A perfSONAR-compliant Looking Glass for the Géant2 Multi-Domain Monitoring Service

S. Verstichel, S. Latré, S. Melis, F. De Turck, B. Dhoedt, P. Demeester
Department of Information Technology
Ghent University, Ghent, Belgium
Gaston Crommenlaan, 8 bus 201
9050 Ghent
fax: +32 9 331 4899 - tel: +32 9 331 4981
stijn.verstichel@intec.ugent.be

Abstract *The absence of some form of network connectivity in the current daily life is almost unthinkable. In order to provide and maintain this connectivity, there is a need for distributed cross-border monitoring to cope with the amount of data involved. A service has therefore been developed making such an exchange possible, using a standardized protocol and data format, based on a common framework for distributed monitoring, perfSONAR. The monitoring service developed provides the ability to retrieve static configuration information from routers and switches in the network. Security measures have also been integrated, in order to protect the access to the routers from malicious attacks. We have found that by using this service together with other webservices in the framework, a clearer view about the status of the network and that of its neighbours can be obtained. These standards also make it possible for other parties to develop specific analysis and visualisation tools using these services as a data source.*

Keywords: distributed network monitoring, Looking Glass, perfSONAR

1 Introduction

Géant2[?] is a project in the scope of EC's Sixth R&D Framework Programme. It is jointly funded by the European Commission and the National Research Networks (NREN) that will be connected to the multi-gigabit backbone to be created. The main objectives are to plan, build and operate this multi-gigabit pan-European backbone research network. Research will also be carried out into the development and deployment of new networking technologies and monitoring tools. It will support users needing advanced networking requirements and examine the future of research networking. Also, initiatives targeted at closing the 'digital divide' will be pursued.

perfSONAR is the all-embracing term for a consortium of organizations with the aim of building a network monitoring framework, capable of interworking through a multitude of networks. It is also a collection of services and a communication protocol, based on SOAP XML[?] messages and following the Open Grid Forum (OGF)[?] Network Measurement Working Group (NM-WG)[?]. An example implementation of a set of services is also provided by the perfSONAR consortium. perfSONAR does not however create new measurement tools, but mostly functions as a middleware wrapper around an existing measurement tool to create an interface to the tool, according to the perfSONAR protocol specified.

Using this framework as a basis, we have created a webservice that provides the functionality of a looking glass. In a networking context, the term "looking glass" is used to represent tools that allow users to look at the configuration of network equipment, such as routers and switches, consult routing tables from different points in the network, perform traceroute analysis starting from a problem point in the network, instead of from the workstation towards the problem. It generally permits the retrieval of show-like commands, such as "show ip bgp".

The following Section ?? gives related work and more information and background about the perfSONAR framework used as a basis to implement the looking glass webservice. The functionality pro-

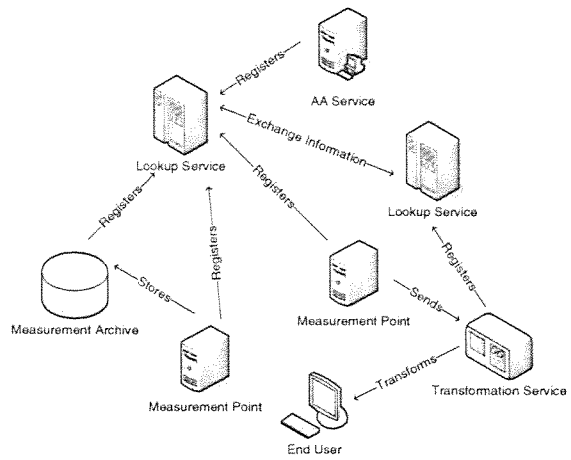

Figure 1: Services and their interactions in the perfSONAR framework

vided by the looking glass webservice is detailed in Section ??. Section ?? details the architecture and internal workings of the webservice. A prototype font-end is presented in Section ?? and the performance is evaluated in Section ??. To conclude this paper, items for future work are identified in Section ??.

