UNIVERSITY OF AMSTERDAM

GRADUATE SCHOOL OF INFORMATICS
System and Network Engineering

GreenSONAR

A multi-domain energy profiling system based on perfSONAR

Lutz Engels <Lutz.Engels@os3.nl>
Todor Yakimov <Todor.Yakimov@os3.nl>

Feb 18, 2013

Supervisors
Dr. Paola Grosso <p.grosso@uva.nl>
Karel van der Veldt <karel.vd.veldt@uva.nl>
Hao Zhu <h.zhu@uva.nl>

University of Amsterdam
Graduate School of Informatics
Science Park 904
1098XH Amsterdam

Contents

1 Introduction

National Research and Education Networks (NRENs), like most other networks, make extensive use of monitoring systems to keep track of their networking devices and to assure continous service to their users.

Due to the nature of NRENs the data that is constantly being collected, is also shared accross domain borders, for research purposes. The sharing is acchieved by the use of distributed *Multi-Domain Monitoring* (MDM) systems that in turn utilize technologies such as *Service Oriented Architecture* (SOA) for the purpose of providing a common access scheme. A framework that is used by NRENs for that purpose is *perfSONAR.*

When taking account of the need to lower CO_2-emissions by reducing energy consumption, the hereabove mentioned MDM systems enable interessting usage scenario as they provide the foundation for inclusion of energy data in the distribution. The availability of the energy data on a multi-domain level introduces possibilities for energy-aware networks.

The project examines more closely whether existing performance monitoring tools that define all of those distributed and robust capabilities can be instrumented for monitoring energy consumption and more general metrics. Furthermore investigation is done on how to define energy metrics and what applications their aivailability provide in green path decissioning.

1.1 Related work

An extensive energy aware semantic model capable of providing energy knowledge has recently been published [10]. However, its complexity does not fit into the scope of this project, as it would require to either additionally extend perfSONAR with ontology capabilities or to flatten its structure, which takes away its strength. Furthermore research has been done on how network performance metrics should be composed [5], which provides a base for the composition of energy metrics.

1.2 Research Questions

This leads to the research questions:

"What metrics need to be considered in order to build energy profiles of networking devices and how can such data be published by using distributed multi-domain monitoring systems."

Specifically saying:
"Is perfSONAR-PS a suitable architecture to achieve energy profiling of computational devices, and what are the necessary steps to be undertaken to evolve perfSONAR-PS in a system we can call 'GreenSONAR'?"

1.3 Thesis Outline

The remainder of the report is organized as follows. In section 2 an introduction to *Energy Profiling* is given. Section 3 gives insight on the choice of *Metrics* for the energy profiles. Afterwards section 4 then presents the *perfSONAR* framework and its implementations. Section 5 (*GreenSONAR*) describes the proposed extension of perfSONAR through which energy-aware networking can be enabled. After presenting the Results in section 6 they are discussed in section 7, which concludes this report.

2 Energy Profiling

Until recently emphasis on IT-infrastructure laid on performance. Only from 2006 research started to be conducted on how to make it more energy efficient [2]. To make an effort to also use current infrastructure in a more efficient manner, there is need to map energy profiles of devices. On the basis of those profiles decissions can be taken towards whether or not, or when certain infrastructure is to be used.

2.1 What is energy profiling?

An *energy profile* is a collection of energy related properties of devices. Within the scope of IT-network infrastructure energy profiling can be described as the extension of currently available performance metrics by including metrics that take into account energy consumption of devices.
These metrics can be used to generate a per device energy profile providing information on the energy efficiency under different circumstances. For a computing device e.g. the power consumption will rise, when the CPU load is high. Furthermore the energy profiles can be taken advantage of for networking path decisioning.

2.2 Why is it needed?

Since at least 2009 there is official scientific consensus that climate change by the amplification of greenhouse gas (GHG) emissions is anthropogenic [4]. Therefore steps need to be taken to reduce the emissions wherever possible.
Power consumption of the Internet in 2010 represented between estimated 1.1 and 1.9% of the global energy consumption [8]. In absolute terms this is between 170 and 307GW. As a lot of the currently operating infrastructure was build with providing performance in terms of speed, but not with energy savings, there is a lot to win.

2.3 What are current restraints?

The main restraints in applying energy profiling to save energy are the lack of standard in both data representation and normalization as well as the lack of sensors in deployed hardware. The first results in administrative limitations, as too much manual intervention is needed to gather energy data at all. Hardware vendors partly implement sensors and interfaces to present the readings, but not in a standardized manner. To perform data gathering on a wide range of devices the variety of different vendor approaches has to be taken into account. This results in too big of an effort. The current methods to gather energy data at all are presented hereunder.

SNMP *Simple Network Management Protocol* is a protocol that allows to manage devices through the network. It uses *Management Information Bases* (MIBs) as extensible means to define the variables that will be exposed by the device. A wide variety of MIBs has been defined, amongst which some that contain energy metrics. Unfortunately (<- please change, not scientific) the definition alone does not mean that it is implemented nor implementable in all devices.

PDU A *Power Distribution Unit* is a device that distributes power to a number of devices. Most PDUs have a more advanced set of functions, amongst which the availability to expose the energy consumption per outlet. Racktivity devices for example can be read out through an *Advanced Programming Interface* or *Application programming interface* (API). The presented readings and the APIs differ per vendor, whereas there is no standard.

CLI Network hardware vendors often equip their devices with a *Command-line interface* as means of configuring and (basic) monitoring of the device. Some vendors make accessible the sensor-readings of devices by CLI-commands. Unfortunately (<- please change, not scientific) neither the CLIs itself, nor the CLI-command to show the readings, nor the way the readings are presented, are standardized.

Furthermore there are data sharing limitations, as nor protocol nor infrastructure is in place to spread the data across domain-boundaries.

2.4 How can above mentioned problems be overcome?

To overcome the above mentioned constrains, standardization has to be introduced on several layers. Chapter 3 aims at providing the base for a standard by defining a metric that is as independent as possible from the mentioned limitations.

3 Metrics

As discussed in chapter 2 energy profiles provide the base for energy aware infrastructure. To enable such profiles metrics need to be defined. This chapter explains the derivation of a suchlike metric.

3.1 Focus

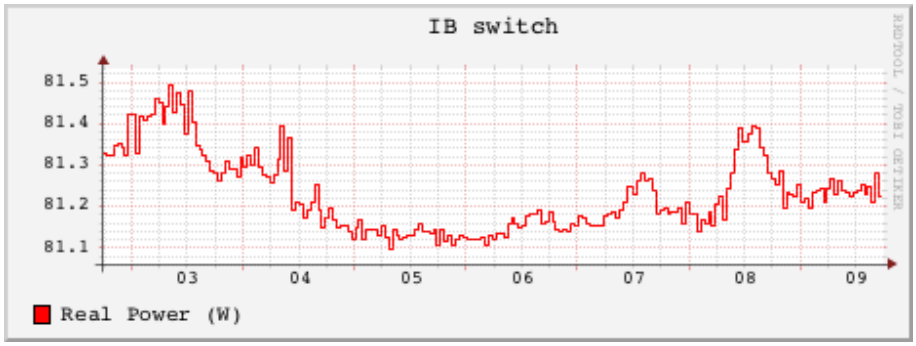
The approach of this research is to be as generic as possible. It aims at being able to include as much infrastructure as possible. To be able to define the metric, it needs to be framed what the generically available data is.

3.1.1 Racktivity PDU readings

In most cases the readings of a *Power Distribution Unit* (PDU) represent the only uniform means of providing energy data within ICT-infrastructure.

