University of Thessaly, 2014-2015

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

THESIS

Title: ”Development of a framework for monitoring and measurement collection of networking resources of the NITOS research facility”

Soulidis Angelos

Supervisor Professor: Athanasios Korakis

Co-supervisor Professor: Antonios Argyriou

Volos, February 2015

Πανεπιστήμιο Θεσσαλίας , 2014-2015

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Τίτλος : “Ανάπτυξη πλατφόρμας παρακολούθησης και καταγραφής μετρήσεων των δικτυακών πόρων της πειραματικής υποδομής του NITOS”

Σουλίδης Άγγελος

Επιβλέπων Καθηγητής : Κοράκης Αθανάσιος

Συνεπιβλέπων Καθηγητής : Αργυρίου Αντώνιος

Βόλος , Φεβρουάριος 2015

Ευχαριστίες

Για την περάτωση της διπλωματικής μου εργασίας που έλαβε χώρα στο τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του πανεπιστημίου Θεσσαλίας θα ήθελα να ευχαριστήσω τους ανθρώπους που βοήθησαν για αυτόν τον σκοπό και επί της ουσίας στην ολοκλήρωση των σπουδών μου.

Αρχικά θα ήθελα να ευχαριστήσω τον Επίκουρο Καθηγητή του τμήματος κ. Αθανάσιο Κοράκη για την προσφορά του θέματος της πτυχιακής μου εργασίας και των μέσων για να την φέρω εις πέρας. Επίσης ένα μεγάλο ευχαριστώ στον υποψήφιο Διδάκτορα Δονάτο Σταυρόπουλο που με βοήθησε πολύ με τις γνώσεις του και ήταν πάντα πρόθυμος να βοηθήσει. Θα ήθελα, τέλος, να ευχαριστήσω την οικογένεια μου για την στήριξη που μου προσέφερε και τους φίλους μου με τους οποίους πέρασα όλα αυτά τα χρόνια και με βοήθησαν και αυτοί να φτάσω στο τέλος των σπουδών μου.

Contents

ΠΕΡΙΛΗΨΗ

Στόχος αυτής της διπλωματικής εργασίας είναι η ανάπτυξη μίας πλατφόρμας που θα επιτρέπει στον χρήστη την παρακολούθηση και συλλογή πληροφοριών σχετικά με τους δικτυακούς πόρους της υποδομής του NITOS. Για να πετύχουμε τον σκοπό αυτό χρησιμοποιήσαμε ένα εργαλείο ανοιχτού κώδικα, το Nagios, το οποίο μας προσφέρει τα αποτελέσματα των μετρήσεων στους πόρους, σε συνδυασμό με το Nagira, ένα RESTAPIμέσω του οποίου αποθηκεύουμε τα αποτελέσματα των μετρήσεων σε κάποια Database. Το σημαντικότερο μέρος της διπλωματικής εργασίας είναι η ανάπτυξη μίας καινούργιας, δικής μας πλατφόρμας, που θα επιτρέπει στον χρήστη να ελέγχει ποιούς πόρους παρακολουθεί, να προσθαφερεί εύκολα και γρήγορα πόρους προς παρακολούθηση, τα χαρακτηριστικά των πόρων που είναι προς παρακολούθηση, να αλλάξει τα χαρακτηριστικά μιας μέτρησης, να ορίσει που θα αποθηκευτούν τα αποτελέσματα καθώς και επιπλέον λειτουργικότητα.

ABSTRACT

The goal of this thesis is the development of a framework for monitoring and measurement collection of networking resources of the NITOS facility. We use Nagios, an open source tool, which enables us to monitor our resources, in association with Nagira, a REST API, which help us retrieve and store the data that Nagios offers to us. The most important part of this thesis is the development of our own framework that gives the user the ability to easily control which resources he is monitoring, easily add/remove a resource, easily add/remove monitoring attributes, define where the information will be stored, and even more functionality.

1 Introduction

1.1 Monitoring tools

In order to accomplish our first goal, which is to monitor the NITOS experimental infrastructure, we use open source tools. These tools help us monitor the switches and the nodes that are part of our infrastructure and more specific we retrieve information about port status, bandwidth/port (for the switches)and disk usage, CPU utilization, memory utilization and other attributes (for the nodes). There are plenty tools that can be used. Most common tools are listed below:

1.1.1NAGIOS

This is one of the most popular web-based Linux Monitoring systems nowadays, actually it's industry standard for IT infrastructure monitoring. Licensed under GPL, Nagios is available for everybody free of charge and allows monitoring availability and response time of network services, usage of system resources like CPU load,RAM allocation etc., number of logged in users and many more. In case of any outage detected by Nagios server or any anomaly you will get an alert from Nagios.

1.1.2CACTI

Cacti is another web based monitoring system written in PHP and licensed under GPL. Unlike Nagios described above Cacti was designed mainly for the graphs — in brief, Cacti polls various services and then graphs resulting data. So you can see CPU load graphs, RAM usage, round trip time stats, bandwidth utilization and much more information collected from various hosts. Cacti does not provide alerts.

1.1.3ZABBIX

ZABBIX

Zabbix is an enterprise class Linux monitoring system with impressive list of capabilities available out of the box. It is licensed under GPL and is written in PHP. In brief Zabbix can do the same tasks as Nagios and Cacti by default: Zabbix easily graphs monitored data and sends alerts to user in case of any problem. Using Zabbix you can create maps of the hosts, group hosts by various categories and so on.

1.1.4Nfsen

Nfsen is open source Netflow collector and analyzer available under open source license. It differs from monitoring tools described here — Nfsen collects only network usage data and shows the interactive graphs based on that data.

1.1.5MRTG

MRTG is yet another open source monitoring tool that collects data at local and/or remote host by means of SNMP protocol. But MRTG is much simpler than Cacti, Nagios or Zabbix so it may be a good choice for small projects.