2 Related Work

Problem solving and performance monitoring across the boundaries of individual networks these days is rather difficult. Their is often no data exchange between the networks, and if such an interchange does exist, it is often achieved without using a standardized data format.

In order to achieve a more collaborating community and standardized data format, a collaboration called perfSONAR[?, ?] has been created. A number of standardized types of services have been defined. Their interactions are represented in Figure ??. To visualise the data obtained through these services, a number of visualisation clients are being developed.

Probably the most important type of services are the "Measurement Points". These services perform measurements on the network for pre-defined metrics, such as available bandwidth, delay or packet loss. It is this type of service, the looking glass webservice belongs to. However, in the perfSONAR framework the official name for this measurement point is the "SSH/Telnet Measurement Point". This name has been chosen in favour of the "Looking Glass Measurement Point", in order to be able to refer to the front-end as the "Looking Glass". The term SSH/Telnet used in the official name of the service refers to the access method the webservice uses to obtain the requested information.

The second most important type of service in the framework is that of the "Measurement Archive". As the name already suggests, it is an archive for storing historical information. Example services have been implemented, using a Round-Robin as well as a Relational Database back-end.

Two other types of services that perform procedures on measurements or give more context to the measurements are the "Transformation Service" and the "Topology Service". The former provides functionality for translating the format of a certain measurement into another measurement, without re-executing the measurement itself. An example of the transformation procedure this service can perform is the conversion of the resolution of the measured metrics or the aggregation of these metrics over a given period of time. The latter type of service can be used to obtain topological information about the network, and can therefore be used to have a clearer view about the actual inherent information in a given measurement.

Other types of services, not really for performing or storing measurements, have also been defined. These services are the "Lookup Service", which fulfills the task of a directory service. It stores the information about where and how to contact the other services in the framework. Also, an "Authorization and Authentication Service" is being developed in order to protect the data from malicious intruders.

A number of looking glass implementations already exist. Many of these can be found here[?]. However, all of them are implemented without keeping a standardized approach in mind. These tools, and more specifically the Rancid Looking Glass[?], have been used as a starting point for the development of the perfSONAR-compliant Looking Glass.

3 Overview

The "SSH/Telnet Measurement Point" service, as it is called, functions as the back-end for the perfSONAR complaint implementation of the looking glasses already available, but currently often implemented as stand-alone CGI/Perl scripts[?]. In short we can describe this "SSH/Telnet Measurement Point" service as a central contacting point inside a network able to retrieve information from

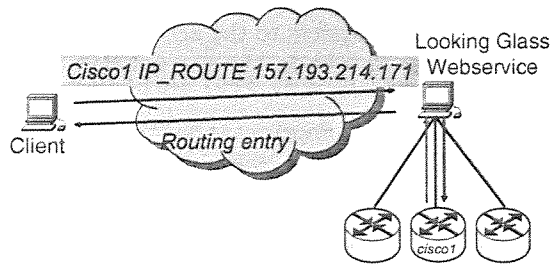

Figure 2: Collaboration between the actors

routers it has access to, using standard protocols such as SSH or Telnet. Mostly these commands are show-like commands for retrieving configuration information such as Routing Tables, Interface Configuration etc.

3.1 Basic functionality of the SSH/Telnet Measurement Point

A client wishing to obtain configuration or other show-like information from a particular network device should be contacting this measurement point to get the desired information. In order to satisfy the request of the client, the measurement point opens a Telnet or SSH connection, depending on the configuration, to that particular device and executes the command in order to retrieve the correct information. The output of the command executed on the router is captured and transmitted back to the client using perfSONAR messages. A comparison probably best used to describe the mechanism of this service, is the proxy-type architecture. The client does not have direct access to the routers and is therefore contacting this "SSH/Telnet Type Measurement Point" to obtain the information needed. The service then contacts the routers on behalf of the client. The return messages follow the reverse path. This means that the "SSH/Telnet Measurement Point" service is returning the results to the client on behalf of the underlying routers.

The intended collaboration between the actors in a normal functioning of the webservice is graphically illustrated in Figure ??.