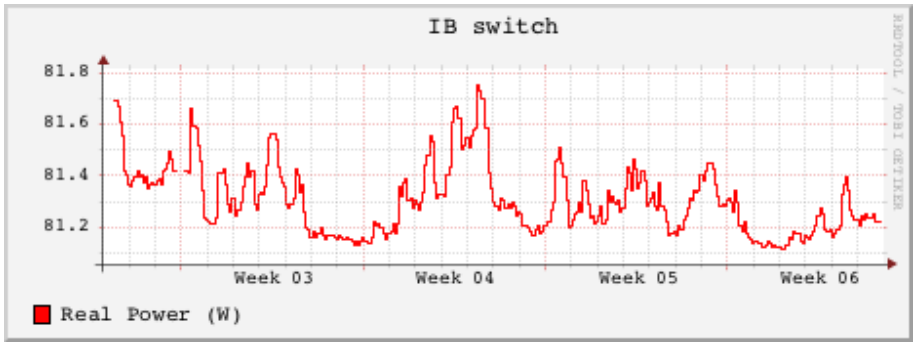
Also the absence of *802.3az (Energy-Efficient Ethernet)* results in nearly constant energy consumption regardless of the utilization.

Examining one of the PDU outlets of the DAS-4 cluster, confirms this behavior. (see figure 3.1 and 3.2)



http://145.100.102.81:22001/view/uva01/p/8?timespan=604800

Figure 3.1: Week overview of the UvA DAS-4 ibswitch (PDU outlet 8)



http://145.100.102.81:22001/view/uva01/p/8?timespan=2592000

Figure 3.2: Month overview of the UvA DAS-4 ibswitch (PDU outlet 8)

3.2 Metric

Parker and Walker [6] introduced the logarithmic *absolute energy efficiency* metric (3.1) that enables the comparison of efficiency of ICT-architecture, ICT-hardware, etc.

$$dB\varepsilon = 10log_{10}\left(\frac{Power/BitRate}{kTln2}\right) \tag{3.1}$$

Since GreenSONAR is an approach to standardize, this metric was chosen, as it provides universal comparability irrespective to specific technology.

The derived definition 3.2 extends $Power/BitRate$ by taking into account the number of ports N_{ports} and distributing the current energy consumption P_{total} evenly amongst them. This value is put into relation to the bandwidth left $Speed_{max} - Util_p * Speed_{max}$. If a port has a utilization of e.g. 80%, it calculates to 20% of the $Speed_{max}$ of that port in $bits/seccond$. The division of energy consumption amongst the ports seems crude, but in an attempt to keep the definition as generic as possible, the most apparent and definitely available variables as input.

$$dB\varepsilon_{cpp} = 10log_{10}\left(\frac{\frac{P_{total}}{N_{ports}}/Util_p * Speed_{max}}{kTln2}\right) \tag{3.2}$$

Where
$dB\varepsilon$: absolute energy efficiency
$dB\varepsilon_{cpp}$: absolute current energy efficiency per port
P_{total}: current total energy consumption of device
N_{ports}: number of ports
S_{max}: maximum speed of port in bits/second
$Util_p$: current port utilization
k: Boltzmann constant ($1.381 * 10^{-23} J/K$)
T: temperature in Kelvin
$kTln2$: absolute minimum energy per bit dissipated

If in future the distribution of energy profiling data is in place and more extensive the following definition might be worth considering to replace aforementioned $\frac{P_{total}}{N_{ports}}$ with definition 3.3. It makes use of baseline measurements of devices [7]. This would require to register baseline measurements for all devices or global historical energy data from which the baseline could be derived. The former represents a unfeasible overhead and the latter is not in place yet.

$$P_{pp} = \frac{P_{tp} - P_b}{N_p} \tag{3.3}$$

Where
P_{pp}: estimated power per port
P_{tp}: average power for a specific throughput level
P_b: average baseline power
N_p: number of ports

A further consideration towards the future is taking advantage of the fact that the derived metric includes the temperature T. Devices perform more or less efficient according to their operation temperature. When temperature-sensor readings get more widely available, they can be included in the calculation and thereby yield more accurate values for the absolute energy efficiency.

4 perfSONAR

With various Network Monitoring Systems (NMS) in place, computer networks are steadily becoming more resilient while also optimizing their performance and availability. Such systems are often part of domains that fall under certain administrative policies and access to monitoring data is limited to the domain's administrators and users. Making it available to third-parties will help approach problems such as jitter, packet loss, transmission delay and bandwidth availability amongst others on a multi-domain level. However, the vast amount of network technologies, policies and managers turn its distribution into a complicated process. Such restrictions have led to initiatives for the development of Network Monitoring Service-Oriented Architectures (NMSOA)[9]. A conjoint effort between educational entities and National Research Networks (NRENs), the perfSONAR framework is one initiative that enables the discovery, collection, storage and distribution of monitoring data. At its heart, the framework strives to diminish all data access-related administrative restrictions and make it easier to troubleshoot end-to-end performance problems on multi-domain paths. The perfSONAR framework originated out of the following three contexts[1]:

- A consortium of organizations who seek to build network performance middleware that is interoperable across multiple networks and useful for intra- and inter-network analysis. One of the main goals is to make it easier to solve end-to-end performance problems on paths crossing several networks.
- A protocol. It assumes a set of roles (the various service types), defines the protocol standard (syntax and semantics) by which they communicate, and allows anyone to write a service playing one of those roles. The protocol is based on SOAP XML messages and following the Open Grid Forum (OGF) Network Measurement Working Group (NM-WG).
- Several, interoperable software packages (implementations of various services) that attempt to implement an interoperable performance middleware framework. Those sets of code are developed by different partners.

[1] http://www.perfsonar.net/

4.1 Architecture of the framework

perfSONAR defines a decentralized system for sharing network measurements. The general monitoring infrastructure implemented by the framework is depicted in Fig. 4.1.

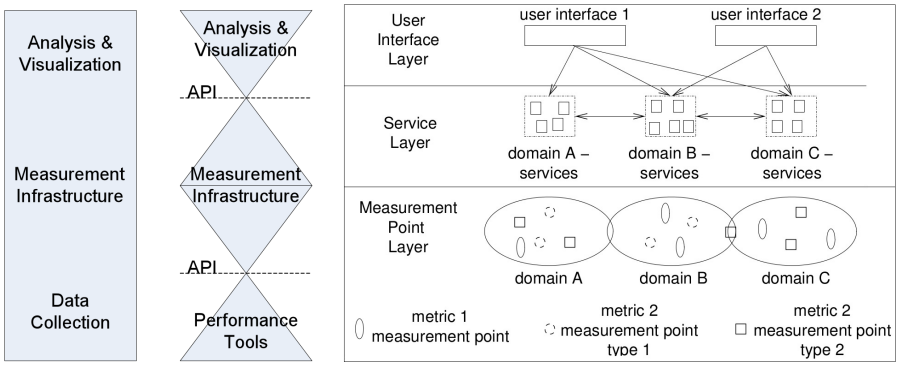


Figure 4.1: The three layered infrastructure of the perfSONAR framework.

The base layer defines domain-specific Measurement Points (MP) capable of collecting distinct metrics obtained by an underlying monitoring tool. Multiple MPs can be deployed within a domain. The Services Layer is the middle layer of the system and can span across multiple administrative domains. It enables the inter-domain exchange of measurement data and management information through disjoint Web Services (WS). The User Interface Layer consists of reporting and visualization tools that can present data in customizable ways.

4.1.1 Measurement Point Layer

At this layer, the tasks of obtaining and storing measurement data are defined. The measurements themselves are carried out by active or passive monitoring tools such as Ping, Iperf or SNMP queries. Each MP is designed as a wrapper around a particular monitoring tool while also defining interfaces for remote invocations. In principle, MPs are highly versatile and do not pose any restriction on what technology or programming language is used for defining them as long as their communication with the rest of the framework is realized in accordance with the communication standard. The current perfSONAR releases provide MPs capable of interfacing with the following resources and measurement tools:

- Telnet/SSH MP - returns the output of "traceroute" and "ping" commands
- Round-Robin Databases (RRD) MP
- Structured Query Language (SQL) Database MP
- Pinger MP - ICMP ping command wrapper
- Bandwidth Control (BWCTL) MP - Iperf TCP/UDP throughput measurement tool wrapper

4.1.2 Services Layer

The Services Layer provides a high degree of access transparency between data collectors and data consumers by diminishing differences in data access and representation. It does so by defining the following services:

- Authentication and Authorization - Authentication Service (AS)
- Discovery of services alike in other domains - Lookup Service (LS)
- Aggregation, correlation and filtering of measurement data - Transformation Service (TS)
- Storage of measurement data collected by MPs - Measurement Archives (MA)

The interaction of entities part of the layer as well as access to the Measurement Point layer is not visible to the user. For this to be achieved, all data providers define a "publisher" interface while all data consumers implement a "subscriber" interface.