2 Nagios 3.0

The tool we chose to use is Nagios, version 3.0. It is a very powerful, widely used tool, which enables us to monitor our experimental infrastructure, using its' numerous plugins. As it is pointed above, Nagios can be combined with other tools, so we will also use the Multi Router Traffic Grapher (MRTG). We will use Nagios 3.0 to retrieve the data that it can give us, through the services it provides to us. Such data is Ping, switch uptime, port status, bandwidth /port. It is necessary that the switch supports Simple

Network Management Protocol (SNMP) and we will also need to install the Multi Router Traffic Grapher (MRTG).

2.1 SNMP

Simple Network Management Protocol (SNMP) is an application-layer protocol defined by the Internet Architecture Board (IAB) in RFC1157 for exchanging management information between network devices. It is a port if Transmission Control Protocol / Internet Protocol (TCP / IP) protocol suite.

SNMP is one of the widely accepted protocols to manage and monitor network elements. Most of the professional-grade network elements come with bundled SNMP agent. These agents have to be enabled and configured to communicate with the network management system (NMS).

SNMP consist of:

- SNMP Manager
- Managed Devices
- SNMP Agent
- Management Information Base (MIB)

2.1.1 SNMP Manager

A manager or management system is a separate entity that is responsible to communicate with the network devices on which an SNMP agent is implemented. This is typically a computer that is used to run one or more network management systems.

2.1.2 Monitoring devices

A managed device or the network element is a part of the network that requires some form of monitoring and management e.g. routers, switches, servers, workstations, printers, UPSs etc.

2.1.3 SNMP Agent

The agent is a program that is packaged within the network element. Enabling the agent allows it to collect the management information database from the device locally and makes it available to the SNMP manager, when it is queried for. These agents could be standard (e.g. Net-SNMP) or specific to a vendor (e.g. HP insight agent).

2.1.4 Management Information Base (MIB)

Every SNMP agent maintains an information database describing the managed device parameters. The SNMP manager uses this database to request the agent for specific information and further translates the information as needed for the Network Management System (NMS). This commonly shared database between the Agent and the Manager is called Management Information Base (MIB). In short, MIB files are the set of questions that a SNMP Manager can ask the agent. Agent collects this data locally and stores it, as defined in the MIB. So the SNMP Manager should be aware of these standard and private questions for every type of agent.

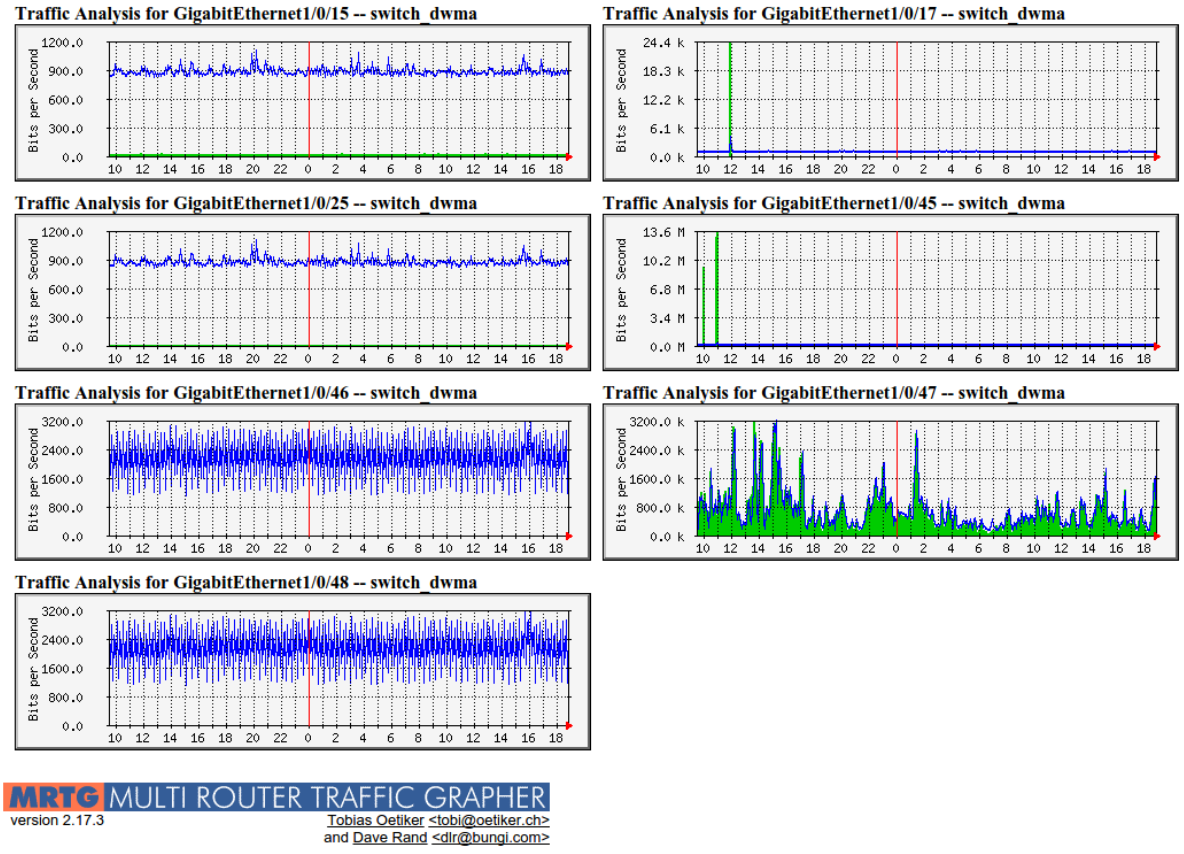
2.2 MRTG

The Multi Router Traffic Grapher (MRTG) is a tool used for monitoring the traffic load on network links. MRTG generates HTML pages containing PNG images, which provide a

LIVE visual representation of this traffic. This is the tool we used combined with Nagios, to check the port bandwidth of each switches' port.

