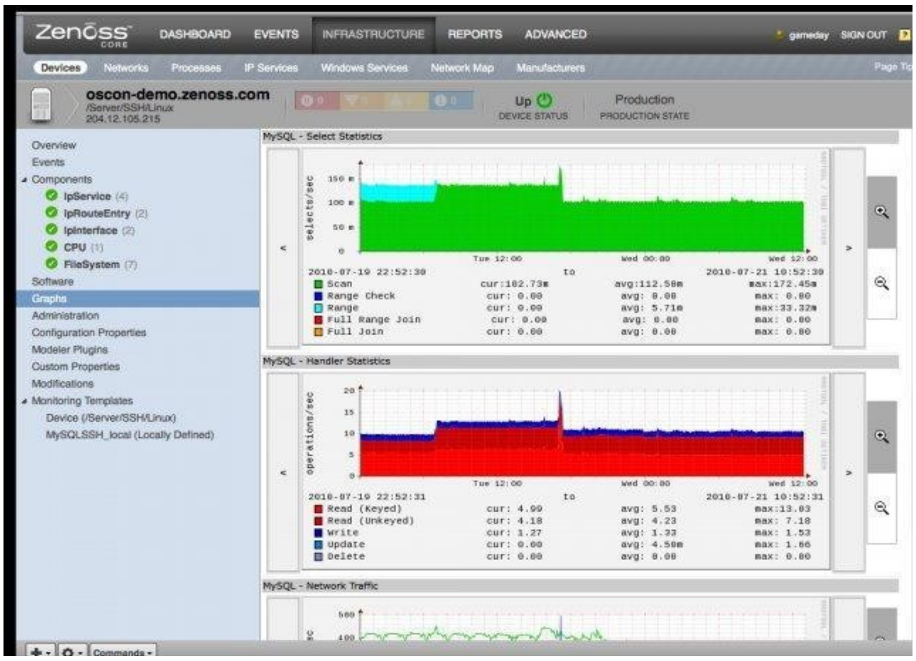
3.19 Nagios

Nagios is an open source, host and service monitoring software. It is developed to run under the GNU/Linux operating system, but it works well under most of Unix-like systems. It is widely deployed in data centers. There is monitoring daemon, running intermittent tests of host and services, that uses external plugins. Nagios is considered to be the most popular system and network monitoring solution, it is extremely complex and robust. Configuration can be done to monitor such services as HTTP, SMTP, POP3, SSH, FTP etc, as well as system resources such as processor load, disk space usage, etc. Environmental factors such as temperature in server room, or air humidity can also be monitored. Due to the monitoring outages can be mitigated and especially it can ensure running of mission-critical applications. Nagios stores information in MySQL database. For Nagios many third party tools are available and there is also commercial support option. The daemon can be used to send notification after exceeding given threshold to send email with warning, text message or even instant messaging. [32, 33,50]

3.20 Zenoss

Zenoss Core is the name of an open source network and system monitoring platform, sponsored by Zenoss, Inc. There are two versions of Zenoss developed by Zenoss, Inc.

They are called Core and Enterprise. The Core version is supported by and belongs to the community. There are value added features that comes in Zenoss



```
2010-07-19 22:52:30              to              2010-07-21 10:52:30
🟩 Scan               cur:102.73m        avg:112.58m        max:172.45m
🟦 Range Check         cur: 0.00          avg: 0.00          max: 0.00
🟦 Range               cur: 0.00          avg: 5.71m         max:33.32m
🟥 Full Range Join      cur: 0.00          avg: 0.00          max: 0.00
🟧 Full Join           cur: 0.00          avg: 0.00          max: 0.00
```

```
2010-07-19 22:52:31              to              2010-07-21 10:52:31
🟥 Read (Keyed)        cur: 4.99          avg: 5.53          max:13.03
🟥 Read (Unkeyed)      cur: 4.18          avg: 4.23          max: 7.18
🟦 Write               cur: 1.27          avg: 1.33          max: 1.53
🟦 Update              cur: 0.00          avg: 4.50m         max: 1.66
🟦 Delete              cur: 0.00          avg: 0.00          max: 0.00
```

Illustration 3: Zenoss [48]

Enterprise version, those are, for instance, synthetic web transactions, extended report library, global dashboard and certified monitors (called ZenPacks). Though we are dealing with open source, Zenoss Enterprise with mentioned additional features can be sold as commercial software with support.

Zenoss Core is complex and its interface could be considered as complicated, but flexible. There was a dramatical improvement of interface with version 2, but there still is room for guidance, the software is not as intuitive as some administrators wish it to be. Zenoss Core solution for monitoring involves the following:

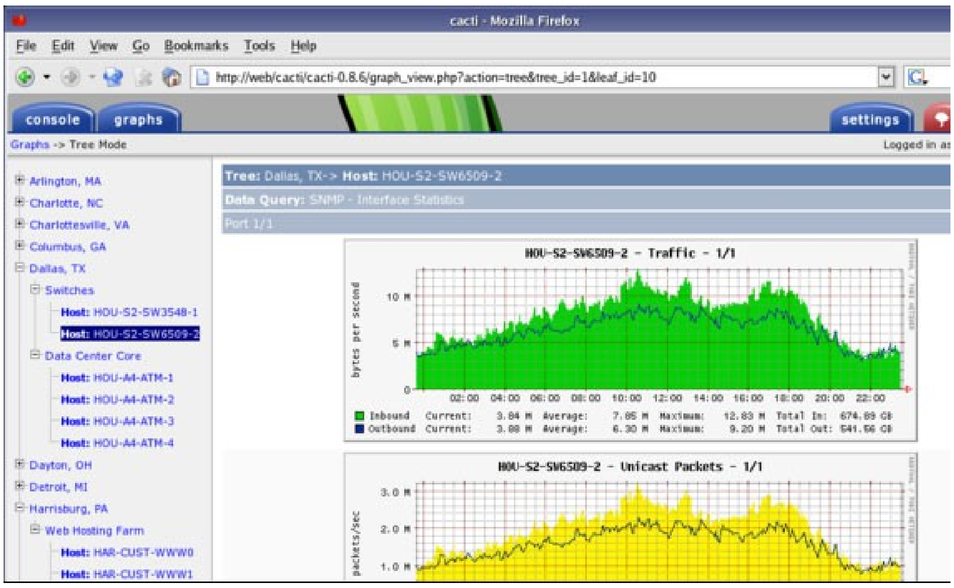
- Monitoring reports
- User and alert management
- Monitoring of availability
- Plugin architecture
- Management of devices
- Management of events
- Graphs of performance

In order to monitor IT assets, such as switches, routers, servers and so on, Zenoss Core is installed to the server. Although Zenoss is intended to run on GNU/Linux server, there are virtual appliances also for Windows and Macintosh that users can use to install working version using VMware.

The web portal is a front end of the Zenoss and it is where administrator spends most of his time in connection with Zenoss. Administrator deals there with a single point of access that does not require specific knowledge about system running at monitored systems. In the web the administrator has available drag and drop dashboard that displays network health. [47, 49]

3.21 Cacti

Cacti is an open source, web-based graphing tool designed as a front-end to RRDtool's data storage and graphing functionality. Cacti allows a user to poll services at predetermined intervals and graph the resulting data. It is generally used to graph time-series data of metrics such as CPU load and network bandwidth utilization. A common usage is to monitor network traffic by polling a network switch or router interface via SNMP.



Ilustrace 4: Cacti [40]

The front end is written in PHP; it can handle multiple users, each with their own graph sets, so it is sometimes used by web hosting providers (especially dedicated server, virtual private server) to display bandwidth statistics for their customers. It can be used to configure the data collection itself, allowing certain setups to be monitored without any manual configuration of RRDtool. Cacti can be extended to monitor any source via shell scripts and executables. [36]

3.22 Icinga

Icinga was is a Nagios fork and it is backward compatible. That means that in Icinga we can use plugins, addons and configurations from Nagios. It preserves all existing features of Nagios, but it also builds patches and features that community requests. [34, 35]

It is an enterprise grade open source monitoring system that is used for network monitoring and we can use that for almost all network resources, it can notify the user about recoveries and errors and it prepares data for reporting about the performance. We can monitor large, complex environments across distant locations, it is extensible and scalable. It is distributed under GPL V2 and can be used for free, distributed and also modified.

Monitoring of network and its resources

- Network services: HTTP, SMTP, POP3,NNTP, SNMP,, PING, and so on
- System resources: CPU load, disk usage, memory usage, and so on
- Switches, routers
- Temperature and humidity sensors, and so on

Alerts and notifications

- Notifications about problems that occurred and were resolved (via email, sms user-defined method)
- Escalations to other support groups

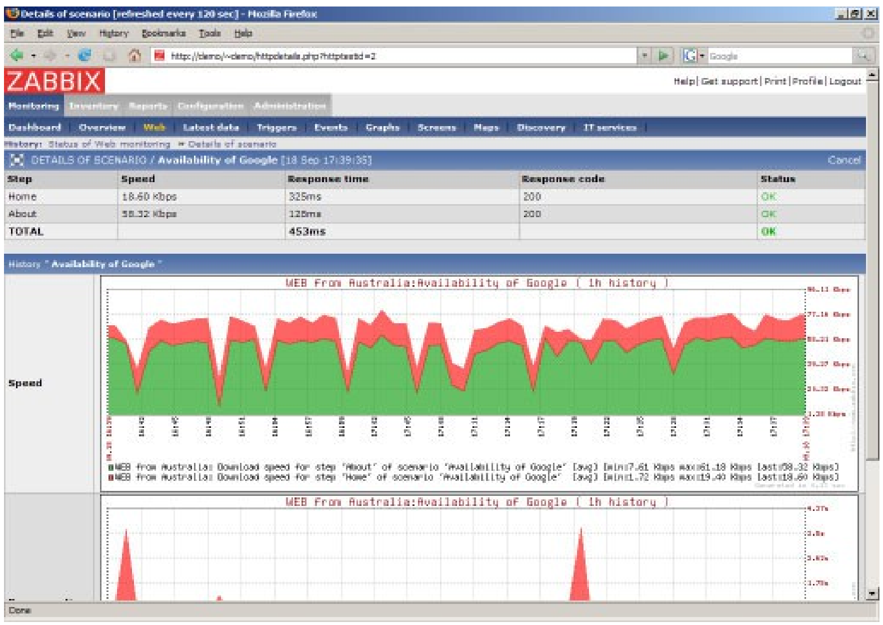
Reports about performance and for planning of upgrades

- Capacity We may plan the growth
- Chart With use of addons for PNG images and so on

[34, 35]

3.23 Zabbix

Zabbix is software that helps to monitor different aspects of IT infrastructure in many ways. It can monitor wide spread of information that an administrator could use. It is a semi-distributed monitoring system where the management is centralized. Most of installations would use Zabbix agent, as monitoring with proxies and nodes as distributed monitoring. But many installations have a single central database.



DETAILS OF SCENARIO / Availability of Google [18 Sep 17:39:35]

Step	Speed	Response time	Response code	Status
Home	18.60 Kbps	325ms	200	OK
About	58.32 Kbps	128ms	200	OK
TOTAL		453ms		OK

Details of scenario [refreshed every 120 sec] - Mozilla Firefox
ZABBIX
Help | Get support | Print | Profile | Logout
Monitoring
Inventory
Reports
Configuration
Administration
Dashboard
Overview
Web
Latest data
Triggers
Events
Graphs
Screens
Maps
Discovery
IT services
History " Availability of Google "
WEB From Australia:Availability of Google (1h history)
Speed

Ilustrace 5: Zabbix [41]

Zabbix provides the following features:

- web interface that is centralized and easy to use
- Server, running on most Unix-like operating systems, including FreeBSD, OpenBSD, GNU/Linux, AIX and Solaris
- Native agents for most Microsoft Windows versions and Unix-like operating systems
- Use of direct monitoring of SNMP and IPMI devices
- Capabilities to draw graphs and other visualization capabilities
- Notifications that can be integrated with other systems
- Configuration with possibility to use templates
- A lot of other features, allowing to implement a complex monitoring solution

If we consider a simplified network from the Zabbix perspective (when we place Zabbix server in the center), the communication of different monitoring aspects is important. In this case the central object is the Zabbix database, that supports several back-ends. Zabbix server was developed in C programming language, and the web front-end was written in PHP. Back-end and front-end can be both running on the same computer or on another server. In case of running front-end and back-end on separated computers, both of them need to access the database. When multiple devices are being monitored and some of them are separated by the firewall, then it obtains the data through a Zabbix proxy. [37]

3.24 Munin

The Munin is an open source application that makes it possible for the administrator to monitor and collect data for PCs, networks, SANS, and so on. Munin is considered to be very flexible also because of plenty of plugins, that are available. Munin comprises three main components:

- Munin master (the component for server)
- Munin node (the component for a client)
- Munin plugin (the program for collecting data)

The client nodes are asked every 5 minutes by the master for a status update. The node then checks its configured plugins and uses them to create the requested information about the status.