4.1.3 User Interface Layer

The User Interface Layer consists of entities such as visualization tools (GUIs) that enable end-users to adapt, format and visualize measurement data in versatile ways with the needs of certain applications and user groups in mind. Users of any service, regardless of whether they are end-user application or services themselves are regarded as clients. These services allow users to perform tests using the lower layers of the framework. From the perspective of users, the Service Layer abstracts the differences of Measurement Points deployed across the different participating domains.

4.1.4 NMWG protocol

The data communication protocol used by the framework's services is based on SOAP (Simple Object Access Protocol) XML messages and adheres to the Global Grid Forum (GGF) Network Monitoring Working Group (NMWG) XML schema. HTTP is used as the underlying transport protocol. The general structure of messages exchanged between services is depicted in Fig. 4.2

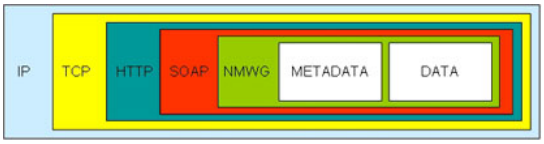


Figure 4.2: The encapsulation of metadata + data in the NMWG protocol.

The NMWG schema is also used for normalizing measurement data. This is achieved by segmenting the presentation of measurement data into two parts: meta data and data. The paradigm allows for easily extending the data types supported by the framework's communication protocol when new MPs and MAs are deployed within the existing infrastructure.

4.2 Implementations

Two perfSONAR implementation exist at current that fully comply with the NM-SOA depicted in Fig. 4.1, namely perfSONAR-PS and perfSONAR MDM. They are actively developed by different NRENs for servicing the requirements of particular users bases and infrastructures. perfSONAR MDM is developed by GÉANT and aims at providing a seamless Multi-Domain Monitoring system for the GÉANT Service Area. Measurement Points and client services provided by this implementation are developed with the needs of NOC and PERT engineers in mind. perfSONAR-PS is developed by ESnet and Internet2 with the purpose of optimizing the troubleshooting process between campus and research networks as well as network providers.

A third implementation (perfSONAR NC) developed by UNINET offers standalone Measurement Points incapable of registering with a local or global lookup service. An overview of available perfSONAR implementations is depicted in Table 4.1

Feature	**perfSONAR-PS/MDM**	**perfSONAR NC**
Services	MP, MA, LS, AS, TS	MP
Protocol	NMWG	NETCONF
Data models	none	YANG
Query Mechanisms	MA/MP dependent	XPATH
Scalability	1000 domains	>200K dinaubs
Code base	>6000 lines, MA specific 3000	<1000 lines, MA specific <200
Programming language	Perl/Java	PHP

Table 4.1: perfSONAR features by release

The major differences between the perfSONAR-PS and MDM releases is the programming language used in their implementation as well as the type of Measurement Points they define. Measurement Points from both releases (Table 4.2) that target the distribution of the same metric are still capable of communicating with one another because of the standardized NMWG protocol. Both releases do not follow a certain data model through which services at the lower two layers of the framework need to be implemented. Therefore, the query mechanisms that need to be used with a given service are highly specific to its own interfaces. A complete comparison of the main characteristics that distinguish the releases from one another can be found in [3]

Measurement Archive	**perfSONAR-PS**	**perfSONAR MDM**
Reverse Traceroute	•	•
Reverse Ping	•	•
BWCTL(Iperf, NutTCP)	•	•
OWAMP (RFC4656)	•	•
SNMP Round-Robin Database	•	•
SQL database		•

Table 4.2: perfSONAR-PS/MDM Measurement Archives

4.3 Extensibility

To instrument the software towards the distribution of new metrics, a deep familiarization with the framework was conducted. The outcome showed that the NMSOA framework adopted by the perfSONAR-PS and MDM distributions is a good choice for a system, which to instrument towards energy profiling of network nodes for the following reasons:

- The software provides a uniform, well-organized system for the collection, distribution and consumption of network-related metrics that mediates
  - data discovery, and
  - data access
- The high degree of access transparency provided by the Service layer enables the seamless integration of newly implemented Measurement Archive in perfSONARs current global infrastructure.
- Instrumenting the software towards the distribution of new metrics can be achieved in an autonomous way by:
  - Provisioning of new NMWG schemas that describes the underlying measurement data
  - Provisioning of new Measurement Points and Archives that collect, store and serve the data.
- Its user base is comprised of various NRENs, which makes it a perfect testbed

for such a new and unstandardized system

4.3.1 Code base extensibility

In order to evaluate what are the important parts of creating a new Measurement Archive, an extensive examination of the code base of perfSONAR-PS was conducted as the software is a community-driven, agile-programming [1] reference to perfSONAR release management) effort, which lacks a standardized documentation base. Development is mainly carried out via conference calls and mailing lists [3].

With a lack of documentation on the subscriber/publisher APIs defined by the different services, determining how to create a new Measurement Archive relied on performing a full installation of the perfSONAR-PS with the purpose of first verifying the valid functionality of the different global and local services. Afterwards, the source code bundles of the different services were examined. The following research questions were defined:

1. How do Measurement Archives register and maintain communication with local and global lookup services?
2. Is there a generic data model followed in the implementation of Measurement Points and Archives?
3. How is data collected by a Measurement Point
    a. Normalized
    b. Made available to a Measurement Archive
4. Is there any stateful control information kept and how is it represented in terms of the NMWG XML schema?

Due to the extensive code base of the software the only question that was concisely answered is 3a. A Measurement Point service wraps around existing monitoring tools by either directly invoking them or by relying on storage containers in which such tools first write collected data. Data is then normalized with regards to the NMWG XML schema. The Measurement Point service also defines mechanisms for pushing data towards Measurement Archives. However, the different implementation of Measurement Archives do not interpret the entire NMWG schema space, but rather define hard-coded routines through which underlying measurement data is referenced to subspaces of the schema. As a result, services such as the SNMP Measurement Archive cannot interpret objects from the entire SNMP OID tree but rather target the exposure of interface statistics alone through archive-specific query routines. Likewise, RRD Measurement Archives do not treat RRD databases in a generic fashion by first building a profile of the underlying Round-Robin Archives of the database and normalizing it in the scope of an NMWG measurement schema.

In order to answer the remaining questions, contact was established with one of ESnet's perfSONAR-PS developers. After sharing the purpose of our research and the concerns in terms of extensibility, it was established that implementing a new Measurement Archive is not feasible within the time limitations of the project.

5 GreenSONAR

Certain functional requirements need to be met in order to implement a scalable energy-aware networking system. While Chapter 4 exploers whether perfSONAR-PS is a suitable entity for serving the role of data dissemination within such a system, this chapter examines more closely what are the underlying functional requirements of the system as a whole by using perfSONAR-NC Measurement Archives as a testbed.

5.1 Blueprint of energy-aware networking

By assuming that a data dissemination model is in place, attention can be given to remaining underlying components of an energy-aware networking solution. Notably, the core of such a system is the capability of putting different types of metrics in context of one another so that an energy profile can be created for a network node. As discussed in Chapter 3 this can be achieve through a formula the input of which is based on:

- per-port utilization readings of a node, and
- node energy consumption readings

Various data sources exist for these metrics. Although port utilization is usually monitored by requesting readings from the network node itself, it is more common to have a PDU as the data source of energy readings although a limited number of network devices offer the tracking of such values via their command line. Moreover, establishing communication with the data sources can be done through different, locally-regulated methodologies. A scalable solution to this is therefore to abstract the operations of accessing, reading and storing data from location-specific data sources into routines separate from the data dissemination system. Therefore, in Figure 5.1 the two metrics are depicted as RRD and SNMP entities.