2.2.1 MRTG output

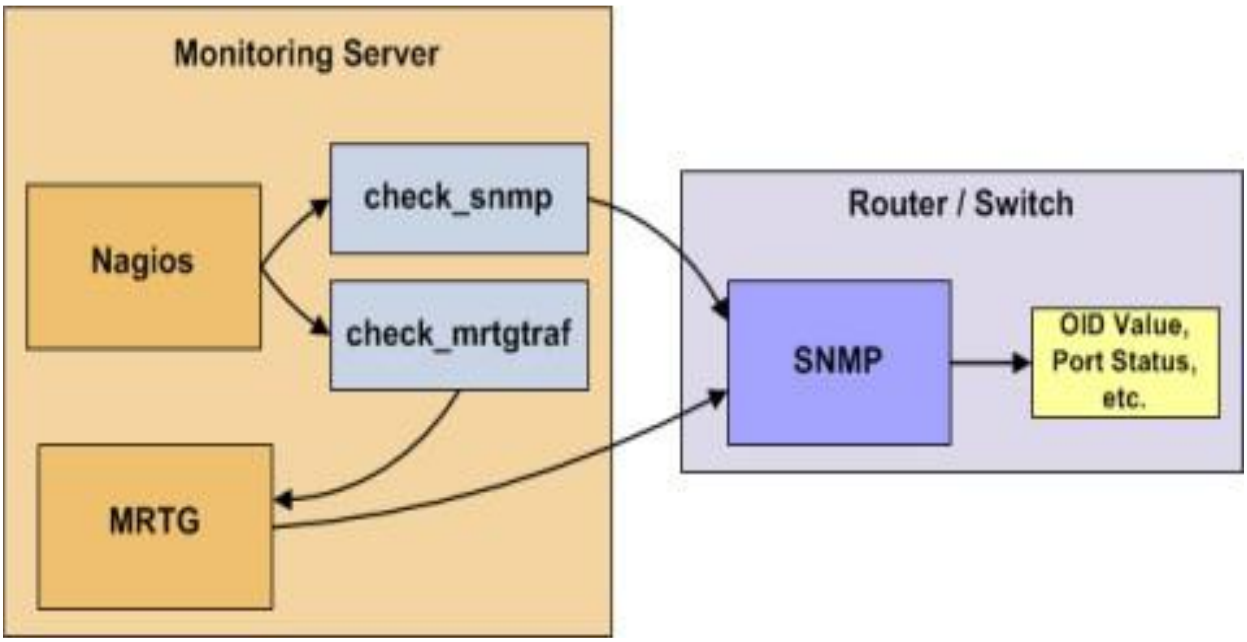
Here you can see the MRTG graphs given in the web interface:



2.3 Connection between Nagios-SNMP-MRTG

The first plugin that Nagios offers us and we use is check_snmp. This enables us to Ping the switch, to monitor how much time it is up (Uptime) and which of its' ports are up or down. Another plugin that Nagios offers us and we use it to monitor the bandwidth/port is check_mrtgtraf plugin.

Below we can see schematically the connection between Nagios – SNMP – MRTG:

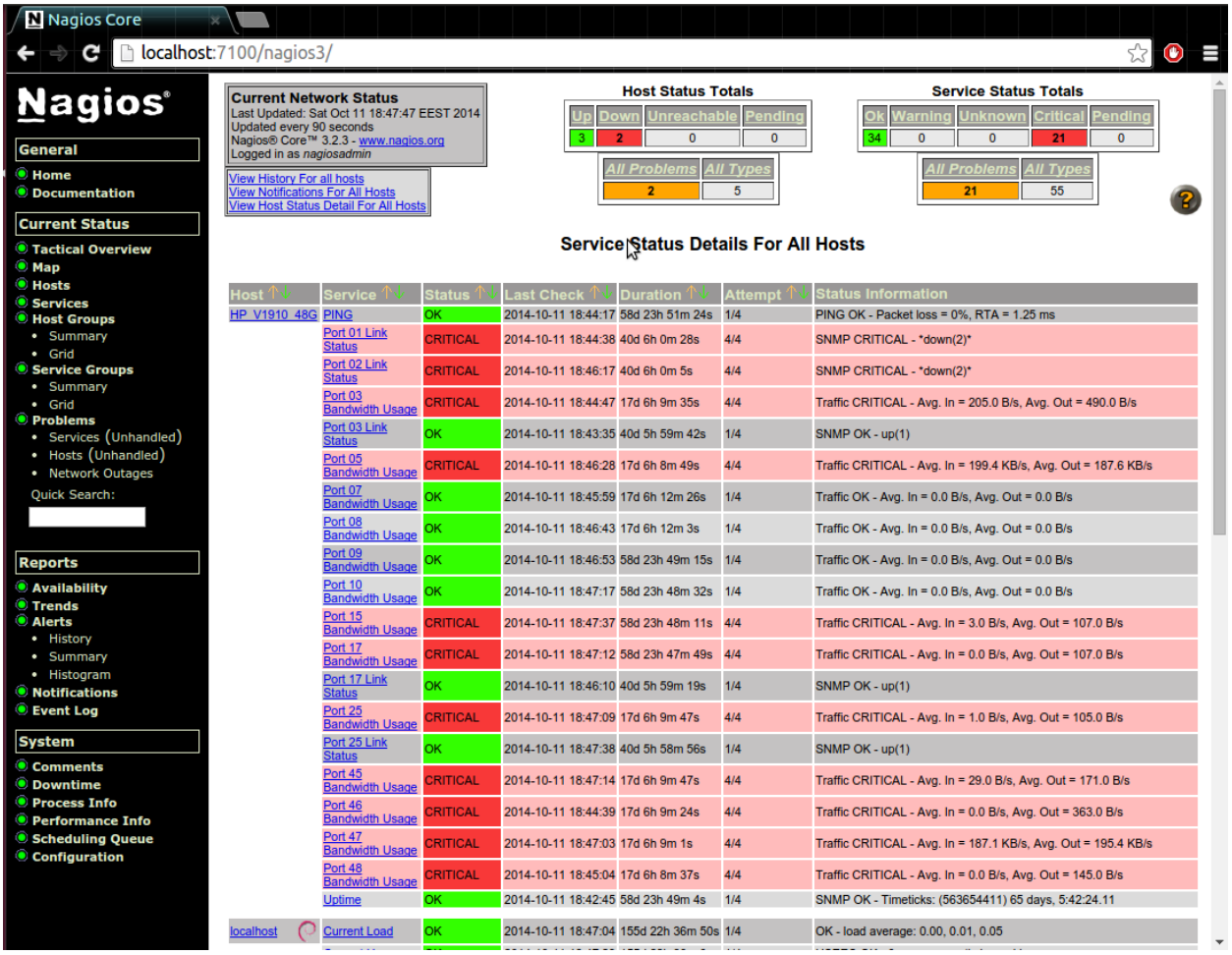


3 Monitoring a switch using Nagios 3.0

Once we install Nagios 3.0 and MRTG we are ready to use Nagios and its' powerful plugins. The only thing we have to do is add the services we want Nagios to execute. In the **Documentation** you can see how Nagios is used and how hosts and services are added in order to be monitored.

3.1 Results

After the procedure above we are ready to see the results. In our case we got the result shown below:



(This is the output in the Nagios' web interface)

3.2 Important comments

As you can see in the picture with the results, every single service written in the switch configuration file which Nagios executes is shown in the web interface. But now it is the best time, after getting the first results, to explain some very important details that will help us understand what we see:

- As you can see next to the service name there are 5 columns.
  - Status: can be
    - OK : Everything is working without problems.
    - CRITICAL : Either there is an error that Nagios cannot overcome and you should check the configuration again, or the result is beyond the critical thresholds you have defined.
    - WARNING : The result is in the warning thresholds that you defined.
  - Last Check : The exact time Nagios checked the service.
  - Duration : How much time Nagios executes the service.
  - Attempt : How many attempts Nagios made until it found the result.
  - Status Information :The result that we want Nagios to return.
- The user can determine the warning/critical thresholds, max attempts value, check interval etc. in the switch configuration file.

User can create hostgroups, so he can add whichever switch he wants in every group and thus organize his infrastructure better. This helps a lot, since Nagios can not only monitor switches, but also Linux/Unix machines, Netware servers, routers, network printers and publicly available services (HTTP, FTP, SSHetc.).

4 Retrieving Measurements

Once we made Nagios working and we are able to see the results, we now need to collect those measurements and store them in a Database. To accomplish that goal we need to use Nagira, a REST API suitable for Nagios.

4.1 NAGIRA

NAGIRA is a very useful REST API that helps us retrieve all the data that Nagios returns and displays at its web interface. Of course we control which data we retrieve, through NAGIRA's features, listed below:

- HTTP GET request to retrieve Nagios server or objects configuration, and for host or service status
- Multiple output formats: add format extension at the end of the route (.xml, .json, .yaml).
- Full or short host or service state information (for short summary add /_state at the end of the route).
- Lists of objects (hosts, services, configured objects) where applicable: add /list at the end of the route.

We can run these HTTP calls in the command line, but we need something more efficient, that enables us to store these measurements in a Database and not only see them in the terminal.

4.2 OML

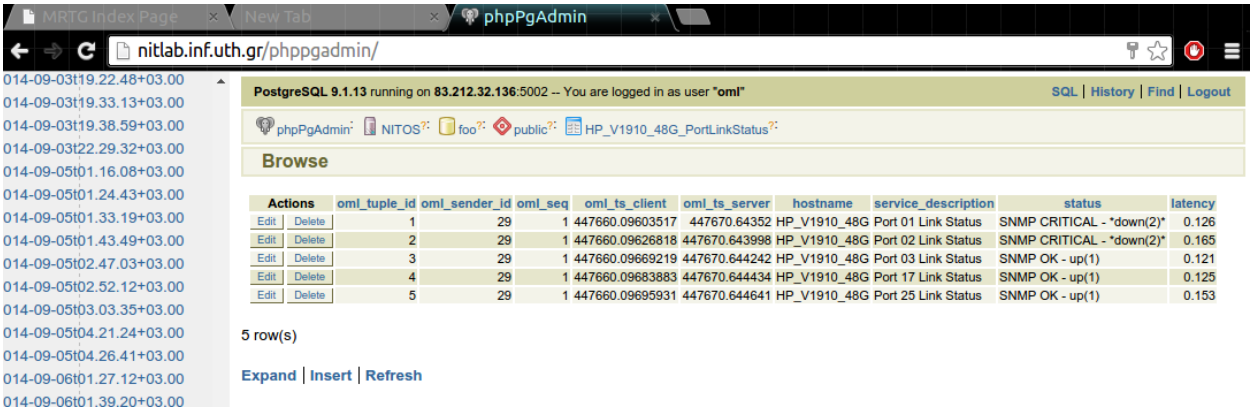
OML is an instrumentation tool that allows application writers to define customizable measurement points (MP) inside new or pre-existing applications. Experimenters running the applications can then direct the measurement streams (MS) from these MPs to remote collection points, for storage in measurement databases. OML was originally conceived to provide measurement facilities for OMF-enabled testbeds, such as the Wireless ORBIT Testbed. It is now a stand-alone tool which can also be run independent of OMF. OML is now a generic software framework for measurement collection.

As described above, we use NAGIRA to retrieve those measurements. But the main task is to store those measurements in a database. For that purpose we used OML scripts. In these scripts, written in ruby, we include all the necessary HTTP calls that NAGIRA offers us, and we forward the output to be stored in our Database.

