09-02-2010

Deliverable DJ2.3.1: Specification of Advanced Features for a Multi-Domain Monitoring Infrastructure

Deliverable DJ2.3.1

Contractual Date:	30-11-2009
Actual Date:	09-02-2010
Grant Agreement No.:	238875
Activity:	JRA2
Task Item:	Task 3
Nature of Deliverable:	R (Report)
Dissemination Level:	PU (Public)
Lead Partner:	PIONIER
Document Code:	GN3-10-002

Authors: Freek Dijkstra (SARA), Ales Friedl (CESNET), Caglar Gulcehre (ULAKBIM), Jon Kare Hellan (NORDUnet), Nina Jeliazkova (BREN), Gina Kramer (DANTE), Roman Lapacz (PSNC), Richa Malhotra (SURFnet), Krzysztof Nowak (PSNC), Arne Oslebo (NORDUnet), Frederic Raspall (i2CAT), David Rincon (i2CAT), Candido Rodriguez (RedIRIS), Afrodite Sevasti (GRNET), Vladimir Smotlacha (CESNET), Manuel Subredu (RoEduNet), Robert Szuman (PSNC), Sven Ubik (CESNET), Valeriu Vraciu (RoEduNet), Petr Zejdl (CESNET)

Abstract

This deliverable specifies how an enhanced prototype of the perfSONAR system developed in GN2 can be delivered that provides additional monitoring functionality for multi-domain operations and services.

Table of Contents

Table of Figures

Table of Tables

Executive Summary

This document outlines the first results and plans of the research work conducted by JRA2 (Multi-Domain Network Service Research) Task 3 (Monitoring) with the purpose to create an enhanced perfSONAR prototype that delivers additional monitoring functionality for multi-domain operations and services.

The document comprises the following sections:

- Analysis of the Multi-Domain Monitoring System perfSONAR (page 5).

  The perfSONAR system's tools and services are analysed to identify what functionality may be improved, added or expanded.

- Advanced Flow-Monitoring Solutions in the perfSONAR System (page 27).

  An investigation of new traffic flow monitoring techniques and algorithms which may be added to the perfSONAR system.

- Authentication and Authorisation in the perfSONAR System (page 44).

  A description of the improvements to be made to perfSONAR's Authentication and Authorisation Service (AS).

- Advanced Traceroute Functionality in the perfSONAR System (page 47).

  A description of an advanced traceroute that could use perfSONAR monitoring information to identify performance problems caused by misconfiguration, faulty components or exceeded link or node capacity in some specific point in a network path.

- Monitoring the AutoBAHN System with perfSONAR (page 51).

  An overview of the integration of perfSONAR and AutoBAHN which will take place during GN3 in order to provide AutoBAHN with a modular monitoring infrastructure and extend perfSONAR's monitoring capabilities to dynamic circuit monitoring.

- Events Handling in the perfSONAR System (page 55).

  A description of different possible events handling extensions to the perfSONAR architecture that will be analysed in order to identify the best solution. This will allow AS authorised clients to access events from any device they want to monitor in the multi-domain network.

- Vertical Cross-Layer Network Monitoring in a Multi-Domain-Monitoring Infrastructure (page 58).

  As a first step towards cross-layer monitoring, an overview of different standardised information models of a multi-layer network is presented. These will be analysed to identify which one is best suited for storing, managing and publishing measurement data in the perfSONAR system.

- Use of SLA in a Multi-Domain Environment (page 64).

  Provides an overview of the work to be conducted to add Service Level Agreement functionality to perfSONAR, and describes how this functionality will use perfSONAR measurement data.

- New Features and Maintenance of OGF NM-WG and OGF NMC-WG (page 68).

  A description of the actions that the involvement with the ongoing work on OGF Network Measurements Working Group (NM-WG) and OGF Network Measurement and Control Working Group (NMC-WG) standards will entail. The goal of this involvement is to improve perfSONAR's schema and protocol.

The completed work conducted by JRA 2 Task 3 will be passed on to SA2 (Multi-Domain Network Services) Task 3 (Service Monitoring Tools and Performance) for assessment and further development in a production environment.

1 Introduction

As the network monitoring functionality has been developed for many years, its importance has become well known to network operators. Many tools, techniques, solutions and procedures are used to provide insight into network behaviour. However, although a rich set of advanced monitoring instruments exists, continuously evolving user expectations and demands create pressure to introduce new services and thus new technologies and standards. Emerging solutions are deployed in production environments very quickly, forcing changes and improvements in monitoring.