For instance, a plugin can return the information about processor load or the information about how full the system disk is. Master plots the data according to the information provided from the nodes. You can easily view these graphs by using a web browser.

The graphs are created in such a way, that there is information for last day, week, month, etc up to the last year. The powerful functionality of Munin is that with relatively easy creation of plugins it draws neat graphs. In case that you prepare a plugin to monitor something connected with the node, then Munin automatically draws neat graphs.There is possibility to find plugins only, when we are not satisfied with the ones that are already provided with Munin. Munin can also send notification alerts, but it is not its primary focus. Munin is a very powerful tool in situation when we are debugging performance problems that are complex.

In a situation when a performance problem occurs, the problem is usually simple to point out, such as web server is not processing as many requests per second as it should or database server is not processing as many queries as it did before.

The biggest challenge is to find the cause of the problem, for example to find out what the reason is for the decrease in performance, before we can start dealing with that. It makes research of what is the problem much easier, when we have access to the graphs to correlate the things that could cause the problem. It is also very helpful for planning purchase of, for instance, new hard drives.[38]

4 Practical part

4.1 Reasons for preparation of monitoring scripts

For network monitoring or in our case for monitoring of system resources already exist wide spread of solutions. There are very complex and robust programs as Nagios, that is suitable for monitoring of hundreds of servers and big corporate networks. There is widely accepted opinion that Nagios has very complicated configuration and instalation, which is one of the reason for popularity of other solutions. Among others very popular is also Cacti, that has wide spread of settings and parameters. When we decide for small solutions for smaller network there is very helpful tool, called RRDtool. That uses Round Robin database as back-end and at same time it has functionality as front-end to draw graphs. With RRDtool we can prepare scripts sufficient for monitoring of smaller network with advantages of preparation for our exact needs, that we can save time for configuration.

4.2 Preparation of the system for our router

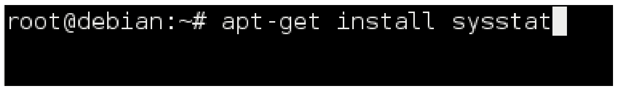
For the purposes of testing we used at the beganning an old computer with configuration such as processor Pentium II with frequency 450 MHz, RAM memory with size 256 MB and hard disk with 4327 MB. It is obviously an obsolete computer, but for purposes of networking in community networks such as CZFree.Net, computers with similar configuration really used to be deployed . Later on was also used for testing notebook HP Compaq 6720s with configuration such as processor Intel(R) Pentium(R) Dual CPU T2410 with frequency 2.00 GHz, 2GB of RAM memory and 250 GB hard drive. Nowadays community networks use more small integrated routers from MikroTik, but for purposes of testing our hardware is sufficient.

4.3 GNU/Linux Debian installation

While consideration of operating system we choose GNU/Linux Debian stable release 7.2.0. Installation consists of several mostly intuitive steps, in the first step we choose Install, in its context it stays for installation in text mode. It is followed by steps, where we select the language, keyboard layout. Then there is a part where hardware components are detected. In the following part we configure the network. Among other things, we set password for the root user and we may create also other user accounts. After that there is part with disk partitioning, where we create partition / called root for the system, also swap partition for virtual memory. Especially for Linux servers there may be several partitions as /var for web servers etc. Then follows installation of base system, followed by configuration of package manager where we choose mirrors with Debian packages. As one of last steps we specify software selection, where we do not need Debian desktop environment, so we remove such option, but we need web server so we choose that and optionally we may choose other components. As the last step we install the GRUB Bootloader to MBR. After rebooting the system we have completed the installation.

4.3.1 Installation of sysstat

For gathering information about CPU Utilization we need iostat, that is part of program sysstat that we can install in following way

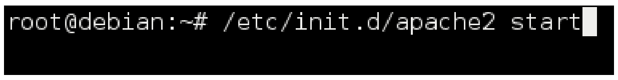


```
root@debian:~# apt-get install sysstat
```

Ilustrace 6: sysstat installation

4.3.2 Apache deployment

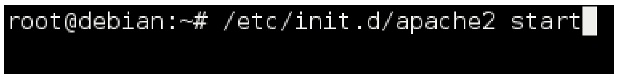
We need to present measured data, for this purpose we may deploy well known web server Apache. If we have not installed Apache before we may do that as follows



```
root@debian:~# /etc/init.d/apache2 start
```

Ilustrace 7: Apache installation

After the configuration of domain name we may start the service as follows

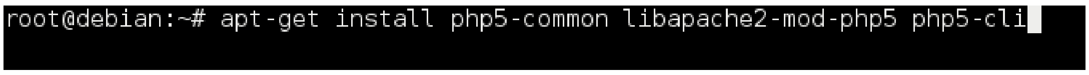


```
root@debian:~# /etc/init.d/apache2 start
```

Ilustrace 8: Start of Apache

4.3.3 PHP Installation

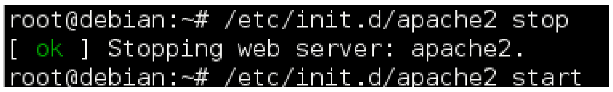
We also use PHP, if it was not installed before we may install that as follows



```
root@debian:~# apt-get install php5-common libapache2-mod-php5 php5-cli
```

Ilustrace 9: PHP installation

After the installation we need to restart the Apache.

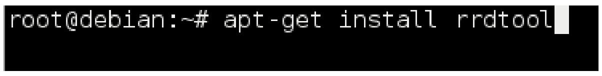


```
root@debian:~# /etc/init.d/apache2 stop
[ ok ] Stopping web server: apache2.
root@debian:~# /etc/init.d/apache2 start
```

Ilustrace 10: Apache restart

4.3.4 RRDtool Installations

We also need to install rrdtool. We may see several articles, where they deploy Perl scripts and also install librrds-perl because of that, but it is not our case.



```
root@debian:~# apt-get install rrdtool
```

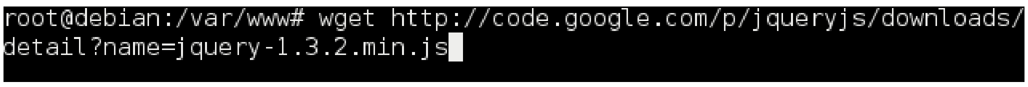
Ilustrace 11: Installation of RRDtool

4.3.5 Apache deployment

RRDtool is the OpenSource industry standard, high performance data logging and graphing system for time series data. RRDtool can be easily integrated in shell scripts, perl, python, ruby, lua or tcl applications.

4.3.6 jQuery deployment

First we need to download JQuerry file, that we can do as follows.



```
root@debian:/var/www# wget http://code.google.com/p/jqueryjs/downloads/
detail?name=jquery-1.3.2.min.js
```

Ilustrace 12: jQuerry deployment

Then in our file such as index.php or index.html we include into the header following code

<script type="text/javascript" src="jquery-1.3.2.min.js"></script>

As a result JQuerry is installed. [42]

4.4 Creation of Round Robin Archives (RRA)

In order to work with RRA, we need to create the archives. We can handle this procedure in script, but this is important step, that is why there is deeper explanation in this section.

For the archive about disk usage

```
rrdtool create /var/www/rrd/disk.rrd --step 300 DS:disk:GAUGE:600:0:U
RRA:MAX:0.5:1:105120
```

With command above we create archive with specific parameters.

- --step 300 stands for how often (in seconds) we are supposed to store the data
- DS stands for data source, it has similar meaning as column in SQL database
- GAUGE is type of the value. We can in general choose from GAUGE, that is suitable for temperature or value that can be measured at a time or COUNTER, that is suitable for for instance network traffic, where is increasing number of packets sent.
- 600 is the heartbeat, which is time it waits for the value, if no value is given to the archive in that time, then unknown value is stored.
- 0 is minimal value that is acceptable
- U stands for second boundary, that observed value should be in. When we do not know expected range of values in advance, it is better to use Unknown.
- MAX is selected consolidated function, meaning that we keep the highest observed value in given time. Other consolidation functions are MIN, that keeps lowest value, LAST that keep newest value in the set and AVERAGE that count average.
- 0.5 means that 50 percents of the data has to be valid in order to store them to archive. If there is less that 50 percents valid then it stores unknown value.
- 1 means storing every round (it can take more entries according the specification)
- 105120 means total number of samples to store, we got the number from calculation 12 * 24 * 365 = 105120 meaning 12 observations per hour multiplied by 24 for hours and multiplied by 365 for days.

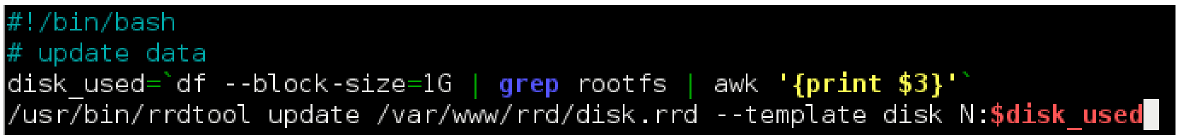
While observing Processor load we observe load caused by Applications (user), system kernel (system), etc.

```
rrdtool create /var/www/rrd/user.rrd --step 300 DS:user:GAUGE:600:0:U RRA:MAX:0.5:1:105120
rrdtool create /var/www/rrd/nice.rrd --step 300 DS:nice:GAUGE:600:0:U RRA:MAX:0.5:1:105120
rrdtool create /var/www/rrd/system.rrd --step 300 DS:system:GAUGE:600:0:U RRA:MAX:0.5:1:105120
rrdtool create /var/www/rrd/iowait.rrd --step 300 DS:iowait:GAUGE:600:0:U RRA:MAX:0.5:1:105120
rrdtool create /var/www/rrd/disk.rrd --step 300 DS:disk:GAUGE:600:0:U RRA:MAX:0.5:1:105120
rrdtool create /var/www/rrd/mem.rrd --step 300 DS:mem:GAUGE:600:0:U RRA:MAX:0.5:1:105120
rrdtool create /var/www/rrd/rx.rrd --step 300 DS:rx:COUNTER:600:0:U RRA:MAX:0.5:1:105120
rrdtool create /var/www/rrd/tx.rrd --step 300 DS:tx:COUNTER:600:0:U RRA:MAX:0.5:1:105120
```

4.5 Obtaining the data

4.5.1 Disk usage

In order to get the data, we have scripts that we use to update Round Robin database. For disk usage there is the following part of script.



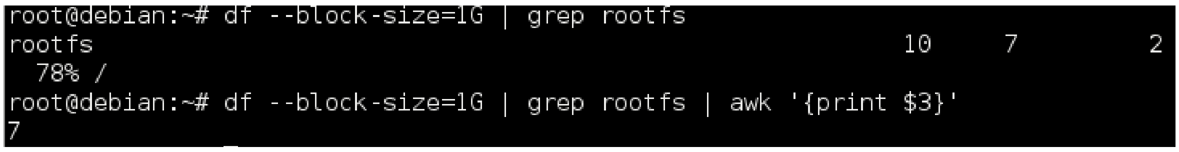
```
#!/bin/bash
# update data
disk_used=`df --block-size=1G | grep rootfs | awk '{print $3}'`
/usr/bin/rrdtool update /var/www/rrd/disk.rrd --template disk N:$disk_used
```

Illustration 13: Update data about disk usage

So in order to get observed value we call command df (which is abbreviation for disk free) with option –block-size=1G, that specify the clock size, there could also be used 1M and output would be in megabytes. Then we use pipe (written as character |), that allows us to use output of one command as input for another.