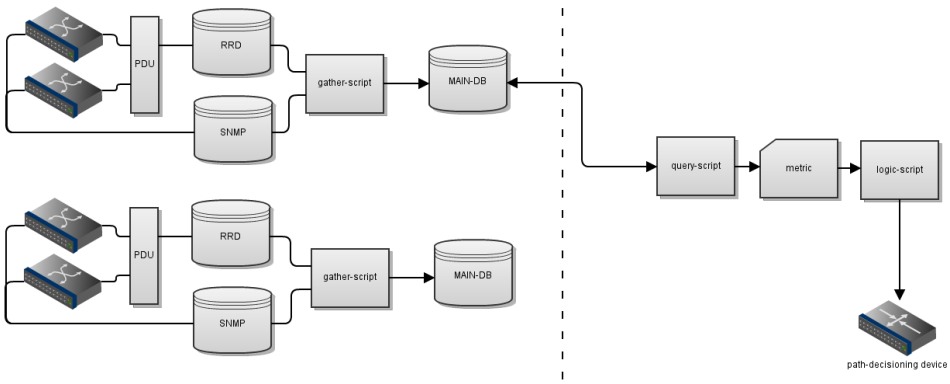


Figure 5.1: GreenSONAR test system

The next component of the system is the Measurement Archive which can be expressed as the combination of the "gather script" and "MAIN-DB" components from the aforementioned figure. An important design consideration for this component is whether it computes the energy weight of local devices itself and then servers a tuple

of device network identifiers and energy weights or whether it provides such needed data to clients to do the computation instead. The former approach however would obscure certain parts of the data by performing aggregation.
The dotted vertical line in the figure is meant to delineate two different administrative domain. The arrow inbetween the "MAIN-DB" and "query-script" components can be perceived as Services Layer provided by perfSONAR NMSOA framework. A client requesting the data can then be used for modifying the routing or forwarding logic of network nodes part of a remote domain. Such a client is constructed by extending perfSONAR's query classes. Various technologies can aid the process of modifying network operations with regards to the networking features supported by local devices:

- SNMP set queries - SNMP queries can be used to modify forwarding logic by the modification of VLANs a switchport is assigned to via the Q-BRIDGE-MIB information base. Also, SNMP queries can be used to modify the routing logic of network devices both in terms of static and dynamically-learned routes
- OpenFlow - As the OpenFlow protocol makes a great deal of distinction inbetween the routing/forwarding plane and the data plane such features are already built into the protocol
- OGF NSI-CS- For devices supporting the Open Grid Forum (OGF) Network Service Interface Connection Service (NSI-CS) standardized APIs can be used for performing such modifications

5.2 Test setup

In order to test what difficulties might arise with the underlying functional requirements of the system, a test model was created. The model takes into account the infrastructure present in the DAS-4 environment at the University of Amsterdam such as PDU energy readings that are stored in RRD archives, however the underlying data sources and how they are accessed is a design consideration rather than a restriction. The setup defines the following components:

- Measurement Archives for distributing RRD and SNMP data (Appendix C)
- A BASH script for gathering interface statistics of local devices and calculating the percentage utilization of their ports (Appendix C)

The two types of measurements served are linked together by using the Network Layer address of nodes. A client of the system, which knows in advance the network location of other participants, can establish the information offered by another participant by issuing a query that shows what archives exist at their location. A response of such a query takes the following form:

```
1  ...
2  <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
3   <data xmlns:default="http://stager.uninett.no/perfsonarnc/1.0">
4    <default:name xmlns="http://stager.uninett.no/psnc/1.0">145.100.102.114_snmp</
         default:name>
5    <default:name xmlns="http://stager.uninett.no/psnc/1.0">145.100.102.114_pdu</
         default:name>
6    <default:name xmlns="http://stager.uninett.no/psnc/1.0">145.100.102.115_snmp</
         default:name>
7    <default:name xmlns="http://stager.uninett.no/psnc/1.0">145.100.102.115_pdu</
         default:name>
8    </data>
9  </rpc-reply>
10 ...
```

By determining the types of archives available at a remote location, a locally-running client script can request the ones that represent devices used by the routing tables of the local routing infrastructure. The SNMP archive is implemented as a flat file. Entries part of the file take the following form:

Timestamp: UpInterfaceCount/TotalInterfaceCount, IF1=%Utilizatoin, IF2=%Utilizatoin, ...

The PDU Measurement archive provides the average of the latest entries appended to a Round-Robin database that describes the energy readings of the respective nodes.

6 Results

6.1 Metric

The derived metric as described in chapter 3 has been tested to gain insight on its applicability and granularity.

6.1.1 Scenario: Switch port

Enabling such test a number of assumptions were made:

- The device is a 100 Mb/s 24 port Switch ($Speed_{max} = 100 * 10^6$ b/s, $N_{ports} = 24$)
- The power consumption is constant $P_{total} = 123W$
- The utilization increases stepwise by 1%
- The temperature is at room temperature. $T = 300K$

Combining these assumptions yields data as presented in table 6.1. The leftmost column contains the calculated value for the *absolute current energy efficiency per port* $dB\varepsilon_{cpp}$.

timestamp	realpower	utilization	bps	abseff
1357588800	123	0.01	1000000	152.52
1357596000	123	0.02	2000000	149.51
1357603200	123	0.03	3000000	147.74
1357610400	123	0.04	4000000	146.49
1357617600	123	0.05	5000000	145.53
1357624800	123	0.06	6000000	144.73
1357632000	123	0.07	7000000	144.06
1357639200	123	0.08	8000000	143.48
1357646400	123	0.09	9000000	142.97
...				

Table 6.1: Constant power consumption, increasing utilization in 1% steps.

Figure 6.1 displays the data from table 6.1 in a graph that shows the curve of *absolute current energy efficiency per port* $dB\varepsilon_{cpp}$ against the *port utilization*.

The graph in figure 6.1 shows how the absolute energy efficiency rises, first steeply until about 20 - 30%, than more gradual towards 100%. On a side note: The absolute energy efficiency is measured Decibel (dB), therefore a decreasing value actually means a efficiency gain.
The steep raise in the first part of the graph can be explained by the fact that the power consumption of the device stays constant, while the utilization (and thereby the bit rate) raises.
In other cases regarding switches, when 802.3az (*Energy-Efficient Ethernet*) is not in place the proportions stay the same, because the fluctuation in the power consumption is negligible.
Conclusion: Energy efficiency wise a port should be utilized above 40%, if the power consumption is constant.

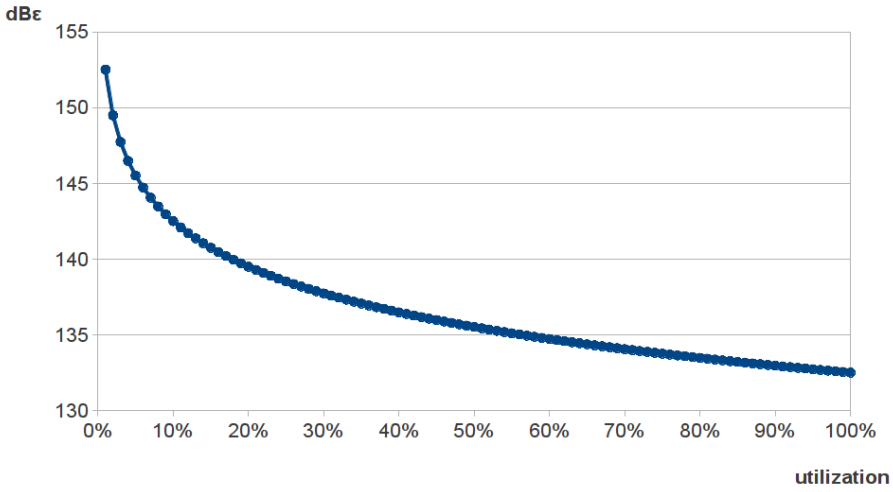


Figure 6.1: Absolute current energy efficiency per port against port utilization.