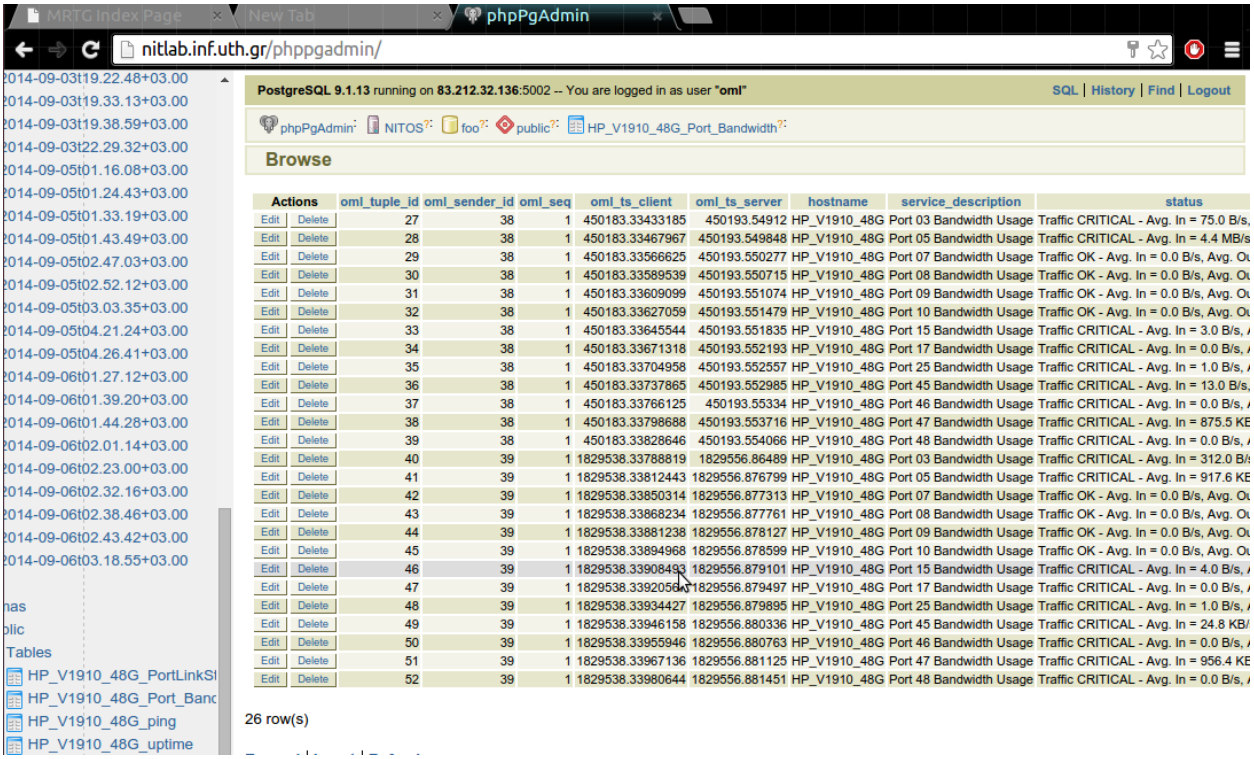
An extra feature we added, is the user to determine which switches' measurements we want to store, by giving its' name right after the script-run command. In that way we can monitor many switches, but store only the measurements we want.

4.3 Results of stored data

Below you can see the results of the stored measurements:



(Port Status)



(Bandwidth Usage / Port)

(Ping)

(Uptime)

4.4 Important comments

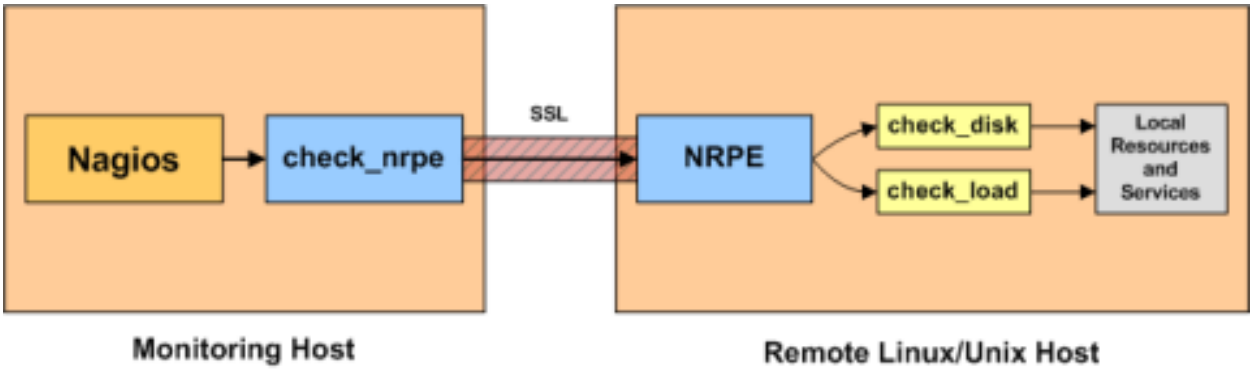
In the scripts we define where the results of the HTTP calls will be stored. In our case the results are stored in the NitLab database, under the domain “foo”, each service owning a table with the results.

5 Monitoring Remote Hosts

After completing the steps that are required to monitor a switch, the next step is to monitor remote hosts. Nagios gives us this possibility by using the Nagios Remote Plugin executor (NRPE).

5.1 NRPE

NRPE allows you to remotely execute Nagios plugins on other Linux/Unix machines. This allows you to monitor remote machine metrics (disk usage, CPU load, etc.). NRPE can also communicate with some of the Windows agent add-ons, so you can execute scripts and check metrics on remote Windows machines as well.



5.2 Plugins

Since we are able to monitor remote hosts using NRPE, let's see which plugins we used to accomplish that goal:

- Standard checks that the NRPE configuration file provide us:
  - Check_load
  - Check_users
  - Check_disk
  - Check_procs

Every check command returns values according to the default thresholds that are defined in the plugin file. Of course the user can change those thresholds as he wish.