The following command grep searches for lines that match the string that we search for. In this case it is rootfs, which is root partition with the system.

After next pipe we use awk, which is a pattern-directed processing language. With the use of that we take the third item in the line



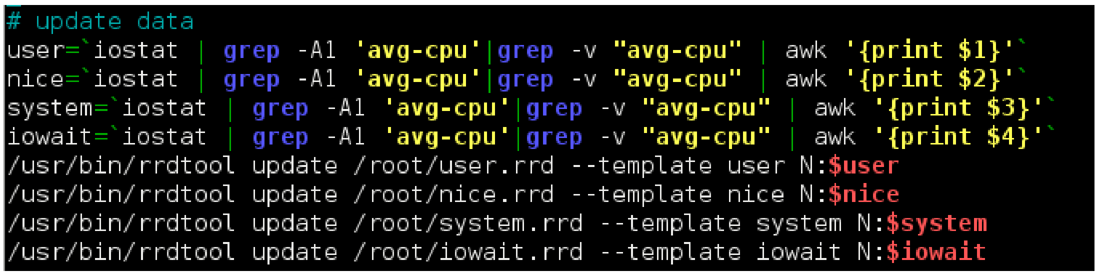
```
root@debian:~# df --block-size=1G | grep rootfs
rootfs                                                      10     7         2
  78% /
root@debian:~# df --block-size=1G | grep rootfs | awk '{print $3}'
7
```

Illustration 14: Explanation of df

With the last command cut, we take only the number that we need to store in Round Robin database.

4.5.2 Processor load

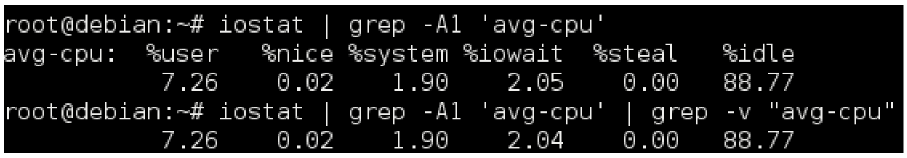
Processor load can be divided into several categories that are good to be separated in the graph. There is value user, which represents CPU utilization that was observed while executing at user level, that means applications. There is also value system, which represents utilization in system mode, that is by system kernel. Next value is iowait, which stands for percentage of time during which CPU was idle, because of for instance outstanding disk I/O request. Last presented value is nice. Here we should mention that by nice we mean processes with particular priority. To make that clear, there are processes that need more CPU time than the others, so nice becames useful when several processes demand more resources than CPU can provide. Nice in the graph that represents CPU utilization at user level with nice priority. [43]



```
# update data
user=`iostat | grep -A1 'avg-cpu'|grep -v "avg-cpu" | awk '{print $1}'`
nice=`iostat | grep -A1 'avg-cpu'|grep -v "avg-cpu" | awk '{print $2}'`
system=`iostat | grep -A1 'avg-cpu'|grep -v "avg-cpu" | awk '{print $3}'`
iowait=`iostat | grep -A1 'avg-cpu'|grep -v "avg-cpu" | awk '{print $4}'`
/usr/bin/rrdtool update /root/user.rrd --template user N:$user
/usr/bin/rrdtool update /root/nice.rrd --template nice N:$nice
/usr/bin/rrdtool update /root/system.rrd --template system N:$system
/usr/bin/rrdtool update /root/iowait.rrd --template iowait N:$iowait
```

Ilustrace 15: iostat commands to update data

We use program iostat, which reports CPU statistics and input/output statistics for devices and partitions.



```
root@debian:~# iostat | grep -A1 'avg-cpu'
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           7.26    0.02    1.90    2.05    0.00   88.77
root@debian:~# iostat | grep -A1 'avg-cpu' | grep -v "avg-cpu"
           7.26    0.02    1.90    2.04    0.00   88.77
```

Ilustrace 16: Explanation of iostat

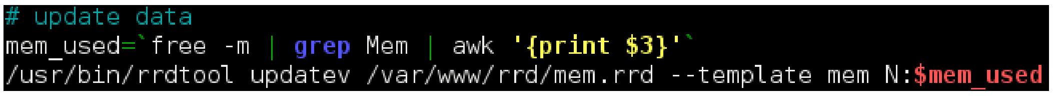
Then we use twice grep command, at first with option -A1 so it selects the line that matches the criterion and one more under that. And then with option -v, so we get the line with numbers, not with description.

As next step we use awk to choose first value from the row.

For other values, such as system, iowait there is basically the same procedure, where we choose different value in the last step.

4.5.3 Memory usage

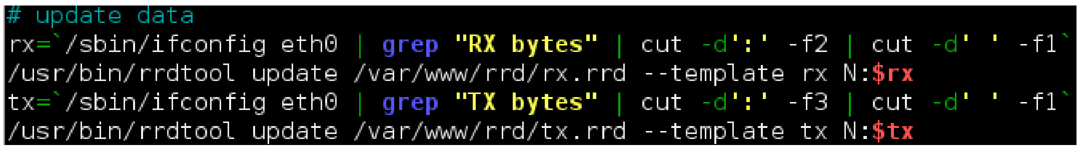
Similarly as in previous cases, here we call command free with parameter “m”. Output is taken as input for grep command and result is proceeded with awk, that selects third item (column) on the line. Result is saved into RRA. [44]



```
# update data
mem_used=`free -m | grep Mem | awk '{print $3}'`
/usr/bin/rrdtool updatev /var/www/rrd/mem.rrd --template mem N:$mem_used
```

Ilustrace 17: Update data about memory

4.5.4 Network traffic

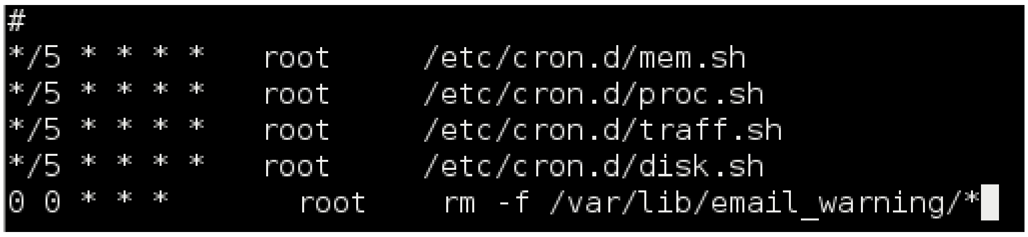


```
# update data
rx=`/sbin/ifconfig eth0 | grep "RX bytes" | cut -d':' -f2 | cut -d' ' -f1`
/usr/bin/rrdtool update /var/www/rrd/rx.rrd --template rx N:$rx
tx=`/sbin/ifconfig eth0 | grep "TX bytes" | cut -d':' -f3 | cut -d' ' -f1`
/usr/bin/rrdtool update /var/www/rrd/tx.rrd --template tx N:$tx
```

Ilustrace 18: Update data about traffic

When we consider network traffic, we are interested in both received and transmitted traffic. In this script is observed interface eth0, but we may easily change that and observe other interfaces too. In the script we call command ifconfig, that shows details about interface. Output of command ifconfig is taken as input for grep, that choses the line that contain searched string and output is proceeded with cut command. Automating Data Collection

We have described above the way how we get the values to save to the database. In Unix-like operating systems there is a time based job scheduler called Cron. We can use for editing either command crontab -e to modify list of tasks of Cron or we can modify the file located at /etc/crontab/.



```
#
*/5 * * * *   root     /etc/cron.d/mem.sh
*/5 * * * *   root     /etc/cron.d/proc.sh
*/5 * * * *   root     /etc/cron.d/traff.sh
*/5 * * * *   root     /etc/cron.d/disk.sh
0 0 * * *       root    rm -f /var/lib/email_warning/*
```

Illustration 19: Crontab

In each line there are 5 symbols that influence how often the task is going to be performed. First position stands for minutes and as we use there */5, with a meaning that those tasks are going to be performed every 5 minutes. In the last line we can see 0 0 * * *, it has a meaning that the task should be performed once a day at midnight. This task is used in connection to sending notification emails. We do not want to send email with alert to administrator every 5 minutes, it is sufficient to do that once a day. Script that is used for notifications of administrator creates a file every time it sends an email to avoid further emails caused by same problem.

4.6 Use of Inline DateTimePicker

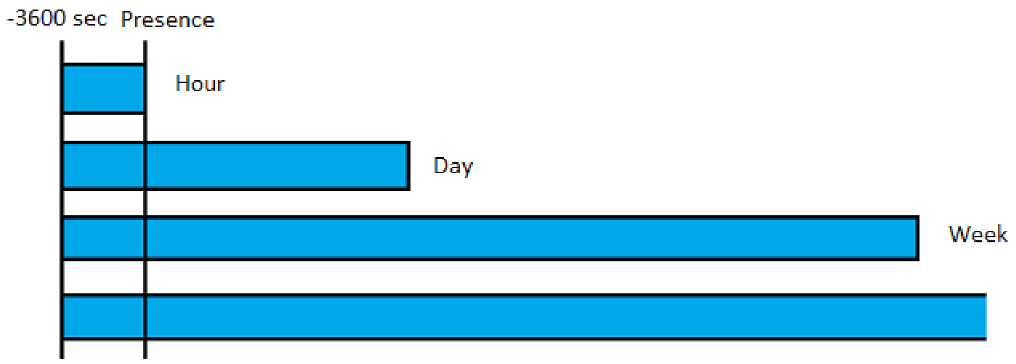
When the network administrator is dealing with a problem, it is very helpful to see performance of observed value for a longer time period. According to the settings that were chosen when RRA was created, it can store the data for period of, for instance, two years. In order to show previous values we use DateTimePicker that was developed by Chupurnov Valeriy and is available at http://plugins.jquery.com/datetimepicker/

4.7 Graphing the data

We have described above the way how we get the values to save to the database. To store data into database is considered to be back-end function, there is special thing about RRDtool, that it can also draw the graphs, and in such a way behaves as front-end. Mentioned functionality is considered to be quite unique.

4.7.1 Disk usage

As it was mentioned before, RRDtool has also functionality as front-end for presenting the observed values. For the graph we specify, that there should be created an image with width 500 pixels and height 150 pixels and file format should be PNG. Then we have several mostly intuitive parameters such as vertical-label, that draws vertical label in the graph. With use of parameter –color we specify desired colors to be used for drawing grid, back, also font color etc. With use of DEF we define the variable to be displayed. With use of AREA, we specify that RRDtool will draw area rather than only a line to display the values. Using the command LINE1 we also specify width of line, because there are also commands as LINE2, LINE3 etc. The command GPRINT is similar to PRINT, but is intended to print into the graph. In the script the function mktime is used, which returns Unix timestamp, corresponding to given argument. The timestamp is the number of seconds between Unix epoch, that is January 1 1970 0:00:00 and the time that we specified. In order to explain how graphs in time are intended to be displayed the following image is added.