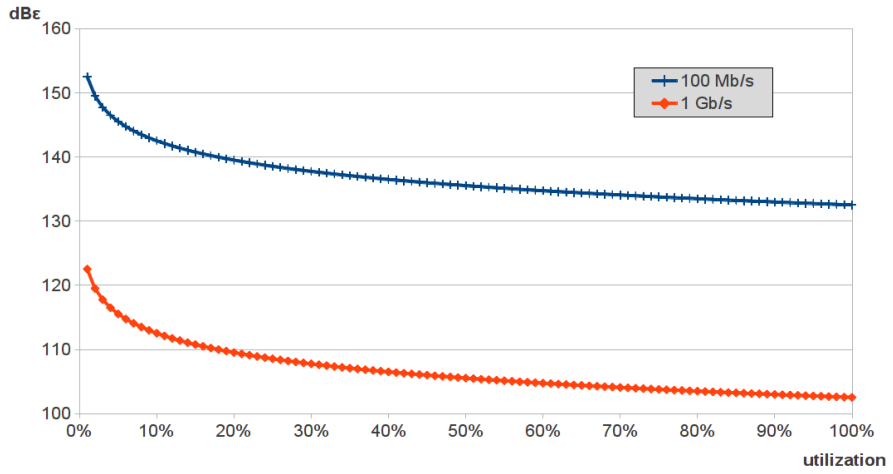


Figure 6.2: Comparison of absolute current energy efficiency of 100 Mb/s and 1 Gb/s port.

Figure 6.2 elucidates the difference in efficiency of a 100 Mb/s port to a 1 Gb/s port, where all other all other variables are shared. It substantiates and extends aforementioned conclusion, as the relationship between utilization and efficiency stays the same on one hand and on the other hand the general efficiency of the 1 Gb/s is distinguishable higher.

6.2 GreenSONAR

Evaluating whether perfSONAR-PS can be adapted to a system that enables energy-aware networking by providing ease of discovery and access to domain-specific measurement data through which energy profiles of network nodes can be constructed was comprised of the following research activities:

- Familiarization with the software and code-base analysis
- Implementation of a test modef based on perfSONAR NC

6.2.1 Familiarization with the perfSONAR software

The findings of the activity have established that perfSONAR-PS is a good starting point for the GreenSONAR project. It offers a centralized management resource through which one can represent measurements of various types irregardles of the underlying metric. The software provices a resource through which the various underlying monitoring tools part of Multi-Domain Monitoring systems can be centralized into a single administrative entity. The software is capable of distributing various per-

formance metrics due to the usage of the NMWG XML data normalisation schema. Such a high degree of flexibility in terms of data representation is an interesting feature as advances in network standards such as *802.3az* enable more granular energy measurements to be performed which would lead to modifications of the energy efficiency model calculation and thereby the underlying data that needs to be served.

A drawback of the perfSONAR-PS and MDM systems is the usage of a SOAP XML protocol. The protocol presents a large amount of network overhead traffic. When requesting data from Measurement Archives that, for example, serve tuples of netwrok addresses and energy weights, the management overhead constitues for more than 99% of network traffic. The application of such systems in demanding environments such as data centers where Layer 2 facilities undergo numerous chages in order to optimize traffic would interfere with the rest of the switching plane.

6.2.2 Implementation of a test modef based on perfSONAR NC

The test model aided the process of understanding the underlying functional dependencies of energy-aware networking. By developing a basic system that facilitates the distribution of energy and network statistics, we examined what other technologies need to be in place to influence the forwarding plane of network devices so that green paths can be constructed. The second important finding has to do with the fact that calculating *absolute energy efficiency* is a computationally-expensive task the complexity of which grows proportionally to the number of switchports in a network.

7 Conclusion

To conclude this report the research questions are answered separatly and pointers are given as towards what further research can be conducted based on the elaborated results.

"What metrics need to be considered in order to build energy profiles of networking devices and how can such data be published by using distributed multi-domain monitoring systems."

The results provided in section 6.1 show that our proposed metric does calculate granular and distinguishable results on a per port base. The attempt to include as much infrastructure as possible succeded in thusfar that the metric is calculated from readouts that can reasonably be assumed to be available for any SNMP-enabled device that is connected to a PDU.
Based on the results of our proposed small-scale implementation (section 6.2) we furthermore regard it possible to make available the metric using a distributed multi-domain monitoring system.

"Is perfSONAR-PS a suitable architecture to achieve energy profiling of computational devices, and what are the necessary steps to be undertaken to evolve perfSONAR-PS in a system we can call 'GreenSONAR'?"

As pointed out above, the distribution of energy profiles in general is possible. perfSONAR-PS does theoretically provide all requirements, but as adressed in Section 6.2 the infrastructural scope in which the system would be used needs to be closely considered. The introduction of perfSONAR-PS in high-demand networks might result in scalability issues in case the forwarding plane of the network relies on data that needs to be obtained from perfSONAR-PS Measurement Archives.

7.1 Future Research

7.1.1 Metric

In future the new metric needs to be tested more thoroughly to determine the exact range of its application. Such could be done by a (small scale) implementation that, within a longer period of time, makes path decisions based on the metric. From the gained data the actual efficiency gain determined.
There is also room to expand on how the metric could be adapted towards e.g. cpu-load instead of utilization to suit the performance readings of devices that are less related to utilization in terms of bit rate.

7.1.2 GreenSONAR

In order to implement new Measurement Archive(s) for perfSONAR-PS capable of publishing metrics that can be used to devise energy profiles for network nodes, a close collaboration with ESnet and Internet2 would be needed. This can be achieved by relying on the perfSONAR-PS mailing list available at:

https://lists.internet2.edu/sympa/info/performance-announce

The application of perfSONAR-PS in high demand networks would not scale well due to the large network overhead presented by the underlying SOAP XML protocol. In such environments, the usage of perfSONAR NC can be more beneficial as its communication protocol is based of NETCOF which offers various flexible modifications of RPC procedures for the purpose of minimizing network payloads.

8 Acknowledgments

We want to thank our supervisors Dr. Paola Grosso, Hao Zhu and Karel van der Veldt from the System and Network Engineering research group of the University of Amsterdam. They have been continuously providing feedback and made time for us seemingly regardless of their own agendas. They also introduced us to the System and Network Engineering Research Group and the interesting research area of Green-IT.
Furthermore we would like to thank the following people (in no particular order): Arne Øslebø from The UNINETT Group for help/feedback regarding perfSONAR-NC, Freek Dijkstra from SARA for providing access to SNMP and PDU readings, as well as insight into energy readings and Andy Lake from ESnet for feedback/help on perfSONAR-PS.

Bibliography

[1] Boote, J., Hanemann A. Kudarimoti L. Louridas P. Marta L. Michael M. Simar N. Tsompanidis I. 2007. Quality Assurance in perfSONAR Release Management.

[2] Christensen, K., Reviriego, P., Nordman, B., Bennett, M., Mostowfi, M., & Maestro, J.A. 2010. IEEE 802.3az: the road to energy efficient ethernet. *Communications Magazine, IEEE*, **48**(11), 50 –56.

[3] (DANTE), Domenico Vicinanza. 2012. perfSONAR MDM - perfSONAR PS Comparison. GÉANT.

[4] Doran, Peter T., & Zimmerman, Maggie Kendall. 2009. Examining the Scientific Consensus on Climate Change. *Eos, Transactions American Geophysical Union*, **90**(3), 22–23.