3.2 Security aspects

The functionality described in the previous paragraph is very basic. Indeed, for a service to be called a "SSH/Telnet Measurement Point" service it is only necessary to provide the functionality described above. Nevertheless, some extra features have been incorporated into the implementation in order to make the service more robust and secure. If only the basic functionality would have been supported, this service would have been very susceptible to malicious attacks. This is because the commands supplied by the client to the service would then have been dispatched straight on to the underlying devices, without keeping in mind that these commands might crash that device.

The first of the extra features to be described here is that of the command-checking functionality of the commands that are to be executed on the device. This means only the commands pre-configured in the "SSH/Telnet Measurement Point" can be executed on the underlying routers. Using this mechanism, network administrators can be sure that only permitted commands are passed through.

A second feature existing next to the basic functionalities is that of parameter-checking capability. All parameters supplied as arguments to the command, are checked against a list of regular expressions. The command not satisfying the regular expressions are not passed on to the device. Again this reassures network administrators the needed protection against malicious requests. Also, a minimum and maximum number of arguments can be configured in the measurement point. Again, if the client does not supply the correct number of arguments or parameters, the command won't be executed on the underlying routers.

Another important feature to make sure that amount of requests does not overburden the underlying device is the ability to configure a caching period for results of commands executed recently. In some cases it is not necessary to have real-time information available for the looking glass. This certainly holds in situation where the looking glass is available to the wider public. In such cases it might be sufficient, to retrieve the information on the fly, when the information concerned is requested for the first time since a long period or the information is not cached at that moment. The results could then be cached for a configured period of time, so as to be able to quickly respond to a similar request within the caching period. It depends of course on the desires from the individual networks, if such a feature is needed and used. Therefore, the caching period can be configured at deploy-time, for each and every individual device. This also means that if needed, the caching period for some device can be completely different from other commands, or that no caching is performed at all.

The last extra feature that was provided in this

measurement point service is the ability for the persons responsible for the deployment of this service to specify the amount of requests that can be processed in any given time period. This is enforced through a token-bucket algorithm, where one can specify the minimum timeframe between two consecutive requests.

4 Architecture

A graphical representation of the architecture of the "SSH/Telnet Measurement Point" is given in Figure ?? and the details of the individual modules are detailed in the subsequent paragraphs.

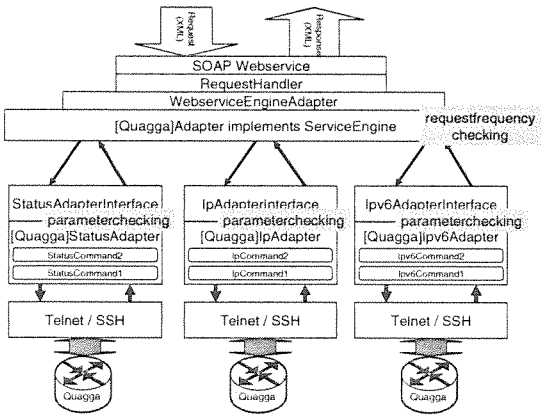

Figure 3: Architecture for the Looking Glass (SSH/Telnet MP) Webservice

4.1 WebServiceEngineAdapter

The entry point for the Telnet/SSH MP webservice is the WebserviceEngineAdapter, which is an extension of the RequestHandler, as already implemented in the perfSONAR framework. The RequestHandler is responsible for the first processing and filtering of the incoming request messages. Based upon the subject of the request message arriving, the RequestHandler decides which underlying ServiceEngine is capable of answering the incoming request. The underlying ServiceEngine for the Telnet/SSH MP, as previously mentioned, is the WebserviceEngineAdapter. This means that all requests arriving at the RequestHandler, addressing Telnet/SSH MP functionality are forwarded by the RequestHandler to an instance of the WerserviceEngineAdapter. The WebserviceEngineAdapter is then responsible for handling the request and returning an answer back to the RequestHandler, which is then in its turn responsible to encapsulate it in a SOAP message and returning it to the original querying actor. It is here the token-bucket algorithm is implemented to limit the number of consecutive requests being processed.