Ilustrace 20: Explanation of timing in graphs

```
<?php

$ret=sscanf($_REQUEST['time'], "%d/%d/%d %d:%d", $year,$month,$day,$hour,$minute);
if($ret == 5)
{
        $ts = mktime($hour, $minute, 0, $month, $day, $year);
}
else $ts=time()-3600;
create_graph("png/diskh.png", $ts, $ts+3600, "Disk space used (last hour)");
create_graph("png/diskd.png", $ts, $ts+86400, "Daily disk space used");
create_graph("png/diskw.png", $ts, $ts+604800, "Weekly disk space used");
create_graph("png/diskm.png", $ts, $ts+2592000, "Monthly disk space used");
create_graph("png/disky.png", $ts, $ts+31536000, "Yearly disk space used");
```

```
echo "<table>";
echo "<tr><td>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts - 3600))."\">";
echo "<img src='png/left.png' class='left'>";
echo "</a>";
echo "<img src='png/diskh.png' alt='Generated RRD image'>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts + 3600))."\">";
echo "<img src='png/right.png' class='right'>";
echo "</a>";
echo "</tr></td>";
echo "<tr><td>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts - 86400))."\">";
echo "<img src='png/left.png' class='left'>";
echo "</a>";
echo "<img src='png/diskd.png' alt='Generated RRD image'>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts + 86400))."\">";
echo "<img src='png/right.png' class='right'>";
echo "</a>";
echo "</td></tr>";
echo "<tr><td>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts - 604800))."\">";
echo "<img src='png/left.png' class='left'>";
echo "</a>";
echo "<img src='png/diskw.png' alt='Generated RRD image'>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts + 604800))."\">";
echo "<img src='png/right.png' class='right'>";
echo "</a>";
echo "</td></tr>";
echo "<tr><td>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts - 2592000))."\">";
echo "<img src='png/left.png' class='left'>";
echo "</a>";
echo "<img src='png/diskm.png' alt='Generated RRD image'>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts + 2592000))."\">";
echo "<img src='png/right.png' class='right'>";
echo "</a>";
echo "</td><td>";
echo "<tr><td>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts - 31536000))."\">";
echo "<img src='png/left.png' class='left'>";
echo "</a>";
echo "<img src='png/disky.png' alt='Generated RRD image'>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts + 31536000))."\">";
echo "<img src='png/right.png' class='right'>";
echo "</a>";
echo "</td></tr>";
echo "</table>";
exit;

function create_graph($output, $start, $end, $title) {
  $options = array(
"-w",
"500",
"-h",
"150",
"-a",
"PNG",
   "--slope-mode",
   "--start", $start,
   "--end", $end,
   "--title=$title",
   "--vertical-label=Disk space used",
   "--lower=0",
   "--color", "BACK#363636",
   "--color", "CANVAS#000000",
   "--color", "GRID#999999",
   "--color", "MGRID#B5B5B5",
   "--color", "FONT#CCCCCC",
   "DEF:disk=/var/www/rrd/disk.rrd:disk:MAX",
   "AREA:disk#FFD700",
   "LINE1:disk#FFD700:Disk space used (GB)",
   "GPRINT:disk:LAST:Last\: %5.2lf",
   "GPRINT:disk:AVERAGE:Avg\: %5.2lf",
   "GPRINT:disk:MAX:Max\: %5.2lf",
   "GPRINT:disk:MIN:Min\: %5.2lf\t\t\t",
  );

 $ret = rrd_graph($output, $options);
 if (! $ret) {
   echo "<b>Graph error: </b>".rrd_error()."\n";
 }
}

?>
```

4.7.2 Memory usage

Now we look at creation of graphs for system memory (RAM). We have described above several parameters that are also used in this graph. There are data from RRA mem.rrd, which is stored at /var/www/rrd/mem.rrd. In the graphs for a day period etc we can see observations only for particular times, because this printscreen was from testing on a notebook, which was turned off, for example, during the nights. With use of GPRINT we also show in the graph min, max, last and average values as they are observed or calculated while the observation.

```
$ret=sscanf($_REQUEST['time'], "%d/%d/%d %d:%d", $year,$month,$day,$hour,$minute);
if($ret == 5)
{
        $ts = mktime($hour, $minute, 0, $month, $day, $year);
}
```

```
else $ts=time()-3600;
create_graph("png/memh.png", $ts, $ts+3600, "Memory used - last hour");
create_graph("png/memd.png", $ts, $ts+86400, "Memory used - day");
create_graph("png/memw.png", $ts, $ts+604800, "Memory used - week");
create_graph("png/memm.png", $ts, $ts+2592000, "Memory used - month");
create_graph("png/memy.png", $ts, $ts+31536000, "Memory used - year");

function create_graph($output, $start, $end, $title) {
  $options = array(
"-w",
"500",
"-h",
"150",
"-a",
"PNG",
    "--slope-mode",
    "--start", $start,
    "--end", $end,
    "--title=$title",
    "--vertical-label=Memory used",
    "--lower=0",
    "--color", "BACK#363636",
    "--color", "CANVAS#000000",
    "--color", "GRID#999999",
    "--color", "MGRID#B5B5B5",
    "--color", "FONT#CCCCCC",
    "DEF:memory=/var/www/rrd/mem.rrd:mem:MAX",
    "AREA:memory#FFD700",
    "LINE1:memory#FFD700:Memory used (MB)",
    "GPRINT:memory:LAST:Last\: %5.2lf",
    "GPRINT:memory:AVERAGE:Avg\: %5.2lf",
    "GPRINT:memory:MAX:Max\: %5.2lf",
    "GPRINT:memory:MIN:Min\: %5.2lf\t\t\t",
  );

  $ret = rrd_graph($output, $options);
  if (! $ret) {
    echo "<b>Graph error: </b>".rrd_error()."\n";
  }
}
```

4.7.3 Processor load

Now we look at part of the script for graphing processor load. Here we can see the use of CDEF, which is intended for calculations with observed values. CDEF works with DEFs that define the variables, but we can not do modifications to DEFs, that is why CDEF is used. When calculations for CDEF are proceeded, the so called RPN (Reverse Polish Notation) math is used. Values are graphed as AREA and we use parameter STACK, that places the values on each other, so in our case load in user mode is stacked on iowait and it is on load by system (kernel) and so on.

```
$ret=sscanf($_REQUEST['time'], "%d/%d/%d %d:%d", $year,$month,$day,$hour,$minute);
if($ret == 5)
{
        $ts = mktime($hour, $minute, 0, $month, $day, $year);
}
else $ts=time()-3600;
create_graph("png/proch.png", $ts, $ts+3600, "Processor load - hour");
create_graph("png/procd.png", $ts, $ts+86400, "Processor load - day");
create_graph("png/procw.png", $ts, $ts+604800, "Processor load - week");
create_graph("png/procm.png", $ts, $ts+2592000, "Processor load - month");
create_graph("png/procy.png", $ts, $ts+31536000, "Processor load - year");

function create_graph($output, $start, $end, $title) {
  $options = array(
"-w",
"500",
"-h",
"150",
"-a",
"PNG",
    "--slope-mode",
    "--start", $start,
    "--end", $end,
    "--title=$title",
    "--vertical-label=Processor load",
    "--lower=0",
    "--color", "BACK#363636",
    "--color", "CANVAS#000000",
    "--color", "GRID#999999",
    "--color", "MGRID#B5B5B5",
    "--color", "FONT#CCCCCC",
    "DEF:user=/var/www/rrd/user.rrd:user:MAX",
    "DEF:nice=/var/www/rrd/nice.rrd:nice:MAX",
    "DEF:system=/var/www/rrd/system.rrd:system:MAX",
    "DEF:iowait=/var/www/rrd/iowait.rrd:iowait:MAX",
    "CDEF:utilization_pct=user,system,iowait,+,+,user,system,nice,iowait,
+,+,+,/,100,*",
    "AREA:user#FFD700:Processor load user(%):STACK",
    "GPRINT:user:LAST:Last\: %5.2lf",
    "GPRINT:user:AVERAGE:Avg\: %5.2lf",
    "GPRINT:user:MAX:Max\: %5.2lf",
    "GPRINT:user:MIN:Min\: %5.2lf\\t\\t\\t\\n",
    "AREA:nice#FF0000:Processor nice (%):STACK",
    "GPRINT:nice:LAST:Last\: %5.2lf",
    "GPRINT:nice:AVERAGE:Avg\: %5.2lf",
    "GPRINT:nice:MAX:Max\: %5.2lf",
    "GPRINT:nice:MIN:Min\: %5.2lf\\t\\t\\t\\n",
    "AREA:system#1E90FF:Processor system (%):STACK",
    "GPRINT:system:LAST:Last\: %5.2lf",
    "GPRINT:system:AVERAGE:Avg\: %5.2lf",
    "GPRINT:system:MAX:Max\: %5.2lf",
    "GPRINT:system:MIN:Min\: %5.2lf\\t\\t\\t\\n",
    "AREA:iowait#00FF00:Processor iowait (%):STACK",
    "GPRINT:iowait:LAST:Last\: %5.2lf",
    "GPRINT:iowait:AVERAGE:Avg\: %5.2lf",
    "GPRINT:iowait:MAX:Max\: %5.2lf",
    "GPRINT:iowait:MIN:Min\: %5.2lf\\t\\t\\t\\n",
    );

  $ret = rrd_graph($output, $options);
  if (! $ret) {
    echo "<b>Graph error: </b>".rrd_error()."\n";
  }
}

?>
```

4.7.4 Network traffic

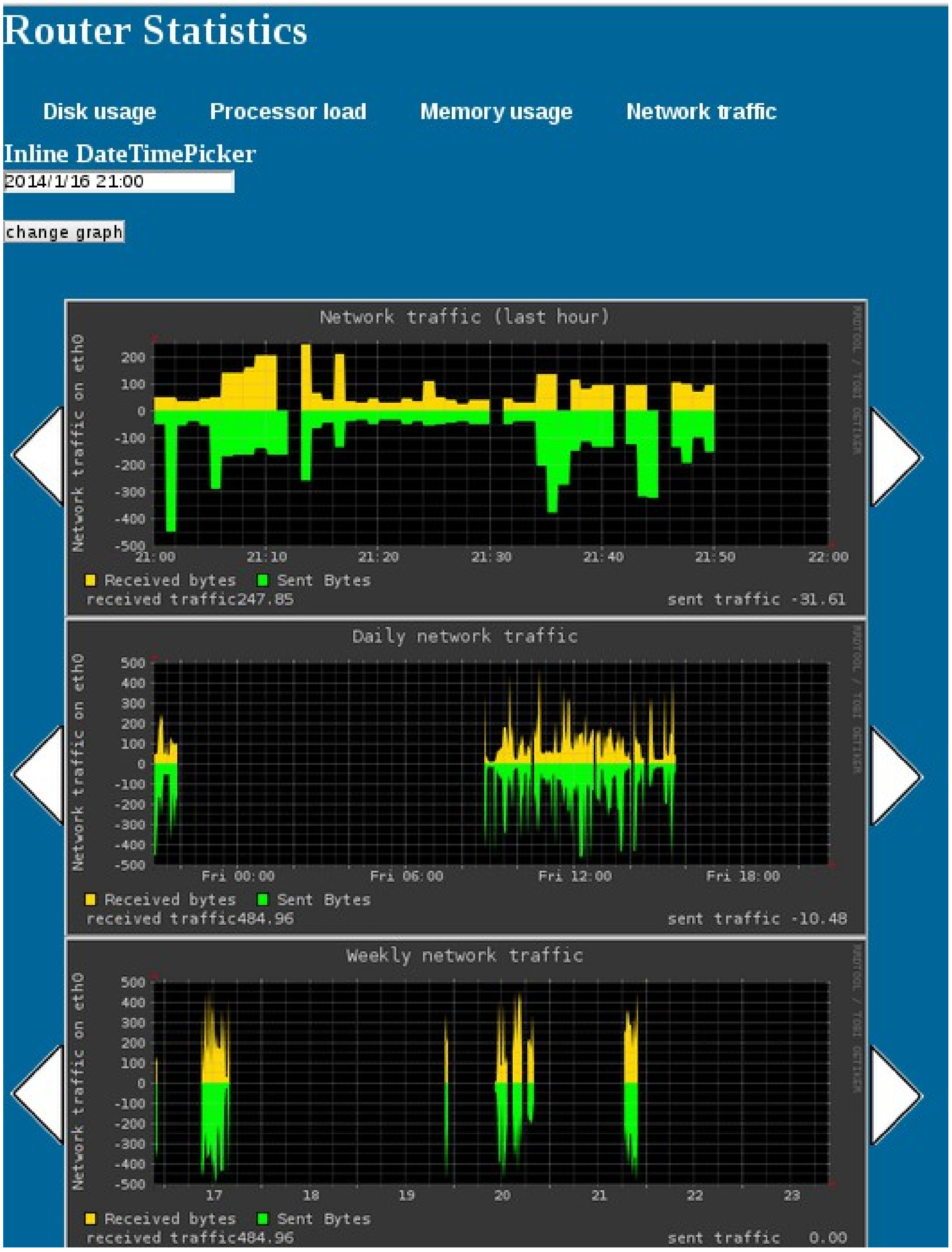
In the script for graphing network traffic we also use CDEF as calculation in order to display transmitted traffic as negative values. Then, when we have Received traffic as positive values, it may seem more logical for administrator. Similarly as in previous examples there are shown values printed into graph with GPRINT, such as average, maximum, minimum. We should also see that in this case we use COUNTER rather than GAUGE as type of variable. GAUGE is suitable for, for instance, temperature, disk load etc, simply said, values that make sense to measure once at that time. With COUNTER situation is different, it is suitable for network traffic as there is still increasing number of packets that passed the interface.

```
$ret=sscanf($_REQUEST['time'], "%d/%d/%d %d:%d", $year,$month,$day,$hour,$minute);
if($ret == 5)
{
        $ts = mktime($hour, $minute, 0, $month, $day, $year);
}
else $ts=time()-3600;
create_graph("png/trafh.png", $ts, $ts+3600, "Network traffic (last hour)");
create_graph("png/trafd.png", $ts, $ts+86400, "Daily network traffic");
create_graph("png/trafw.png", $ts, $ts+604800, "Weekly network traffic");
create_graph("png/trafm.png", $ts, $ts+2592000, "Monthly network traffic");
create_graph("png/trafy.png", $ts, $ts+31536000, "Yearly network traffic");

function create_graph($output, $start, $end, $title) {
  $options = array(
"-w",
"500",
"-h",
"150",
"-a",
"PNG",
   "--slope-mode",
   "--start", $start,
   "--end", $end,
   "--title=$title",
   "--vertical-label=Network traffic on eth0",
   "--lower=0",
   "--color", "BACK#363636",
   "--color", "CANVAS#000000",
   "--color", "GRID#999999",
   "--color", "MGRID#B5B5B5",
   "--color", "FONT#CCCCCC",
   "DEF:received=/var/www/rrd/rx.rrd:rx:MAX",
   "DEF:sent=/var/www/rrd/tx.rrd:tx:MAX",
   "CDEF:sentm=0,sent,-",
   "AREA:received#FFD700:Received bytes",
   "AREA:sentm#00FF00:Sent Bytes",
   "COMMENT:\\n",
   "GPRINT:received:MAX:received traffic%6.2lf",
   "COMMENT: ",
   "GPRINT:sentm:MAX:sent traffic %6.2lf",
  );

  $ret = rrd_graph($output, $options);
  if (! $ret) {
   echo "<b>Graph error: </b>".rrd_error()."\n";
  }
}
```