- Current Load: Returns OK / WARNING /CRITICAL depending on the load amount
- Current Users: Returns the number of the users logged in the host
- Disk Space: Returns OK / WARNING /CRITICAL depending on the free disk space
- Total Processes: Returns the number of the processes running on the host.

Those plugins were not enough for us, because we need to retrieve specific measurements for these services. So we had to add some more plugins,which we found in the http://exchange.nagios.org/directory/Plugins, where you can find thousand plugins for Nagios.

So the plugins that we used are listed below:

- Non-standard plugins we used:
  - Check_mem
  - Check_cpu_stats.sh
  - Check_iftraffic_nrpe.sh
  - Check_ifconfig.sh

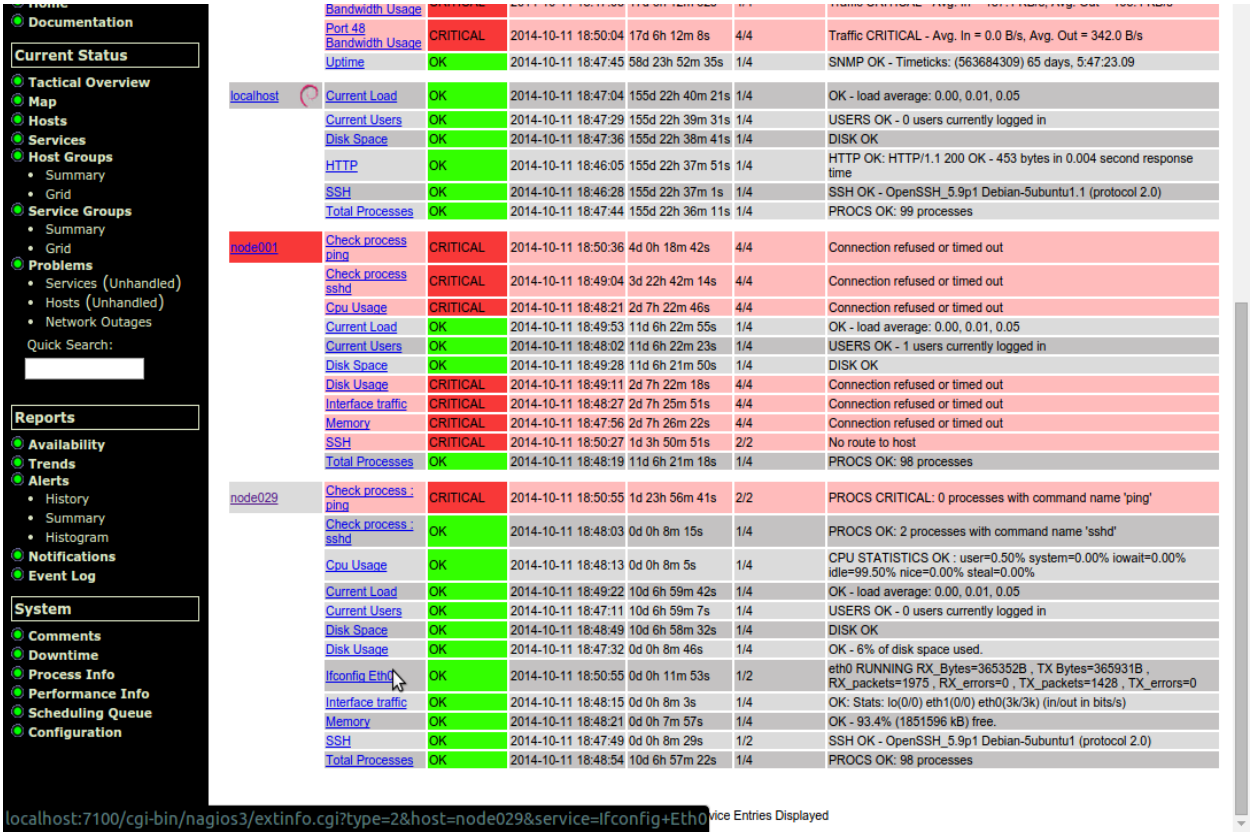
- Check_mem: Used to monitor free memory (percentage + free memory in KB).
- Check_cpu_stats.sh: Returns the percentage of CPU usage (user, system, iowait, idle, nice, steal). To use this plugin the installation of the package “sysstat” is required, so we can use a tool it provides, named “iostat”.
- Check_iftraffic_nrpe.sh: Returns the amount of in/out bits of every interface available on the host.
- Check_ifconfig.sh: Returns the amount of RX and TX bytes and packets on the interface we chose.

For the disk usage we created a plugin according to this guide https://www.digitalocean.com/community/tutorials/how-to-create-nagios-plugins-with-bash-on-ubuntu-12-10, changing the OK/WARNING/CRITICAL thresholds to 90%.

We also used check_procs plugin to monitor if specific processes are running on the host. So we added two new services to check if “sshd” and “ping” are running on the host.

5.3 Results

After adding all the plugins that were written above, we are ready to see the results in the Nagios web interface:



As we can see the host “node029” is added beneath the “localhost” and “node001” sections. As we can see at the top of the page there are two services that refer to the switch we monitored and are also in the same web interface.