[5] Hanemann, A., Liakopoulos, A., Molina, M., & Swany, D.M. 2006. A study on network performance metrics and their composition. *Campus-Wide Information Systems*, **23**(4), 268–282.

[6] Parker, M.C., & Walker, S.D. 2011. Roadmapping ICT: An Absolute Energy Efficiency Metric. *Optical Communications and Networking, IEEE/OSA Journal of*, **3**(8), A49 –A58.

[7] Pavlov, D., & Soeurt, J. 2012. *Green Computing in IEEE 802.3az-enabled Clusters*. M.Phil. thesis, Universiteit van Amsterdam, the Netherlands.

[8] Raghavan, Barath, & Ma, Justin. 2011. The energy and emergy of the internet. *Pages 9:1–9:6 of: Proceedings of the 10th ACM Workshop on Hot Topics in Networks*. HotNets-X. New York, NY, USA: ACM.

[9] Sampaio, L., Koga, I., Costa, R., Monteiro, H., Monteiro, J.A.S., Vetter, F., Fernandes, G., & Vetter, M. 2007 (sept.). Implementing and Deploying Network Monitoring Service Oriented Architectures: Brazilian National Education and Research Network Measurement Experiments. *Pages 28 –37 of: Network Operations and Management Symposium, 2007. LANOMS 2007. Latin American*.

[10] Zhu, H., van der Veldt, K., Grosso, P., Zhao, Z., Liao, X., & de Laat, C. 2012. Energy-aware semantic modeling in large scale infrastructures. *In: The IEEE International Conference on Green Computing and Communications, GreenCom*, vol. 2012.

A Abbreviations

API Advanced Programming Interface *or* Application programming interface

AS Authentication Service

BWCTL Bandwidth Control

CLI Command-line interface

GGF Global Grid Forum

LS Lookup Service

MA Measurement Archive

MP Measurement Point

NM-WG Network Measurement Working Group

NMS Network Monitoring System

NMSOA Network Monitoring Service-Oriented Architecture

NOC Network operations center

NREN National Research and Education Network

NSI Network Services Interface

OID object identifier

OGF Open Grid Forum

PDU Power Distribution Unit

perfSONAR PERFormance Service Oriented Network monitoring ARchitecture

perfSONAR MDM $\sim$ Multi-Domain Monitoring

perfSONAR NC $\sim$ Netconf

perfSONAR-PS $\sim$ Perl Services

RRD Round-Robin Database

SNMP Simple Network Managment Protocol

SOA Service-Oritend Architecture

SOAP Simple Object Access Protocol

TCL Tool Command Language

TS Transformation Service

WS Web Service

B Test data

Table B.1: 100 Mb/s switch(port): Constant power consumption, increasing utilizationn in 1% steps.

timestamp	realpower	utilisation	bps	abseff
1357588800	123	0.01	1000000	152.52
1357596000	123	0.02	2000000	149.51
1357603200	123	0.03	3000000	147.74
1357610400	123	0.04	4000000	146.49
1357617600	123	0.05	5000000	145.53
1357624800	123	0.06	6000000	144.73
1357632000	123	0.07	7000000	144.06
1357639200	123	0.08	8000000	143.48
1357646400	123	0.09	9000000	142.97
1357653600	123	0.1	10000000	142.52
1357660800	123	0.11	11000000	142.10
1357668000	123	0.12	12000000	141.72
1357675200	123	0.13	13000000	141.38
1357682400	123	0.14	14000000	141.05
1357689600	123	0.15	15000000	140.75
1357696800	123	0.16	16000000	140.47
1357704000	123	0.17	17000000	140.21
1357711200	123	0.18	18000000	139.96
1357718400	123	0.19	19000000	139.73
1357725600	123	0.2	20000000	139.51
1357732800	123	0.21	21000000	139.29
1357740000	123	0.22	22000000	139.09
1357747200	123	0.23	23000000	138.90
1357754400	123	0.24	24000000	138.71
1357761600	123	0.25	25000000	138.54
1357768800	123	0.26	26000000	138.37
1357776000	123	0.27	27000000	138.20
1357783200	123	0.28	28000000	138.04
1357790400	123	0.29	29000000	137.89
1357797600	123	0.3	30000000	137.74
1357804800	123	0.31	31000000	137.60
1357812000	123	0.32	32000000	137.46
1357819200	123	0.33	33000000	137.33
1357826400	123	0.34	34000000	137.20
1357833600	123	0.35	35000000	137.07
1357840800	123	0.36	36000000	136.95
1357848000	123	0.37	37000000	136.83
1357855200	123	0.38	38000000	136.72
1357862400	123	0.39	39000000	136.60
1357869600	123	0.4	40000000	136.49
1357876800	123	0.41	41000000	136.39
1357884000	123	0.42	42000000	136.28
1357891200	123	0.43	43000000	136.18
1357898400	123	0.44	44000000	136.08
1357905600	123	0.45	45000000	135.98
1357912800	123	0.46	46000000	135.89

Table B.1: (continued)

timestamp	realpower	utilisation	bps	abseff
1357920000	123	0.47	47000000	135.79
1357927200	123	0.48	48000000	135.70
1357934400	123	0.49	49000000	135.61
1357941600	123	0.5	50000000	135.53
1357948800	123	0.51	51000000	135.44
1357956000	123	0.52	52000000	135.36
1357963200	123	0.53	53000000	135.27
1357970400	123	0.54	54000000	135.19
1357977600	123	0.55	55000000	135.11
1357984800	123	0.56	56000000	135.03
1357992000	123	0.57	57000000	134.96
1357999200	123	0.58	58000000	134.88
1358006400	123	0.59	59000000	134.81
1358013600	123	0.6	60000000	134.73
1358020800	123	0.61	61000000	134.66
1358028000	123	0.62	62000000	134.59
1358035200	123	0.63	63000000	134.52
1358042400	123	0.64	64000000	134.45
1358049600	123	0.65	65000000	134.39
1358056800	123	0.66	66000000	134.32
1358064000	123	0.67	67000000	134.25
1358071200	123	0.68	68000000	134.19
1358078400	123	0.69	69000000	134.13
1358085600	123	0.7	70000000	134.06
1358092800	123	0.71	71000000	134.00
1358100000	123	0.72	72000000	133.94
1358107200	123	0.73	73000000	133.88
1358114400	123	0.74	74000000	133.82
1358121600	123	0.75	75000000	133.76
1358128800	123	0.76	76000000	133.71
1358136000	123	0.77	77000000	133.65
1358143200	123	0.78	78000000	133.59
1358150400	123	0.79	79000000	133.54
1358157600	123	0.8	80000000	133.48
1358164800	123	0.81	81000000	133.43
1358172000	123	0.82	82000000	133.38
1358179200	123	0.83	83000000	133.32
1358186400	123	0.84	84000000	133.27
1358193600	123	0.85	85000000	133.22
1358200800	123	0.86	86000000	133.17
1358208000	123	0.87	87000000	133.12
1358215200	123	0.88	88000000	133.07
1358222400	123	0.89	89000000	133.02
1358229600	123	0.9	90000000	132.97
1358236800	123	0.91	91000000	132.93
1358244000	123	0.92	92000000	132.88
1358251200	123	0.93	93000000	132.83
1358258400	123	0.94	94000000	132.78
1358265600	123	0.95	95000000	132.74
1358272800	123	0.96	96000000	132.69
1358280000	123	0.97	97000000	132.65
1358287200	123	0.98	98000000	132.60
1358294400	123	0.99	99000000	132.56
1358301600	123	1	100000000	132.52

Table B.2: 1 Gb/s switch(port): Constant power consumption, increasing utilizationn in 1% steps.