Another important feature provided is that the WebserviceEngineAdapter is able to dispatch multiple requests to different underlying ServiceEngines in parallel. This is certainly true in the case when in a single incoming request message, multiple commands on multiple devices are specified. The retrieval of every command of every device is encapsulated in a separate thread and these threads are executed in parallel. It is not until all results are returned from the devices or error situations are caught, that the WebserviceEngineAdapter concatenates all results in a single response message to be dispatched back to the original requestor.

4.2 [...]Adapter

Next in line is the Adapter. This Adapter is a class holding all business logic specific to a particular device. So, in the first implementation, there was an Adapter for a Quagga device and for a Cisco Router. At the moment Juniper devices are supported as well. More specifically Telnet based login to Quagga, Cisco and Junipers using username and password is supported. More secure login on Cisco and Juniper is provided as well. This SSH access can be username/password based or username/public-key based. One of the key things the Adapter is thus responsible for, and has the knowledge of, is the actual information for addressing and connecting to the underlying device.

4.3 [...]IP/IPv6/BGP/STATUSAdapter

The last level in the top-down breakdown of this architecture is the SubAdapter. This SubAdapter is responsible for a specific subset of queries supported. The current supported subsets are IP information, IPv6 information, BGP information and Status information. There is a specific reason why these subsets are created. Inside the SubAdapter, there is a list of available Commands. These will be discussed in the next subsection.

The individual adapters are responsible for the evaluation and approval of the supplied parameters. Some commands needs parameters or arguments before they can be executed correctly, e.g. the ip address as an argument to the command for

retrieving the routing entry table. The validation of the parameters can either be done through the application of regular expressions configured in the configuration file of the measurement point service, or can be done by a proprietary implementation of the interface

```
org.perfsonar.service.
measurementPoint.lookingGlassType.
engine.commands.
ParameterCheckerInterface
```

and naming the class as following:

```
<name of the device>Checker.java.
```

where the name of the device matches the one configured int the configuration file. An example of such an enforcing class is that for the device with the name

```
JuniperGeantAmsterdam
```

the checker class is called

```
JuniperGeantAmsterdamChecker.java
```

This class is then loaded at runtime during execution of the request. Depending on the answer of this checker class, the request will go to the router or not.

4.4 Command

The Command is merely a placeholder for all information necessary, to be able to fetch the configuration from the underlying device. The information contained in this Data Type is:

- The corresponding EventType
- The device specific command to be executed on the underlying device
- A list of parameters
- The result.

At the moment the result is only a container with the cleaned-up output the Device returns after executing the Command. If desired, some more intelligent functionality on the result could be provided, e.g. parsing of the output into some more meaningful classes.

4.5 Communication with the device

The actual communication with the device is at the moment provided by two Java-packages called J2SSH[?] and JTA[?], implementing the SSH and Telnet protocol. It uses standard Input/Output Streams to transmit commands to the device, and to collect the output from the device. It also provides functionality for scripting, where one can specify a sequence of expected commands and results from the device.

5 Front End

The "Looking Glass Client" being described here, is used as the visualisation of the behind the scenes active "SSH/Telnet Measurement Point". It serves as a perfSONAR integrated replacement of the currently available looking glasses written in Perl or CGI. The ultimate goal of this tool is to provide users the ability to obtain static, real-time configuration information from network equipment such as routers and this in a transparent way. The same commands and parameters can be used without the need for the user to know the type of underlying device, or the actual device specific command to be executed to retrieve the desired information. Another aspect is that the end-users do not have to contact the device directly, but through the web-service. This again helps in limiting the access to the configuration of the device and protecting the devices against malicious attacks.

A screenshot of the tool is presented in Figure ??, and the features of this front-end are:

- Wizard-style request composition, where a request is constructed step-by-step. First by selecting the devices to be contacted, then by selecting the commands to be executed, i.e. the information to retrieved, and finally by entering any necessary or optional parameters.
- Multiple devices can be specified one request, e.g. to perform a side-by-side comparison.
- Multiple parameters can be inserted.
- Extra information presented to the user about the commands to be executed. This is a textual representation of the command to be executed and also a representation of the necessary and optional parameters to be provided for this command.
- Representation of the result codes. These are codes that represent the user with feed-

back about the status of the execution of the command, similar to the status codes used in HTTP-requests.