Now “node029” results are:

node029	Check process : ping	CRITICAL	2014-10-11 18:50:55	1d 23h 56m 41s	2/2	PROCS CRITICAL: 0 processes with command name 'ping'
	Check process : sshd	OK	2014-10-11 18:48:03	0d 0h 8m 15s	1/4	PROCS OK: 2 processes with command name 'sshd'
	Cpu Usage	OK	2014-10-11 18:48:13	0d 0h 8m 5s	1/4	CPU STATISTICS OK : user=0.50% system=0.00% iowait=0.00% idle=99.50% nice=0.00% steal=0.00%
	Current Load	OK	2014-10-11 18:49:22	10d 6h 59m 42s	1/4	OK - load average: 0.00, 0.01, 0.05
	Current Users	OK	2014-10-11 18:47:11	10d 6h 59m 7s	1/4	USERS OK - 0 users currently logged in
	Disk Space	OK	2014-10-11 18:48:49	10d 6h 58m 32s	1/4	DISK OK
	Disk Usage	OK	2014-10-11 18:47:32	0d 0h 8m 46s	1/4	OK - 6% of disk space used.
	Ifconfig Eth0	OK	2014-10-11 18:50:55	0d 0h 11m 53s	1/2	eth0 RUNNING RX_Bytes=365352B , TX Bytes=365931B , RX_packets=1975 , RX_errors=0 , TX_packets=1428 , TX_errors=0
	Interface traffic	OK	2014-10-11 18:48:15	0d 0h 8m 3s	1/4	OK: Stats: lo(0/0) eth1(0/0) eth0(3k/3k) (in/out in bits/s)
	Memory	OK	2014-10-11 18:48:21	0d 0h 7m 57s	1/4	OK - 93.4% (1851596 kB) free.
	SSH	OK	2014-10-11 18:47:49	0d 0h 8m 29s	1/2	SSH OK - OpenSSH_5.9p1 Debian-5ubuntu1 (protocol 2.0)
	Total Processes	OK	2014-10-11 18:48:54	10d 6h 57m 22s	1/4	PROCS OK: 98 processes

5.4 Important comments

As we can see everything works fine on this host and no WARNING or CRITICAL messages appear. The only CRITICAL message appearing in the picture refers to the process “ping” which was not running on the host the time that the screenshot was taken. That’s why we run the ping process on the host to see if the monitoring service is actually working and the result was correct.

All those services can very easily be applied on every host of the NITOS infrastructure so we can monitor every host we want, anytime we want.

6 REST API

The main part of this thesis is the development of a framework for monitoring and measurement collection of NITOS research facility. So we decided to develop our own REST API, using Sinatra (http://www.sinatrarb.com) and Ruby. NAGIRA is an already existing REST API, but we needed a more complete, powerful and easier to use REST API. Once we use Sinatra, it is obvious that all the commands are HTTP calls, used under specific syntax and parameters that are defined and shown in the appendix at the end of this thesis.

6.1 Nagios Measurement Information

Nagios store the information about its measurements in the status.dat file. That information is about hosts and each host services. In other words, host characteristics like hostname, service names that are monitored on this host, uptime and many others, and service characteristics like service-name, monitoring parameters, monitoring time and many others, are all included in this status.dat file. We need to access this file to

create our REST API so we created a file (nagios_stats.rb) in order to parse it and get the information we need.

6.2 Monitor Hosts' and Services' status

The first and most simple functionality that our REST API supports is the overall control of the hosts and the services. This means that the user can make the following requests:

- Host status
- Specific service
- Host list
- Service list

By requesting a host's status the user can see all the information that Nagios keeps about this host. By requesting a specific service the user gets all the information kept for this specific service of a specific host. By requesting the host list what is returned is the list with all the hosts that Nagios was assigned to monitor and finally by requesting a service list the user gets the names of the services that a specific host is monitored for. To achieve this goal, we used the file that we created and were mentioned above (nagios_stats.rb) and all the results are returned in json format.

6.3 Add Hosts/Services

Using Nagios we noticed that it is very hard for someone that has never used it before to easily handle it. In other words, it is very hard for someone that simply wants to monitor an infrastructure to use Nagios because he needs to insert hardcoded data in order to make everything work. For that purpose we develop our API (by creating Nagios_config_files.rb) giving the user the opportunity to add:

- New Hosts
- New Services

Adding a new host means a new switch, a new node and anything else Nagios supports monitoring. Adding a new service under an already defined and configured host commands Nagios to monitor a specific host's attribute under specific parameters.

6.4 Edit Hosts/Services

Until now there is no possibility to modify and already existing host or service without modifying Nagios files via an editor. Now we make it easier for the user to modify the host or service he wants just by inserting in the terminal the attributes he wants to change and the new values (Nagios_edit.rb).

6.5 Delete Hosts/Services

The same difficulty in adding hosts/services in Nagios exist in removing them. This can be done only by finding the host's/service's configuration file in Nagios configuration folder and deleting it. Our REST API supports a simple HTTP call that removes the host or the service that the user wants to remove just by passing the name as an argument of this HTTP call in the command line. This is much easier and neither demands the user to know where Nagios configuration files are nor risking deleting another file by mistake.

6.6 OML scripts/Monitoring Jobs

We use OML scripts, just like we did when we monitored the switch, to store the measurements that Nagios offers us in a database. We now automate the procedure by using HTTP calls to schedule an OML script to run for monitoring a specific service of a specific host, see a list of the monitoring jobs running for a host's service, see details of a specific job and delete a specific job.

6.6.1 Schedule a monitoring Job

Using our REST API the user can now schedule (using Rufus Scheduler/ https://github.com/jmettraux/rufus-scheduler) a monitoring job to run with the interval and the duration given by him as arguments in the HTTP call, along with the data that he wants to be monitored by that job.

6.6.2 Monitoring Jobs

By typing another HTTP call user can get a list of all the monitoring jobs that are running for a specific host and include a specific service, without considering the data that is monitored for this service.

6.6.3 Job details

After seeing the list of the jobs running for a host and a service, the user can isolate the job he wants (by using the jobid given as a result in the previous HTTP call) and see its details in the console by typing another HTTP call that includes that jobid.

6.6.4 Delete a Job

The same way we get a job's details, we can delete a running job just by passing as arguments, to another HTTP call, the hostname, servicename and jobid of the job we want to delete.

7 Documentation

It is important for the user to know exactly how to use this REST API. Using the commands above, the user can exploit all the possibilities that are offered. It must be mentioned that by using this REST API **we can also deploy Nagios to monitor switches** and storing the results, in other words we can easily do what we did in the first section of this thesis.