timestamp	**realpower**	**utilisation**	**bps**	**abseff**
timestamp	realpower	utilization	bps	abseff
1357588800	123	0.01	1000000000	122.52
1357596000	123	0.02	2000000000	119.51
1357603200	123	0.03	3000000000	117.74
1357610400	123	0.04	4000000000	116.49
1357617600	123	0.05	5000000000	115.53
1357624800	123	0.06	6000000000	114.73
1357632000	123	0.07	7000000000	114.06
1357639200	123	0.08	8000000000	113.48
1357646400	123	0.09	9000000000	112.97
1357653600	123	0.10	10000000000	112.52
1357660800	123	0.11	11000000000	112.10
1357668000	123	0.12	12000000000	111.72
1357675200	123	0.13	13000000000	111.38
1357682400	123	0.14	14000000000	111.05
1357689600	123	0.15	15000000000	110.75
1357696800	123	0.16	16000000000	110.47
1357704000	123	0.17	17000000000	110.21
1357711200	123	0.18	18000000000	109.96
1357718400	123	0.19	19000000000	109.73
1357725600	123	0.20	20000000000	109.51
1357732800	123	0.21	21000000000	109.29
1357740000	123	0.22	22000000000	109.09
1357747200	123	0.23	23000000000	108.90
1357754400	123	0.24	24000000000	108.71
1357761600	123	0.25	25000000000	108.54
1357768800	123	0.26	26000000000	108.37
1357776000	123	0.27	27000000000	108.20
1357783200	123	0.28	28000000000	108.04
1357790400	123	0.29	29000000000	107.89
1357797600	123	0.30	30000000000	107.74
1357804800	123	0.31	31000000000	107.60
1357812000	123	0.32	32000000000	107.46
1357819200	123	0.33	33000000000	107.33
1357826400	123	0.34	34000000000	107.20
1357833600	123	0.35	35000000000	107.07
1357840800	123	0.36	36000000000	106.95
1357848000	123	0.37	37000000000	106.83
1357855200	123	0.38	38000000000	106.72
1357862400	123	0.39	39000000000	106.60
1357869600	123	0.40	40000000000	106.49
1357876800	123	0.41	41000000000	106.39
1357884000	123	0.42	42000000000	106.28
1357891200	123	0.43	43000000000	106.18
1357898400	123	0.44	44000000000	106.08
1357905600	123	0.45	45000000000	105.98
1357912800	123	0.46	46000000000	105.89
1357920000	123	0.47	47000000000	105.79
1357927200	123	0.48	48000000000	105.70
1357934400	123	0.49	49000000000	105.61
1357941600	123	0.50	50000000000	105.53
1357948800	123	0.51	51000000000	105.44
1357956000	123	0.52	52000000000	105.36
1357963200	123	0.53	53000000000	105.27
1357970400	123	0.54	54000000000	105.19
1357977600	123	0.55	55000000000	105.11
1357984800	123	0.56	56000000000	105.03
1357992000	123	0.57	57000000000	104.96
1357999200	123	0.58	58000000000	104.88

Table B.2: (continued)

timestamp	realpower	utilization	bps	abseff
1358006400	123	0.59	59000000000	104.81
1358013600	123	0.60	60000000000	104.73
1358020800	123	0.61	61000000000	104.66
1358028000	123	0.62	62000000000	104.59
1358035200	123	0.63	63000000000	104.52
1358042400	123	0.64	64000000000	104.45
1358049600	123	0.65	65000000000	104.39
1358056800	123	0.66	66000000000	104.32
1358064000	123	0.67	67000000000	104.25
1358071200	123	0.68	68000000000	104.19
1358078400	123	0.69	69000000000	104.13
1358085600	123	0.70	70000000000	104.06
1358092800	123	0.71	71000000000	104.00
1358100000	123	0.72	72000000000	103.94
1358107200	123	0.73	73000000000	103.88
1358114400	123	0.74	74000000000	103.82
1358121600	123	0.75	75000000000	103.76
1358128800	123	0.76	76000000000	103.71
1358136000	123	0.77	77000000000	103.65
1358143200	123	0.78	78000000000	103.59
1358150400	123	0.79	79000000000	103.54
1358157600	123	0.80	80000000000	103.48
1358164800	123	0.81	81000000000	103.43
1358172000	123	0.82	82000000000	103.38
1358179200	123	0.83	83000000000	103.32
1358186400	123	0.84	84000000000	103.27
1358193600	123	0.85	85000000000	103.22
1358200800	123	0.86	86000000000	103.17
1358208000	123	0.87	87000000000	103.12
1358215200	123	0.88	88000000000	103.07
1358222400	123	0.89	89000000000	103.02
1358229600	123	0.90	90000000000	102.97
1358236800	123	0.91	91000000000	102.93
1358244000	123	0.92	92000000000	102.88
1358251200	123	0.93	93000000000	102.83
1358258400	123	0.94	94000000000	102.78
1358265600	123	0.95	95000000000	102.74
1358272800	123	0.96	96000000000	102.69
1358280000	123	0.97	97000000000	102.65
1358287200	123	0.98	98000000000	102.60
1358294400	123	0.99	99000000000	102.56
1358301600	123	1.00	100000000000	102.52

C Code listings

The following code snipplets are also available online at:

http://code.google.com/p/greensonar/source/browse/#svn%2Ftrunk%2Fcode

The files in the perfSONARNC folder offer a complete release that can be deployed by running the "pncs" PHP file part of the main folder. In order to simulate test results, one can use the following queries. Further information can be found on the perfSONARNC website: queries. Further information can be found on the perfSONARNC website:

https://trac.uninett.no/perfsonarnc

Reading out the available Measurement Archives can be done with the following query:

```
1  code/perfSONARNC/maquery/maquery.php -q "//pn:ma/pn:name" localhost
```

Queries for reading out Measurement Archives:

```
1  code/perfSONARNC/maquery/maquery.php -r 1 192.168.1.1_snmp snmp table
       standard -o 1 localhost -f "2013-02-06 09:00" -t "2013-02-06 09:10"
2
3  code/perfSONARNC/maquery/maquery.php -r 1 192.168.1.1_pdu pdu table standard
       -o 1 localhost -f "2013-02-06 09:00" -t "2013-02-06 09:10"
```

SNMP Measurement Archive listing:

snmpma1.class.php

```
1  <?php
2
3  /*
4   A very simple syslog MA that provides access to entries in the syslog file
5   in 5 minute intervals. This MA is part of the tutorial on how to implement
6   your own MAs
7   */
8
9  require_once("ma.abs.php");
10
11 class snmpma1 extends MA {
12
13   var $data;
14
15   function getName()
16   {
17     return "192.168.1.1_snmp";
18   }
19
20     function getTableorSingleData($source,$timeperiods,$report,$obspoints,$type,$
         view,$filter,$sort,$rowlimit,$arguments)
21     {
```

```
22              $ret=array();
23              $count=0;
24              foreach($timeperiods as $tp)
25              {
26                      $ret[$tp['resolution']][$tp['timestamp']][1]=array();
27                      $interfaceStats=fopen("/home/madave/Documents/perfSONARNC/
                            snmpma1/testSyslog.db","r");
28                      while($str=fgets($interfaceStats))
29                      {
30                              $time = $this->getTime($str);
31                              preg_match("/:.*/",$str,$m);
32                              if($rowlimit==0 or $rowlimit>$count++)
33                                      $ret[$tp['resolution']][$tp['timestamp'
                                            ]][1][]=array($time,$m[0]);
34                      }
35                      fclose($interfaceStats);
36              }
37              return $ret;
38      }
39
40      function getDatasources()
41      {
42        return array('192.168.1.1_snmp'=>"192.168.1.1_snmp");
43      }
44
45      function getObspoints($source)
46      {
47        return array(array('id'=>1,
48              'name'=>"localhost"));
49      }
50
51
52        function getTimeInfo($source)
53        {
54                $first=$this->getTime(`head /home/madave/Documents/perfSONARNC/
                      snmpma1/testSyslog.db -n 1`);
55                $last=$this->getTime(`tail /home/madave/Documents/perfSONARNC/snmpma1
                      /testSyslog.db|tail -n 1`);
56
57                $ret=array();
58                $ret[]=array('id'=>1,
59                                        'name'=>"5 min",
60                                        'duration'=>300,
61                                        'plot_from'=>"",
62                                        'first'=>$first,
63                                        'last'=>$last);
64                $ret[]=array('id'=>2,
65                                        'name'=>"Hour",
66                                        'duration'=>3600,
67                                        'plot_from'=>1,
68                                        'first'=>$first,
69                                        'last'=>$last);
70                return $ret;
71        }
72
73        private function getTime($str)
74        {
75                preg_match("/.*:/", $str, $result);
76                return $result[0];
77        }
78
79  }
80  ?>
```