- Persistent configuration of the available "SSH/Telnet Measurement Points", so that the addresses of the Measurement Point don't have to be reconfigured every time the client is executed.

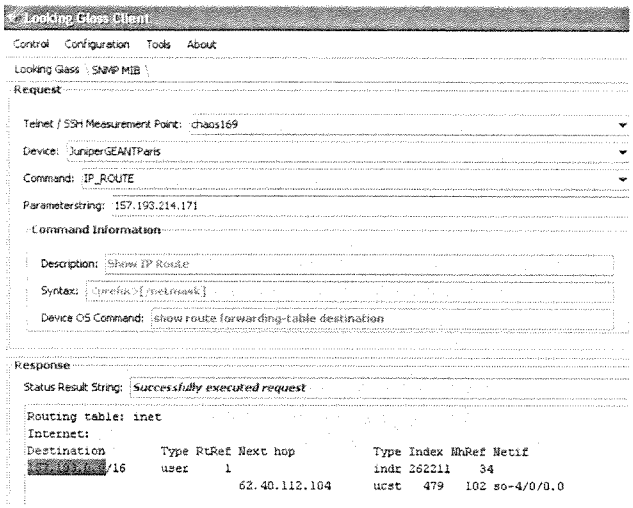

Figure 4: Looking Glass front-end

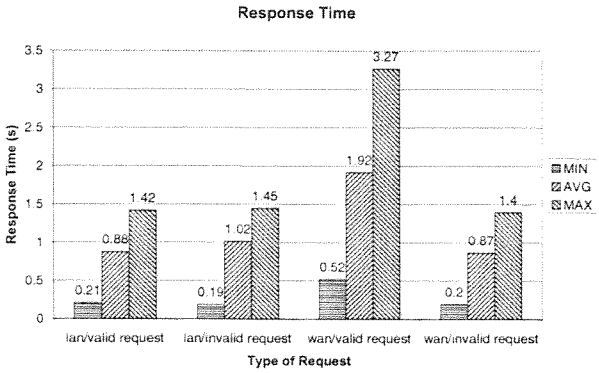

Figure 5: Response time performance metric of the Looking Glass webservice

6 Performance evaluation

Evaluating the response times from the "SSH/Telnet Measurement Point", the results from Figure ?? are obtained. The machine on which the webservice runs is a Linux Debian Sarge machine with a kernel-version 2.6.8-2. The processor is an AMD Athlon(tm) 64 Processor 3200+ processor, with 512MB of memory available. The client machine also has a Linux Debian distribution as operating system, kernel-version 2.6.8-2. An AMD Athlon(tm) 64 Processor 3000+ is installed with 256MB of memory available. Both machines run the Java 1.5.0-10 version. The webservice is deployed inside a Apache Tomcat container version 5.5.20.

The two leftmost collections of response times indicate the time taken to retrieve the results from the underlying router, when that router is located in the same Local Area Network (LAN). The difference between the two has to be found in the type of request transmitted to the router. In the case of a good request, the supplied commands and parameters pass the tests configured at deploy-time. These requests result in an actual connection to the router and retrieval of real-time information. In the other case, some requested information fails the checking procedure and the webservice returns an error return code. Therefore, a connection to the underlying device is not even established. The two rightmost collections represent the response times in the case the router to be contacted and the webservice do not reside in the same LAN. The standard deviation of the response times for the four different types of request on a total of 15 iterations is 0.40 s, 0.19 s, 0.88 s and 0.40 s resp.

7 Future extensions

The functionality the service provides at the current stage can of course be further extended towards a more intelligent way of working. Reference configuration files could be supplied by the deployer of the service. Once up-and-running, the service would then check the actual configuration of routers for the command supplied at regular intervals and match them against the reference configuration. Deviations from this reference configuration would then be notified to the Network Operation Centers (NOCs), responsible for or using these routers.