7.1 Restrictions

There are few Restrictions that the user must be aware of, while using our REST API. Firstly, any name given to a new host or service or to any new entity, and is longer than one word, must be given with underscore between the words. Another Restriction is that when the user adds a new monitoring service he must know and write in the command exactly the same name of the attributes that he wishes to monitor, as they are given by Nagios. Moreover, like the attributes' names, attention must be given in all the commands that include a jobid, as it is unique and must be given correctly.**It is important to notice that for the commands below the user provides the host's name and the service's name by replacing "hostname" and "servicename" respectively, according to the Restrictions.**

7.1.1 Return service status

```
curl localhost:4567/hosts/"hostname"/services/"servicename"
```

Writing the command above the user sees in the terminal all the details that Nagios offers about a specific service of a specific host. Both host and service must have already been configured and Nagios running this service.

7.1.2 Return host status

```
curl localhost:4567/hosts/”hostname”
```

Likewise service status, user can see host status simply by writing the command above.

7.1.3 Return service list

```
curl localhost:4567/hosts/”hostname”/services
```

This command returns all the services that Nagios is running for a specific host. So only the hostname is required.

7.1.4 Return host list

```
curl localhost:4567/hosts
```

Like the previous command, this command returns a list of all the hosts that Nagios is deployed to monitor (in our case nodes or switches).

7.1.5 Create new host

```
curl -i -X POST -d
'{"host_name":”hostname”,"ip_address":”ipaddress”}'
localhost:4567/hosts
```

When the user wants to create a new host all he has to do is provide the hostname and the IP address of this host (by replacing “ipaddress”).

7.1.6 Create new service

```
curl -i -X POST -d
'{"service_description":”servicedescr”,"check_command":”chec
kcomm”}' localhost:4567/hosts/”hostname”/services
```

To create a new service for a host, service description and check command must be provided (by replacing “servicedescr” and “checkcomm” respectively and respecting the name Restrictions). Host’s name is given in the HTTP call URL.

7.1.7 Edit host

```
curl -i -X PUT -d
'{"host_name":”hostname”,"ip_address":”newipaddress”}'
localhost:4567/hosts
```

Editing a host means changing its IP address. To do that, the user only needs to write the hostname and the new IP address in the command above (by replacing “newipaddress” and respecting the name Restrictions).

7.1.8 Edit service

```
curl -i -X PUT -d
'{"service_description":”servicedescr”,"check_command":”chec
kcomm”}' localhost:4567/hosts/”hostname”/services
```

Editing a host's service means changing the service description and the check command. To do this, the user only needs to write the new service description and the new check command in the command above (by replacing "servicedescr" and "checkcomm" respectively and respecting the name Restrictions). Host's and service's name is given in the HTTP call URL.

7.1.9 Delete host

```
curl -X DELETE localhost:4567/hosts/"hostname"
```

User can delete a host just by typing the command above, which contains the name of the host to be deleted.

7.1.10 Delete service

```
curl -X DELETE
localhost:4567/hosts/"hostname"/services/"servicename"
```

User can delete a host's service just by typing the command above, which contains the name of the service to be deleted.

7.1.11 Schedule a monitoring Job

```
curl -i -X POST -d
'{"uri":"omlcollect","domain":"domainname","interval":"users
interval","duration":"usersduration","metrics":["metric1","m
etric2"]}'
localhost:4567/hosts/"hostname"/services/"servicename"/monit
oring
```

This command schedules a monitoring job to start with specific parameters that the user defines.These are:

- OML collect URI : TCP server where the measurements will be stored (replacing “omlcollect”)
- Domain name : Domain name for the measurements in the database (replacing domainname)
- Interval : Measurements interval (replacing usersinterval)
- Duration : How long the measurements will last (replacing usersduration)
- Metrics : Which attributes of a monitoring service (e.g. output, duration, name etc.) will be stored in the database (replacing metrics1, metcrics2 or adding more in the same way if the user wants to).

Host’s and service’s name is given in the HTTP call URL. It must be mentioned **that one or more jobs can be assigned on the same host’s service** but they must either have different interval or duration or the metrics that will be stored in the database.

7.1.12 Return monitoring Jobs list

```
curl
localhost:4567/hosts/”hostname”/services/”servicename”/monit
oring
```

By typing this command, the user can see in the terminal all the jobs that are running for the service given in the HTTP call URL.

7.1.13 Return details of a monitoring Job

```
curl
localhost:4567/hosts/”hostname”/services/”servicename”/monit
oring/”jobid”
```

In order to type this command and see the result, the user must first run the previous command that returns the list of the monitoring jobs and pick a jobid that he wants. Then, by replacing the “jobid” in the command above with the jobid he picked, the user can see the metrics that this job measures and stores in the database.

7.1.14 Delete a monitoring Job

```
curl –X DELETE
localhost:4567/hosts/”hostname”/services/”servicename”/monit
oring/”jobid”
```

This command is used to delete a running monitoring job the same way the previous command works, just by writing in the HTTP call URL the jobid, in addition with the hostname and servicename.

7.2 Important comments

All the commands above express the full functionality of this REST API. Although users is not aware of the files that are created, modified or deleted, it is important to mention that the configuration files are separated. This means that a host’s configuration file and its’ services’ configuration files are separated from each other, making it easier to handle them and therefore reducing the possibility to delete or edit files that are vital for Nagios to work properly.

8 Conclusion

To sum all the work that has been done up, we can say that we developed a very powerful framework. After using Nagios firstly to monitor a switch and then NITOS' nodes we spotted its' weaknesses and developed a framework very easy to use. In other words, a new user that wants to monitor an infrastructure using Nagios just needs to read our REST API's documentation, use the commands that were presented above in a very detailed way and in a very short time see the measurements collected in his database. Moreover, he can easily handle Nagios without needing to write code or search in the Nagios' configuration files to add/delete/modify hosts or services.