RRD Measurement Archive listing:

pduma1.class.php

```
1   <?php
2   require_once("ma.abs.php");
3
4   class pduma1 extends MA {
5
6     var $data;
7
8     function getName()
9     {
10      return "192.168.1.1_pdu";
11    }
12
13    function getTableorSingleData($source,$timeperiods,$report,$obspoints,$type,$
          view,$filter,$sort,$rowlimit,$arguments)
14    {
15      $ret=array();
16      $count=0;
17      foreach($timeperiods as $tp)
```

```
18      {
19        $ret[$tp['resolution']][$tp['timestamp']][1]=array();
20        $pduReading=`rrdtool lastupdate outlet17.rrd | cut -d ' ' -f 1,2 | tail -n
              +3`;
21        while($str=fgets($pduReading))
22        {
23          $time = $this->getTime($str);
24
25          if($rowlimit==0 or $rowlimit>$count++)
26            $ret[$tp['resolution']][$tp['timestamp']][1][]=array($time,substr($str
                  ,11));
27        }
28        fclose($interfaceStats);
29      }
30      return $ret;
31    }
32
33    function getDatasources()
34    {
35      return array('pduma1'=>"pduma1");
36    }
37    function getObspoints($source)
38    {
39      return array(array('id'=>1,
40            'name'=>"localhost"));
41    }
42
43
44    function getTimeInfo($source)
45    {
46      $first=$this->getTime(`head /home/madave/Documents/perfSONARNC/syslogma/
              testSyslog.db -n 1`);
47      $last=$this->getTime(`tail /home/madave/Documents/perfSONARNC/syslogma/
              testSyslog.db|tail -n 1`);
48
49      $ret=array();
50      $ret[]=array('id'=>1,
51            'name'=>"5 min",
52            'duration'=>300,
53            'plot_from'=>"",
54            'first'=>$first,
55            'last'=>$last);
56      $ret[]=array('id'=>2,
57            'name'=>"Hour",
58            'duration'=>3600,
59            'plot_from'=>1,
60            'first'=>$first,
61            'last'=>$last);
62      print "getTimeInfo returns:";
63      print_r ($ret);
64      return $ret;
65    }
66
67    private function getTime($str)
68    {
69      preg_match("/^\d{10}/", $str, $result);
70      return $result[0];
71    }
72  }
73  ?>
```

Script that gathers interface utilization by using SNMP:

interface_percentage_utilization.sh

```
1   #!/bin/bash
2
3   # This script creates a measurement archive that describes the following
        interface statistics
4   # with regards to time:
5   # - Count of all interfaces
6   # - Interfaces that have their protocol up (Operational status of up)
7   # - Per interface percentage utilizatoin
8   #
9   # An entry(one line) from the archive takes the following form:
10  #
11  #         Timestamp: UpInterfaceCount/TotalInterfaceCount, Interface1=
        PercentageUtilization, ...
12  #
13  # The current utilization of a port is expressed in percentiles by the following
        formula:
14  #
15  #
16  #           Max(Delta(InOctets), Delta(OutOctets) * 8(20% management) * 100%
17  #           ---------------------------------------------------------------
18  #                         ProbesInterval * InterfaceCapacity
```

```
#
#
# The script takes as configurable parameters:
#
# - IP(s) of device(s)
# - SNMP community of devices(should be shred)
# - Interval inbetween the two probes needed by the calculation. Should be large enough (30s)
#   to accommodate for the lack of tracking time inbetween the different SNMP queries


#Data containers of local devices whose port utilization stats need to be obtained
device=( "192.168.1.1" "192.168.1.2" )
#SNMP community
community="public"
#Time between the two probes used in measuring interface utilization.
probeTime=30
#Statistics storage location
outputDir="/opt/perfSONARNC/SNMP\ archives/"


#
for ipAddrs in "${device[@]}"
do
  :
  #Get the operational status of interfaces
  ifOper=`snmpwalk -v 2c -c $community ${device[$i]} ifOperStatus`
  #Get the capacity of interfaces
  ifSpeed=`snmpwalk -v 2c -c $community ${device[$i]} ifSpeed`
  #Count how many interfaces this device has
  interfaceCount=`echo "$ifOper" | wc -l`

  #Do two consequtive probes of IF-MIB::ifInOctets and IF-MIB::ifOutOctets 30s apart so that utilization can be calculated
  ifInOctets[0]=`snmpwalk -v 2c -c  $community $ipAddrs ifInOctets`
  ifOutOctets[0]=`snmpwalk -v 2c -c  $community $ipAddrs ifOutOctets`
  sleep 30s
  ifInOctets[1]=`snmpwalk -v 2c -c  $community $ipAddrs ifInOctets`
  ifOutOctets[1]=`snmpwalk -v 2c -c  $community $ipAddrs ifOutOctets`


    #Build up an array of up interfaces by reading the contents of ifOperStatus
    index=0
    while IFS= read -r interfaceStatus
    do
        if [[ "$interfaceStatus" =~ "up" ]]; then
            #Extract interface index
            ifOperStatus[$index]=`echo "$interfaceStatus" | grep -oiE '\..*\s=' | sed 's/[^0-9]//g'`
            index=$((index + 1))
        fi
    done <<< "$ifOper"

    #Build up an array representing utilization of a port in the form intefaceIndex:PercentageUtilization
    for ifIndex in "${ifOperStatus[@]}"
    do
        :
        #Calculate input octets in interval
        probe1_inOctets=`echo -n "${ifInOctets[0]}" | grep -i "ifInOctets." $ifIndex | cut -d ' ' -f 4`
        probe2_inOctets=`echo -n "${ifInOctets[1]}" | grep -i "ifInOctets." $ifIndex | cut -d ' ' -f 4`
        measuredInputOctets="$(($probe2_inOctets - $probe1_inOctets))"

        #Calculate output octets in interval
        probe1_outOctets=`echo -n "${ifOutOctets[0]}" | grep -i "ifOutOctets." $ifIndex | cut -d ' ' -f 4`
        probe2_outOctets=`echo -n "${ifOutOctets[1]}" | grep -i "ifOutOctets." $ifIndex | cut -d ' ' -f 4`
        measuredOutputOctets="$(($probe2_outOctets - $probe1_outOctets))"

        #Get the max of in/out octets in interval or just save one of them if equal load
        if [[ $measuredInputOctets > $measuredOutputOctets ||
            $measuredInputOctets -eq $measuredOutputOctets ]]; then
                maxOctets="$measuredInputOctets"
        else [[ measuredInputOctets < outUtilization ]]
                maxOctets="$measuredOutputOctets"
        fi

        #Get the capacity of this interface
        ifCapacity=`echo $ifSpeed | cut -d ' ' -f 4`

        #Calculate percentage utilization and store it
        num="$(($maxOctets*8*100))"
        denum="$(($probeTime*$ifCapacity))"
```

```
96          ifPercentUtilization[$ifIndex]=`printf "%.2f\n" $(echo "scale=2; $num/
                $denum" | bc)`
97      done
98          #Write statistics to file
99          echo -n `date +%s`: "${#ifOperStatus[@]}"/"$interfaceCount", "` >>
                $outputDir"/"$ipAddrs
100         for i in ${!ifPercentUtilization[*]}
101         do
102                 :
103                 echo -n $i"="${ifPercentUtilization[$i]}", " >> $outputDir"/"
                        $ipAddrs
104         done
105                 echo >> $outputDir"/"$ipAddrs
106 done
```