A second extra functionality to be thoroughly investigated and developed in future work is the adaptation and integration of the "Authentication and Authorization Service" as mentioned in Section ??. Based on the identity attached to the user making the request and the clearance level confirmed by this AA service, other commands and functionality would be provided and allowed to the user.

8 Conclusions

The webservice presented in this paper supports the overall goal of the European Géant2 project in such a way that it provides a standardized data format for retrieving looking glass information across the boundaries of the individual NREN's. Not only does it provide the end-users with the ability to retrieve the necessary information from the devices, it also provides mechanisms to protect the devices from malicious attacks. This is an important feature in situations where the uptime and integrity of the network is of extreme importance.

The architecture is made in a modular and easily extendable way. This provides for an easy reconfiguration towards each and every network's needs. Together with the future extensions, this "Measurement Point" provides the network administrators with a tool that is easily configurable and has a standardized way of communicating. It is part of the Multi-Domain Monitoring Service as specified by the Géant2 project, and thus helps in improving the monitoring of the European Research Network.

Acknowledgment

The authors would like to thank the members of the Géant 2 project and Belnet[?] for the stimulating discussions during the development of the "Looking Glass Webservice" for the Multi-Domain Monitoring Service for the European Research Network. Filip De Turck would also like to thank the Fund for Scientific Research (FWO-V) for the support through his post-doctoral grant.

References

[1] Géant 2: *GÉANT2, the seventh generation of pan-European research and education network*, http://www.geant2.net/.

[2] A. Hanemann, J. Boote, E. Boyd, J. Durand, L. Kudarimoti, R. Lapacz, M. Swany, S. Trocha, and J. Zurawski.: *Perfsonar: A service oriented architecture for multi-domain network monitoring.*, In Service-Oriented Computing - ICSOC 2005, LNCS 3826, Springer Verlag, pages 241254, Amsterdam, The Netherlands, December 2005.

[3] perfSONAR: *PERFormance Service Oriented Network monitoring ARchitecture*, http://www.perfsonar.net/.

[4] Internet2: *Internet2, the foremost U.S. advanced networking consortium.*, http://www.internet2.edu/.

[5] SOAP: *Simple Object Access Protocol (SOAP) 1.1.*, W3C Recommendation, June 2003, http://www.w3.org/TR/soap/.

[6] OGF: *Open Grid Forum: Open Forum, Open Standard*, http://www.ogf.org/.

[7] Global Grid Forum: *A Hierarchy of Network Performance Characteristics for Grid Applications and Services.*, June 2003, http://nmwg.internet2.edu/.

[8] Traceroute: *Overview of traceroute and looking glass tools*, http://www.traceroute.org/.

[9] Rancid: *Really Awesome New Cisco confIg Differ*, http://www.shrubbery.net/rancid/.

[10] SSHTools: *SSHTools is a suite of Java SSH applications providing a Java SSH API, SSH Terminal, SSH secured VNC client, SFTP client and SSH Daemon.*, http://sourceforge.net/projects/sshtools.

[11] JTA: *Telnet+SSH+Terminal, legacy communication tools for Java(tm)*, http://javassh.org/space/start.

[12] BELNET: *The network of knowledge*, http://www.belnet.be/

PROCEEDINGS OF
THE 2007 INTERNATIONAL CONFERENCE ON
PARALLEL AND DISTRIBUTED PROCESSING TECHNIQUES AND
APPLICATIONS

PDPTA 2007

Volume I

Editor

Hamid R. Arabnia

Associate Editors

Sanjay Ahuja, Jesus Carretero
Ping-Tsai Chung, Jose D. Garcia
Kazuki Joe, Mario Nakamori, Chung Ng
Hiroaki Nishikawa, Zornitza Prodanoff
Christian Rehn, Ashu M. G. Solo

WORLDCOMP'07
June 25-28, 2007
Las Vegas Nevada, USA
www.world-academy-of-science.org

ISBN: 1-60132-020-5, 1-60132-021-3 (1-60132-022-1)
Printed in the United States of America