4.8 Graphing the observed values

In this chapter we can see graphs, that RRDtool prepared according to observed values. For each observed value there are is five graphs on the web page, for hour, day, week, month and year. Because of too big size of the printscreens there are examples usually with only graphs for hour and day. In real deployment router should be running non-stop, but in our testing environment (implemented at notebook) there are observations only for parts of days and for times, when the notebook was not running there is an unknown value in the graph.

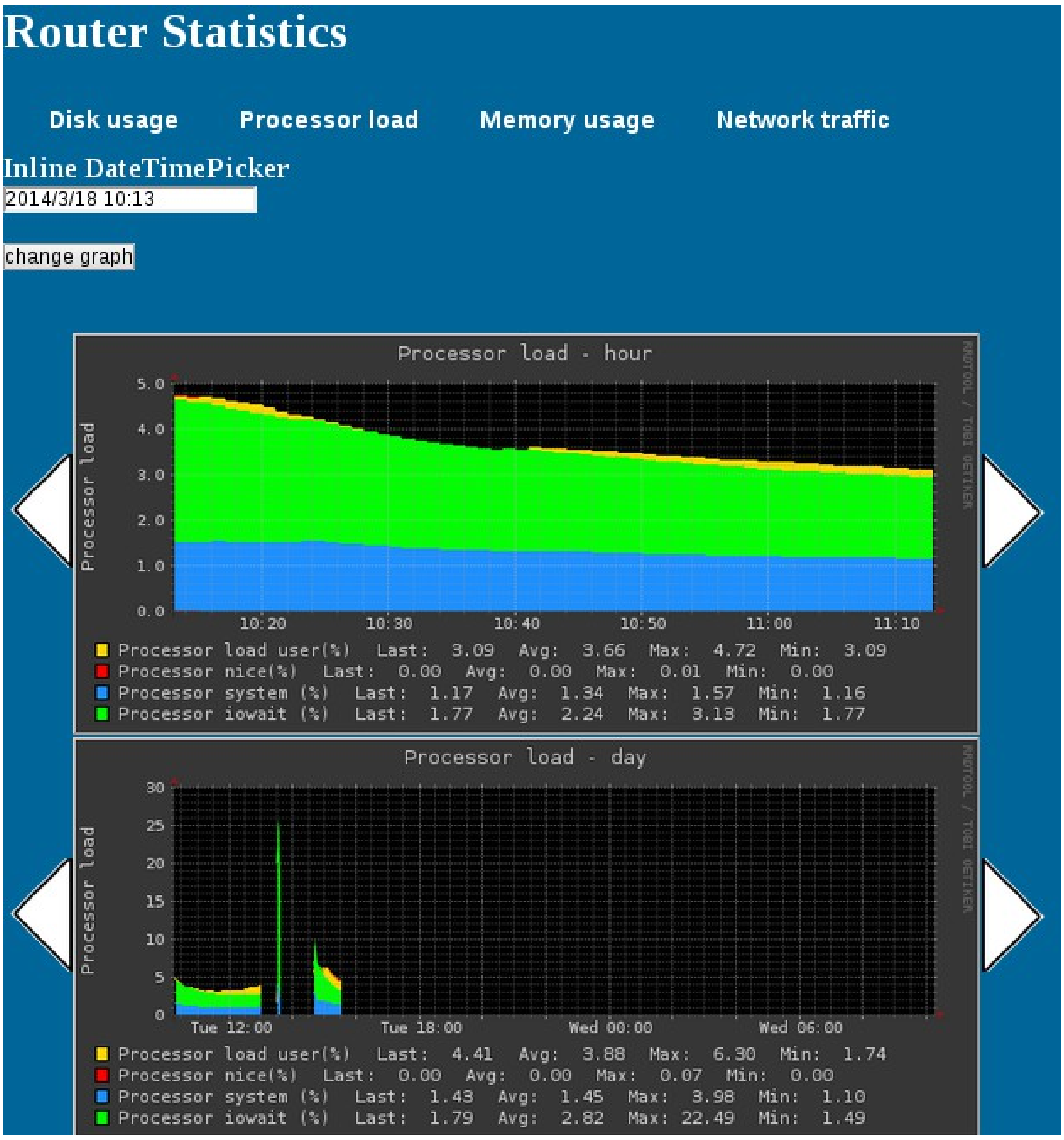
4.8.1 Network traffic



Ilustrace 21: Network traffic graphs

When we consider graphing of network traffic, administrator must create Round Robin Archive wisely, because the highest acceptable value is usually set, for instance, on 499. But if the observed value is higher, then observation is considered as unknown. In case when we set such a value, then we easily get unknown values even when a user, for example, watches a video that is streamed from the Internet. This problem can be solved with settings of values that we store, in case of disk space we can use --block-size parameter. Another important fact to consider is that routers usually have more network interfaces, so we should observe all of them, except Loopback, which is a special virtual network interface.

4.8.2 Processor load



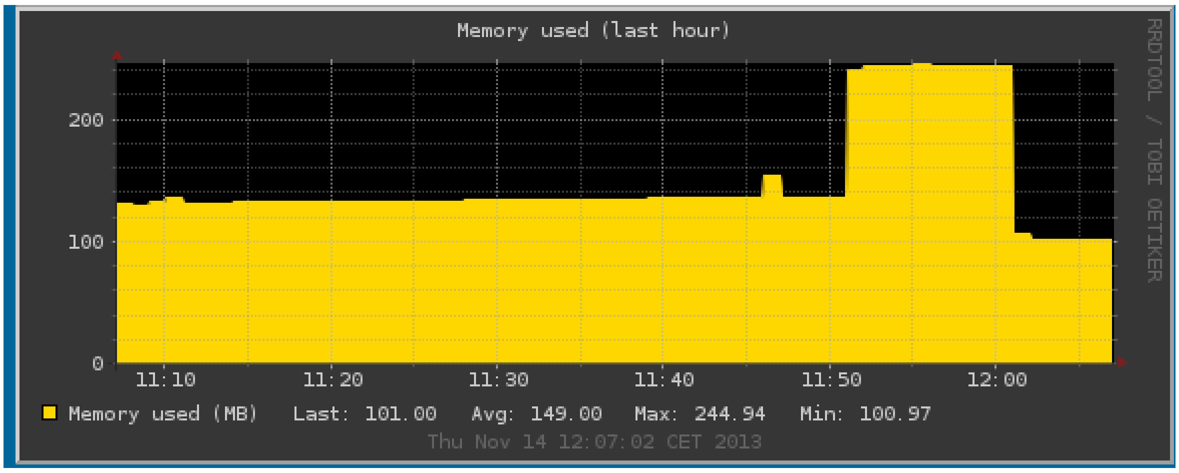
```
Processor load - hour
🟨 Processor load user(%)  Last:  3.09  Avg:  3.66  Max:  4.72  Min:  3.09
🟥 Processor nice(%)  Last:  0.00  Avg:  0.00  Max:  0.01  Min:  0.00
🟦 Processor system (%)  Last:  1.17  Avg:  1.34  Max:  1.57  Min:  1.16
🟩 Processor iowait (%)  Last:  1.77  Avg:  2.24  Max:  3.13  Min:  1.77

Processor load - day
🟨 Processor load user(%)  Last:  4.41  Avg:  3.88  Max:  6.30  Min:  1.74
🟥 Processor nice(%)  Last:  0.00  Avg:  0.00  Max:  0.07  Min:  0.00
🟦 Processor system (%)  Last:  1.43  Avg:  1.45  Max:  3.98  Min:  1.10
🟩 Processor iowait (%)  Last:  1.79  Avg:  2.82  Max: 22.49  Min:  1.49
```

Ilustrace 22: Processor load graphs

Each observed value is at the web page five graphs, for hour, day, week, month and year. In this print screen only hour and day graphs are cutted. We can see that processor load is caused by several different modes, such as system mode, that basically represent use by system kernel, then user mode, which are for instance

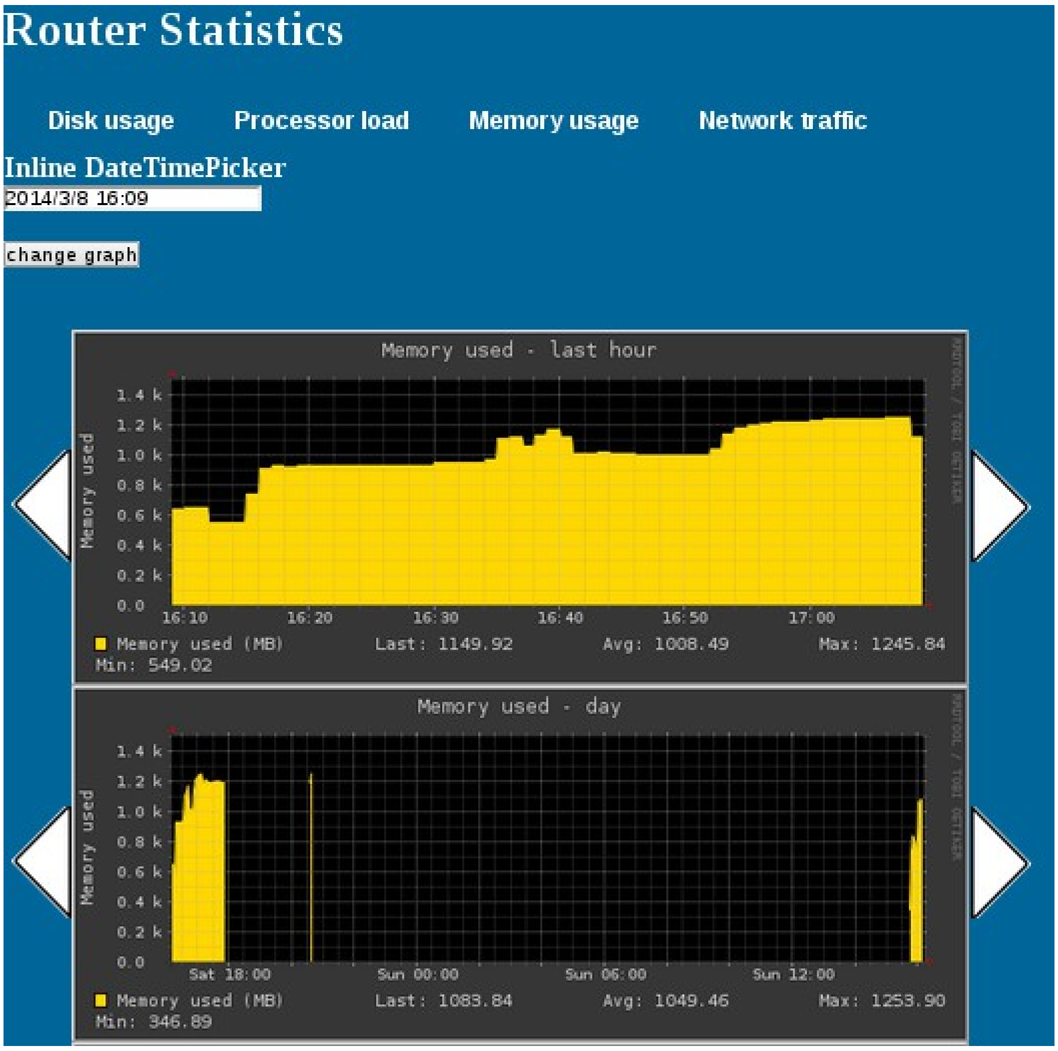
4.8.3 Memory used

When the script is graphing memory, there is considerable parameter –upper-limit=, where we can specify the total size of memory. It is very important for realization of how much memory is actually used, because if we do not specify this parameter, than the highest observation in given period appears as maximal value. For example if only small part of memory is used, then it still looks like, all the memory is used. To present explained situation part of the graph from previous development is presented.



Ilustrace 23: Memory used explanation

We can see, that only comparably small part of memory was used, but in the graph it appears differently. When we consider use of memory in GNU/Linux we could also think of SWAP partition and graph the usage of it. Another thing to consider is possibility of graphical separation of memory used for buffers, caches etc. According the opinion of author current setting is sufficient.


Router Statistics

Disk usage Processor load Memory usage Network traffic

Inline DateTimePicker

2014/3/8 16:09

change graph

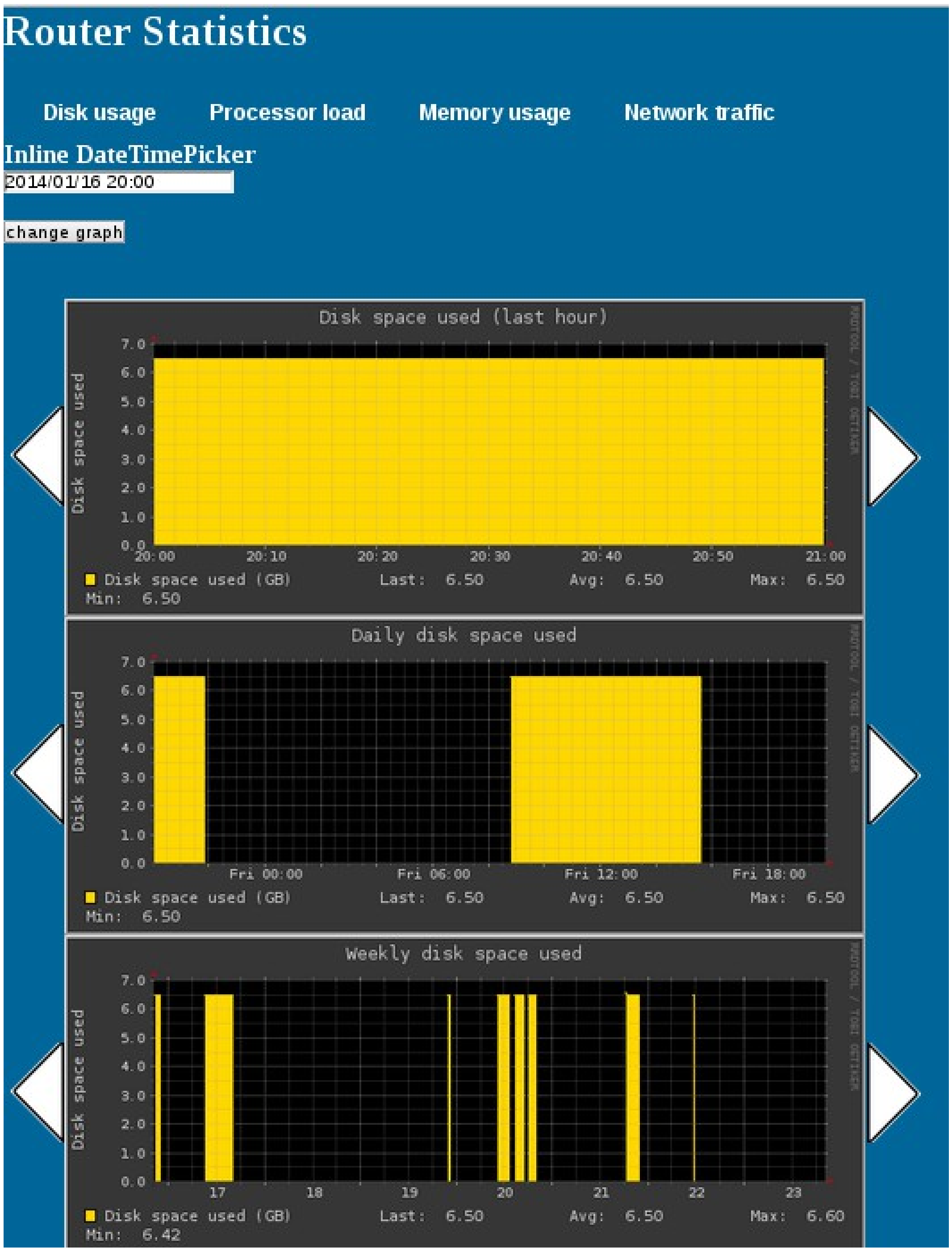
■ Memory used (MB) Last: 1149.92 Avg: 1008.49 Max: 1245.84
Min: 549.02

■ Memory used (MB) Last: 1083.84 Avg: 1049.46 Max: 1253.90
Min: 346.89

Ilustrace 24: Memory used graph

4.8.4 Disk space used

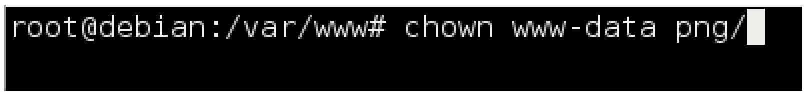
When we think of monitoring hard disk, there could be in general monitoring of number of write/read requests per second, but in case of router this information is not really important for us. Or if we would be dealing with server, there could be several partitions such as /usr or /var, so in such case we should graph the usage of all partitions. For graphing disk space usage, there is considerable parameter –upper-limit=, which can also help us to have more clearly arranged values, as it was explained in previous chapter.


Router Statistics
Disk usage
Processor load
Memory usage
Network traffic
Inline DateTimePicker
2014/01/16 20:00
change graph
Disk space used (last hour)
Disk space used
■ Disk space used (GB) Last: 6.50 Avg: 6.50 Max: 6.50 Min: 6.50
Daily disk space used
■ Disk space used (GB) Last: 6.50 Avg: 6.50 Max: 6.50 Min: 6.50
Weekly disk space used
■ Disk space used (GB) Last: 6.50 Avg: 6.50 Max: 6.60 Min: 6.42

Ilustrace 25: Disk spaced used graphs

4.9 Setting permissions for Apache

In previous chapters we can see, that PNG files, displaying the graphs are supposed to be saved to location /var/www/png/. There would be problem, with access rights because mentioned directory is after creation owned by root and Apache user www-data, does not has there the access rights to write, we need to change that. We can do that in following way



```
root@debian:/var/www# chown www-data png/
```

Ilustrace 26: setting permissions for apache

With use of the command chown we changed the owner of folder /var/www/ and because of that write attempts should be successful.

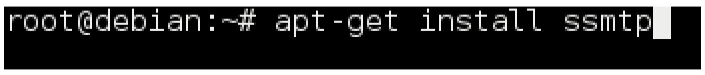
4.10 Sending notification emails

As it is mentioned in previous chapters, it is really helpful to send notification emails, when use of some system resource exceed give threshold.

There is several possible programs that allow us to do that, author of this thesis chose program ssmtp.

First step is to create Gmail account, in our case account router.notification.warning@gmail.com was created.

In the second step we need to install and configure ssmtp program.



```
root@debian:~# apt-get install ssmtp
```

Ilustrace 27: Installation of ssmtp

Followed with configuration of /etc/ssmtp/ssmtp.conf

Where we edit following parts

```
root=router.notification.warning@gmail.com
mailhub=smtp.gmail.com:587
hostname=router.notification.warning@gmail.com
UseSTARTTLS=YES
AuthUser=router.notification.warning
AuthPass=password #here we write real password, for given email account
FromLineOverride=yes
```

We should also edit /etc/ssmtp/revaliases

```
root:router.notification.warning@gmail.com:smtp.gmail.com
```

http://www.howtogeek.com/51819/how-to-setup-email-alerts-on-linux-using-gmail/

Now we have correct settings and system should be able to send emails. As a next step we focus on a script for sending the emails. In our case we send email, when disk usage exceed value of 90 percent. Thing to consider is that script is going to be executed each time, when Cron measures the disk space usage. We have to avoid sending email every minute, when script is launched, that is why we create a folder.

```
root@debian:~# mkdir /var/lib/email_warning/
```

Where is created file called sent, each time when email is being send to administrator and once a day set cronjob deleting the file so notification is being send once a day.

```
#!/bin/bash
used=`df | grep rootfs | awk '{print $5}' | cut -c1-2`
sentdisk="/var/lib/email_warning/sent"
if [ $used -ge 85 ]
then
        if [ ! -e "$sentdisk" ]
        then
          touch /var/lib/email_warning/sent
          echo "Hard drive is running out of free space" | sudo ssmtp karel.marsicek@gmail.com
        fi
fi
```

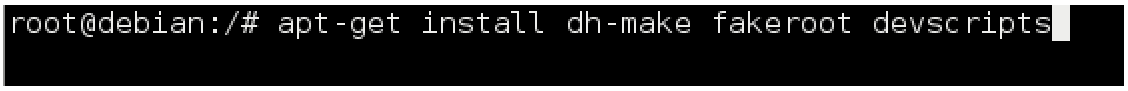
And as last step we add entry to crontab for deleting the file everyday

```
0 0 * * * rm -f /var/lib/email_warning/*
```

4.11 Creation of deb package

When we have working scripts we can prepare a simple package that would help the administrator to deploy them. That is the reason why in the following chapter we describe creation of a simple deb package. In the current development phase package is not going to be part of the official Debian repositories, that is the reason why the creation can be simplified and not following all the Debian packaging guidelines.

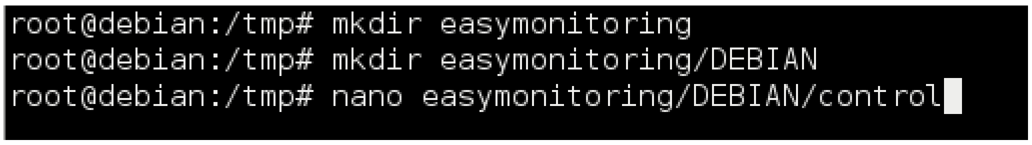
Firstly, our system should contain the required tools that can be installed in the following way.



```
root@debian:/# apt-get install dh-make fakeroot devscripts
```

Ilustrace 28: Installation of tools for package creation

As a second step we create directories and we can start with editing needed files



```
root@debian:/tmp# mkdir easymonitoring
root@debian:/tmp# mkdir easymonitoring/DEBIAN
root@debian:/tmp# nano easymonitoring/DEBIAN/control
```

Ilustrace 29: Preparation of directories

In next step we edit control file at the illustration above we open file in nano editor.

```
Package: mymonitoring
Version: 0.1
Section: utils
Priority: optional
Architecture: all
Essential: no
Depends: sysstat, apache2, php5-common, libapache2-mod-php5, php5-cli, rrdtool, php5-rrd
Maintainer: Karel Marsicek
Description: Package for monitoring of system resources
 This package was created while creation of master thesis at CULS Prague.
 RRDtool is used. There are scripts for gathering the data copied to the system.
 Scripts for presentation of data in PHP are also copied to the system.
 Cronjobs are added in the postinst phase.
```

We can see the dependencies specification, which is important for use in order to have all needed programs in place.

In next step we edit install file that specifies for us which files are supposed to be copied to given system location. We copy to the destination system scripts and also content of InlineDateTimePicker that is at our page as calendar to choose different observation period.

```
DEBIAN/mem.php var/www/
```

```
DEBIAN/disk.php var/www/
DEBIAN/traffic.php var/www/
DEBIAN/proc.php var/www/
DEBIAN/proc.sh etc/cron.d/
DEBIAN/mem.sh etc/cron.d/
DEBIAN/disk.sh etc/cron.d/
DEBIAN/traffic.sh etc/cron.d/
DEBIAN/bower.json var/www/
DEBIAN/datetimepicker.jquery.json var/www/
DEBIAN/jquery.datetimepicker.css var/www/
DEBIAN/jquery.datetimepicker.js var/www/
DEBIAN/jquery.js var/www/
DEBIAN/MIT-LICENSE.txt var/www
DEBIAN/left.png var/www/png/
DEBIAN/right.png var/www/png/
```

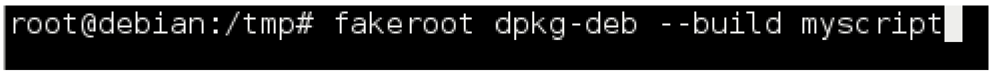
Then we edit postinst file, that says which steps should be proceeded after the installation.

```
mkdir /var/www/rrd/
mkdir /var/www/png/
chown www-data /var/www/png/
echo "Round Robin Archives are being created, it may take several minutes"
rrdtool create /var/www/rrd/user.rrd --step 300 DS:user:GAUGE:600:0:U RRA:MAX:0.5:1:105120
rrdtool create /var/www/rrd/nice.rrd --step 300 DS:nice:GAUGE:600:0:U RRA:MAX:0.5:1:105120
rrdtool create /var/www/rrd/system.rrd --step 300 DS:system:GAUGE:600:0:U RRA:MAX:0.5:1:105120
rrdtool create /var/www/rrd/iowait.rrd --step 300 DS:iowait:GAUGE:600:0:U RRA:MAX:0.5:1:105120
rrdtool create /var/www/rrd/disk.rrd --step 300 DS:disk:GAUGE:600:0:U RRA:MAX:0.5:1:105120
rrdtool create /var/www/rrd/mem.rrd --step 300 DS:mem:GAUGE:600:0:U RRA:MAX:0.5:1:105120
rrdtool create /var/www/rrd/rx.rrd --step 300 DS:rx:COUNTER:600:0:U RRA:MAX:0.5:1:105120
rrdtool create /var/www/rrd/tx.rrd --step 300 DS:tx:COUNTER:600:0:U RRA:MAX:0.5:1:105120


echo "*/5 * * * *   root    /etc/cron.d/mem.sh" >> /etc/crontab
echo "*/5 * * * *   root    /etc/cron.d/proc.sh" >> /etc/crontab
echo "*/5 * * * *   root    /etc/cron.d/traffic.sh" >> /etc/crontab
echo "*/5 * * * *   root    /etc/cron.d/disk.sh" >> /etc/crontab
echo "0 0 * * *   root      rm -f /var/lib/email_warning/*" >> /etc/crontab
```

```
mem_total=`free -m | grep Mem | awk '{print $2}'`
sed -i 's/--upper-limit=/--upper-limit='$mem_total'/g' /var/www/mem.php
disk_total=`df --block-size=1G | grep rootfs | awk '{print $2}'`
sed -i 's/--upper-limit=/--upper-limit='$disk_total'/g' /var/www/disk.php

if [ `ip link show | grep \< | awk '{print $2}' | awk '{print substr($0, 0, length($0))}' | grep -v lo | wc -l` -eq 1 ]
then
  int=`ip link show | grep \< | awk '{print $2}' | awk '{print substr($0, 0, length($0))}' | grep -v lo`
  sed -i "s/eth0/'$int'/g" /etc/cron.d/traffic.sh
elif [ `ip link show | grep \< | awk '{print $2}' | awk '{print substr($0, 0, length($0))}' | grep -v lo | wc -l` -eq 2
]
then
  cp /etc/cron.d/traffic.sh /etc/cron.d/traffic2.sh
  cp /var/www/traffic.php /var/www/traffic2.php
  echo "Another interface page created can be found at localhost/traffic2.php"
  j="traffic"
        for i in `ip link show | grep \< | awk '{print $2}' | awk '{print substr($0, 0, length($0))}' | grep -v lo`
       do
        sed -i "s/eth0/$i/g" /etc/cron.d/$j.sh
        sed -i "s/eth0/$i/g" /var/www/$j.php
        j="traffic2"
       done
  rrdtool create /var/www/rrd/rx2.rrd --step 300 DS:rx:COUNTER:600:0:U RRA:MAX:0.5:1:105120
  rrdtool create /var/www/rrd/tx2.rrd --step 300 DS:tx:COUNTER:600:0:U RRA:MAX:0.5:1:105120

  sed 's/tx.rrd/tx2.rrd/g' /etc/cron.d/traffic2.sh
  sed 's/rx.rrd/rx2.rrd/g' /etc/cron.d/traffic2.sh
  echo "* * * * *   root     /etc/cron.d/traffic2.sh" >> /etc/crontab
  echo "Additional page with graphs about traffic was created /localhost/traffic2.php"
fi
wget -P /var/www/ http://code.google.com/p/jqueryjs/downloads/detail?name=jquery-1.3.2.min.js
mv /var/www/index.html /var/www/oldindex.html
/etc/init.d/apache2 restart
```

There are created Round Robin Archives, prepared the directories etc. Important step is adjusting the parameter –upper-limit for our graph of used memory and disk space, because of that we can compare measured value with total size.

We can finally make a package



```
root@debian:/tmp# fakeroot dpkg-deb --build myscript
```

Ilustrace 30: Package creation

Simple package is now ready to be installed, now we will describe needed steps. Before we start command apt-get update could be very helpful.

```
dpkg -i mymonitoring.deb
```

With previous command we almost certainly receive error message about missing dependencies. Missing dependencies can be fixed with following command

```
apt-get -fy install
```

After mentioned step monitoring should be working and may be tested in web browser.

5 Discussion

Graphs are presented at the web page. When we see that there was some non standard behavior of the system, we may choose in the time picker concrete day, hour and so on and we can also change observed period with arrows at left and right sides of graph. This helps to orient in the graph in a better way and can be considered as an advantage.

Scripts gain from advantages of RRDtool. RRDtool uses Round Robin Archives, which have the final size when they are created. So, even in case when we run out of free disk space, RRDtool still works properly.

Prepared scripts could be deployed in a smaller network, especially community networks such as CZFree.Net could consider deployment of it. A drawback for such use is that these community networks nowadays very often use devices RouterBoard from company Mikrotik, where at some point can be used GNU/Linux system, but by default RouterOS is used, which is not compatible with prepared scripts.

Another aspect why it is supposed to be used in a small network environment is that for monitoring of bigger networks it is more suitable to use software that uses SNMP for obtaining information about the nodes and their monitored parameters, such as Nagios. These scripts are intended to be used at each machine separately.

6 Conclusion

The work fulfilled its aims. Scripts for creation of the graphs are presented in the work and print screens of result are also presented. There are scripts that cover problems of observation of:

- Processor load with separated modes for system mode, user mode, nice mode and so on.
- Script for disk space used. It presents in simple form how full is disk with root file system. In the script there is also functionality for alerting administrator when disk is running out of free space
- Script for memory used. It is presented in simple form, in further development could be divided for caches, buffers and so on.
- Network traffic that shows in one graph received and transmitted traffic

There was also prepared package for GNU/Linux Debian that helps administrator to use prepared scripts. Files with configuration such as postinst, which in general specify steps after installation (in our case scripts are copied to the system) is presented in the work.

For the purposes of testing we used at the beginning obsolete computer with following configuration Pentium II with frequency 450 MHz, RAM memory with size 256 MB and hard disk with 4327 MB. Later on we also used for testing notebook HP Compaq 6720s with configuration such as processor Intel(R) Pentium(R) Dual CPU T2410 with frequency 2.00 GHz, 2GB of RAM memory and 250 GB hard drive.

6.1 Resources

1. What is the Linux Kernel and What Does It Do?. [online]. [cit. 2013-10-20]. Available from: http://www.howtogeek.com/howto/31632/what-is-the-linux-kernel-and-what-does-it-do/
2. JANSSEN, Cory. Kernel. [online]. [cit. 2013-10-20]. Available from: http://www.techopedia.com/definition/3277/kernel
3. 1.1. History. [online]. [cit. 2014-03-18]. Available from: http://tldp.org/LDP/intro-linux/html/sect_01_01.html
4. Linux History. [online]. [cit. 2014-03-18]. Available from: http://www.livinginternet.com/i/iw_unix_gnulinux.htm
5. Linux Bible 2010 Edition: Running Debian GNU/Linux. Indianapolis, Indiana: Wiley Publishing, Inc, 2010. ISBN 978-0-470-48505-7.
6. About Debian: WHAT is Debian?. [online]. [cit. 2014-03-18]. Available from: https://www.debian.org/intro/about
7. Linux Bible 2010 Edition: Debian packages. Indianapolis, Indiana: Wiley Publishing, Inc, 2010. ISBN 978-0-470-48505-7.
8. Linux Bible 2010 Edition: Debian releases. Indianapolis, Indiana: Wiley Publishing, Inc. ISBN 978-0-470-48505-7.
9. About Linux's Copyright. [online]. [cit. 2014-03-18]. Available from: http://docstore.mik.ua/orelly/linux/run/ch01_06.htm
10. GNU General Public License (GNU GPL or simply GPL). [online]. [cit. 2014-03-18]. Available from: http://searchenterpriselinux.techtarget.com/definition/General-Public-License
11. CronHowto. [online]. [cit. 2014-03-18]. Available from: https://help.ubuntu.com/community/CronHowto
12. Schedule Tasks on Linux Using Crontab. [online]. [cit. 2014-03-18]. Available from: http://kvz.io/blog/2007/07/29/schedule-tasks-on-linux-using-crontab/?utm_source=feweekly&utm_campaign=issue0&utm_medium=web
13. Apache. [online]. [cit. 2014-03-18]. Available from: http://www.ntchosting.com/apache-web-server.html

14. Apache Web server. [online]. [cit. 2014-03-18]. Available from: http://www.webopedia.com/TERM/A/Apache_Web_server.html
15. Beginning PHP 5.3. Indianapolis, Indiana: Wiley Publishing, Inc., 2010. ISBN 978-0-470-41396-8.
16. PHP:Introduction. [online]. [cit. 2014-03-18]. Available from: http://www.php.net/manual/en/security.intro.php
17. GNU Bash. [online]. [cit. 2014-03-18]. Available from: http://www.gnu.org/software/bash/
18. BashScripting. [online]. [cit. 2014-03-18]. Available from: https://help.ubuntu.com/community/Beginners/BashScripting
19. 12 Useful "df" Commands to Check Disk Space in Linux. [online]. [cit. 2014-03-18]. Available from: http://www.tecmint.com/how-to-check-disk-space-in-linux/
20. Linux and Unix free command. [online]. [cit. 2014-03-18]. Available from: http://www.computerhope.com/unix/free.htm
21. Linux and Unix iostat command. [online]. [cit. 2014-03-18]. Available from: http://www.computerhope.com/unix/iostat.htm
22. HowTo: Use grep Command In Linux / UNIX – Examples. [online]. [cit. 2014-03-18]. Available from: http://www.cyberciti.biz/faq/howto-use-grep-command-in-linux-unix/
23. Linux Cut - Linux Commands. [online]. [cit. 2014-03-18]. Available from: http://lowfatlinux.com/linux-columns-cut.html
24. 10 Practical Linux Cut Command Examples to Select File Columns. [online]. [cit. 2014-03-18]. Available from: http://www.thegeekstuff.com/2013/06/cut-command-examples/
25. Awk. [online]. [cit. 2014-03-18]. Available from: http://tldp.org/LDP/abs/html/awk.html
26. How to Use Awk to Find and Sort Text in Linux, GnuCash. [online]. [cit. 2014-03-18]. Available from: http://www.linux.com/learn/tutorials/724060-amazing-awk-fu-gnucash-import-find-blocks-of-text-remove-dupes-without-sorting/
27. Linux / Unix Command: sed. [online]. [cit. 2014-03-18]. Available from: http://linux.about.com/od/commands/l/blcmdl1_sed.htm

28. Top 10 Reasons To Use Network Monitoring. [online]. [cit. 2014-03-18]. Available from: http://www.businesscomputingworld.co.uk/top-10-reasons-to-use-network-monitoring/
29. Why Monitor Network Performance?. [online]. [cit. 2014-03-18]. Available from: http://blog.globalknowledge.com/technology/why-monitor-network-performance/#sthash.dpYVpL4L.dpbs
30. Introduction to RRD - Round Robin Database. [online]. [cit. 2014-03-19]. Available from: http://www.loriotpro.com/Products/On-line_Documentation_V5/LoriotProDoc_EN/V22-RRD_Collector_RRD_Manager/V22-A1_Introduction_RRD_EN.htm
31.
32. The Nagios Setup Explained. [online]. [cit. 2014-03-19]. Available from: http://www.linuxforu.com/2011/07/nagios-setup-guide/
33. Nagios: System and Network Monitoring Book. [online]. [cit. 2014-03-19]. Available from: http://www.cyberciti.biz/tips/book-review-nagios-system-network-monitoring.html
34. ICINGA – An advanced opensource monitoring tool | Nagios fork. [online]. [cit. 2014-03-19]. Available from: http://www.unixmen.com/icinga-an-advanced-opensource-monitoring-tool/
35. What is Icinga?. [online]. [cit. 2014-03-19]. Available from: https://www.icinga.org/about/
36. Cacti (software). [online]. [cit. 2014-03-19]. Available from: http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Cacti_(software).html
37. Zabbix 1.8 Network Monitoring. Birmingham: Packt Publishing Ltd., 2010. ISBN 978-1-847197-68-9.
38. Instant Munin Plugin Starter. Birmingham: Packt Publishing Ltd., 2013. ISBN 978-1-84969-674-6.
39. Vytváříme vlastní distribuci Linuxu. Brno: Computer Press, a.s., 2010. ISBN 978-80-251-2433-8.
40. Cacti. [online]. [cit. 2014-03-19]. Available from: http://djdesignerlab.com/2012/05/28/10-most-organized-network-monitoring-tools/

41. Zabbix. [online]. [cit. 2014-03-19]. Available from: http://djdesignerlab.com/2012/05/28/10-most-organized-network-monitoring-tools/
42. Installing jQuery?. [online]. [cit. 2014-03-19]. Available from: http://stackoverflow.com/questions/1458349/installing-jquery
43. How do I Find Out Linux CPU Utilization?. [online]. [cit. 2014-03-19]. Available from: http://www.cyberciti.biz/tips/how-do-i-find-out-linux-cpu-utilization.html
44. RRDtool – graph data. [online]. [cit. 2014-03-19]. Available from: http://khmel.org/?p=236
45. Permissions. [online]. [cit. 2014-03-19]. Available from: http://linuxcommand.org/lts0070.php
46. Linux Pocket Guide, 2nd Edition. Cambridge: O'Reilly Media, Inc., 2012. ISBN 978-1-449-31669-3.
47. Network and System Monitoring with Zenoss Core. Birmingham: Packt Publishing Ltd., 2011. ISBN 978-1-849511-58-2.
48. Zenoss Overview. [online]. [cit. 2014-03-21]. Available from: http://sourceforge.net/projects/zenoss/
49. Zenoss Core. [online]. [cit. 2014-03-22]. Available from: http://www.networkworld.com/reviews/2007/061807-open-source-management-test-zenoss.html
50. Nagios Installation and configuration. [online]. [cit. 2014-03-19]. Available from: http://www.debianhelp.co.uk/nagiosinstall.htm

6.2 Supplements

6.2.1 Web page source code

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=windows-
1250">
   <title>Router configuration and statistics</title>
      <style>
      * { margin: 0; padding: 0; }
      ul { margin: 20px 10px; }
      ul li { float: left; list-style-type: none; }
      ul li a { background: #006699; color: white; display: block; font-
family: Arial, Helvetica, sans-serif; padding: 10px 20px; text-
decoration: none; }
      ul li a:hover { background: #0069d4; }
      .left { float:left; border:none; position:relative; top:80px; }
      .right { float:right; border:none; position:relative; top:80px; }
  /* Creation of menu was inspired by http://css-plus.com/2010/03/how-to-
create-a-simple-horizontal-navigation-menu-from-scratch/ */
      </style>
            <link rel="stylesheet" type="text/css"
href="jquery.datetimepicker.css"/ >
            <script src="jquery.js"></script>
            <script src="jquery.datetimepicker.js"></script>
            <script src="my.js"></script>
  </head>
  <body style="background:#006699">
    <font color="white">
        <h1>Router Statistics</h1>
            <b>
            <ul>
            <li><a href="disk.php">Disk usage</a></li>
            <li><a href="index.php">Processor load</a></li>
            <li><a href="mem.php">Memory usage</a></li>
            <li><a href="traffic.php">Network traffic</a></li>
            </ul>
            </b>
            <br />
            <br />
                <br />
<h3>Inline DateTimePicker</h3>
</font>
<form action="index.php" method="get">
      <input type="text" name="time" class="datetimepicker" value="<?=
$_REQUEST['time'] ?>"/><br><br>
      <input type="submit" value="change graph">
</form>
<!-- Inline DateTimePicker is available at
http://plugins.jquery.com/datetimepicker/ -->
<br />
<br />
<?php

$ret=sscanf($_REQUEST['time'], "%d/%d/%d %d:%d", $year,$month,$day,$hour,
$minute);
if($ret == 5)
{
      $ts = mktime($hour, $minute, 0, $month, $day, $year);
}
else $ts=time()-3600;
create_graph("png/proch.png", $ts, $ts+3600, "Processor load - hour");
create_graph("png/procd.png", $ts, $ts+86400, "Processor load - day");
create_graph("png/procw.png", $ts, $ts+604800, "Processor load - week");
create_graph("png/procm.png", $ts, $ts+2592000, "Processor load -
month");
create_graph("png/procy.png", $ts, $ts+31536000, "Processor load -
year");
```

```
echo "<table>";
echo "<tr><td>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts - 3600))."\">";
echo "<img src='png/left.png' class='left'>";
echo "</a>";
echo "<img src='png/proch.png' alt='Generated RRD image'>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts + 3600))."\">";
echo "<img src='png/right.png' class='right'>";
echo "</a>";
echo "</tr></td>";
echo "<tr><td>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts - 86400))."\">";
echo "<img src='png/left.png' class='left'>";
echo "</a>";
echo "<img src='png/procd.png' alt='Generated RRD image'>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts + 86400))."\">";
echo "<img src='png/right.png' class='right'>";
echo "</a>";
echo "</td></tr>";
echo "<tr><td>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts - 604800))."\">";
echo "<img src='png/left.png' class='left'>";
echo "</a>";
echo "<img src='png/procw.png' alt='Generated RRD image'>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts + 604800))."\">";
echo "<img src='png/right.png' class='right'>";
echo "</a>";
echo "</td></tr>";
echo "<tr><td>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts -
2592000))."\">";
echo "<img src='png/left.png' class='left'>";
echo "</a>";
echo "<img src='png/procm.png' alt='Generated RRD image'>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts +
2592000))."\">";
echo "<img src='png/right.png' class='right'>";
echo "</a>";
echo "</td><td>";
echo "<tr><td>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts -
31536000))."\">";
echo "<img src='png/left.png' class='left'>";
echo "</a>";
echo "<img src='png/procy.png' alt='Generated RRD image'>";
echo "<a href=\"?time=".urlencode(date('Y/n/j G:i', $ts +
31536000))."\">";
echo "<img src='png/right.png' class='right'>";
echo "</a>";
echo "</td></tr>";
echo "</table>";
exit;

function create_graph($output, $start, $end, $title) {
  $options = array(
"-w",
"500",
"-h",
"150",
"-a",
"PNG",
    "--slope-mode",
    "--start", $start,
    "--end", $end,
    "--title=$title",
    "--vertical-label=Processor load",
    "--lower=0",
    "--color", "BACK#363636",
    "--color", "CANVAS#000000",
    "--color", "GRID#999999",
    "--color", "MGRID#B5B5B5",
    "--color", "FONT#CCCCCC",
    "DEF:user=/var/www/rrd/user.rrd:user:MAX",
    "DEF:nice=/var/www/rrd/nice.rrd:nice:MAX",
    "DEF:system=/var/www/rrd/system.rrd:system:MAX",
    "DEF:iowait=/var/www/rrd/iowait.rrd:iowait:MAX",
    "CDEF:utilization_pct=user,system,iowait,+,+,user,system,nice,iowait,
+,+,+,/,100,*",
    "AREA:user#FFD700:Processor load user(%):STACK",
    "GPRINT:user:LAST:Last\: %5.2lf",
    "GPRINT:user:AVERAGE:Avg\: %5.2lf",
    "GPRINT:user:MAX:Max\: %5.2lf",
    "GPRINT:user:MIN:Min\: %5.2lf\\t\\t\\t\\n",
    "AREA:nice#FF0000:Processor nice (%):STACK",
    "GPRINT:nice:LAST:Last\: %5.2lf",
    "GPRINT:nice:AVERAGE:Avg\: %5.2lf",
    "GPRINT:nice:MAX:Max\: %5.2lf",
    "GPRINT:nice:MIN:Min\: %5.2lf\\t\\t\\t\\n",
    "AREA:system#1E90FF:Processor system (%):STACK",
    "GPRINT:system:LAST:Last\: %5.2lf",
    "GPRINT:system:AVERAGE:Avg\: %5.2lf",
    "GPRINT:system:MAX:Max\: %5.2lf",
    "GPRINT:system:MIN:Min\: %5.2lf\\t\\t\\t\\n",
    "AREA:iowait#00FF00:Processor iowait (%):STACK",
    "GPRINT:iowait:LAST:Last\: %5.2lf",
    "GPRINT:iowait:AVERAGE:Avg\: %5.2lf",
    "GPRINT:iowait:MAX:Max\: %5.2lf",
    "GPRINT:iowait:MIN:Min\: %5.2lf\\t\\t\\t\\n",
   );

  $ret = rrd_graph($output, $options);
  if (! $ret) {
    echo "<b>Graph error: </b>".rrd_error()."\n";
  }
}

?>
          <br />
          <br />
  </body>
</html>
```