The GN3 project has been created to address new networking trends. Task 3 (Monitoring) of the JRA2 activity (Multi-Domain Network Service Research) focuses on innovative ideas that will bring comprehensive monitoring functionality into the multi-domain environment.

This document outlines the research work's first results and plans for JRA2 Task 3. The included information details what issues and ideas are investigated, why they have been chosen and how they may be tackled. It is not intended to provide all technical details and final agreed solutions. The main purpose is to outline the work of the group at the early stages of the Task and expected results.

1.1 GN2 Inheritance

The GN3 project is the successor of GN2 which provided high quality results and products. One of its main achievements was the multi-domain network monitoring framework perfSONAR (Performance Service-Oriented Network Monitoring Architecture). This is a set of distributed services which offer a range of functionalities that a well-defined and complete multi-domain monitoring system requires (for example, network information and measurement results storage, passive tests, active tests, authentication, authorisation, directory service, visualisation, etc.).

These services can run in multiple domains, communicate using a single, standard protocol, and share information securely. An invaluable advantage of the system is that the network operators do not have to abandon their existing monitoring tools. Instead perfSONAR can be used as an extension that provides new features as well as uniform standardised access to already deployed tools. This may encourage network operators to introduce perfSONAR to their networks.

GN2 took a first step towards a multi-domain network environment with a complete set of services that are accessible in a coherent manner. The concept was devised, tackled and the main elements were implemented. During GN3 perfSONAR will be used for new ideas and solutions, as it provides a suitable platform from which all of JRA2 Task 3's goals can be achieved.

2 Analysis of the Multi-Domain Monitoring System perfSONAR

This section analyses the tools and services the perfSONAR system provides in order to determine areas where functionality may be improved, added or expanded.

2.1 System Completeness and Expansion versus Simplicity

The perfSONAR project designs and builds network performance middleware and visualisation tools. One of the project's main goals is to help solve end-to-end performance problems on paths that cross several domains. perfSONAR provides complex multi-domain monitoring services, satisfying a number of requirements:

- Standardised access to multi-domain network information.
- Increased efficiency through on-demand troubleshooting tools.
- Easy identification of information sources.
- Network performance validation.
- Optimised network usage through regular TCP (Transmission Control Protocol) throughput tests.
- Fast problem identification through consistent end-to-end service monitoring.
- Monitoring of virtual networks.
- Secure access.

2.2 perfSONAR Tools

The perfSONAR MDM (Multi-Domain Monitoring) service currently includes a set of tools that covers the most important area of network monitoring and measurement at the IP layer or above:

- Throughput Measurement Tools

  The perfSONAR MDM Bandwidth Controller (BWCTL) service is intended for throughput measurement. The deployment of BWCTL Measurement Points (MPs) in National Research and Education Networks (NRENs) provides an opportunity to measure throughput between the majority of PoPs in the GEANT-NREN networks. The client can specify data flow parameters for the measurement (for example, window size of TCP relation), so it is possible to check network parameters for different network end point setups.

- Delay Measurement Tools

  The deployment of tools based on delay measurement (e.g. Hades) enables the network operators and systems administrators to obtain parameters derived from a timestamped flow of probing packets. This includes one-way delay [RFC2679], delay jitter [RFC3393], packet loss rate [RFC2680] and also alternative path routes.

- Passive Measurement Tools

  perfSONAR tools based on passive monitoring have recently been developed (e.g. PacketLoss) or are under development (e.g. ABW2). Many network parameters and characteristics can only be obtained using passive measurement which does not influence the network traffic. This includes classification of flows, link load measurement and exact packet loss of real data flows.

2.3 perfSONAR Services

The modular design and open-source nature of the perfSONAR services allows domain administrators to implement and combine tools according to their individual requirements. Domains can thus create customised domain controls with maximum flexibility. perfSONAR integrates all measurement tools by providing common services and visualisation tools. The complex set of perfSONAR services supports access to measurement data and network information across multiple administrative domains. Each individual service is in charge of a specific functionality.

The following types of perfSONAR services are available:

- Measurement Points.

  Measurement Points (MPs) collect and publish data obtained by measurement tools. The measurement is carried out either on client demand or automatically in scheduled time intervals. The MP also presents an interface between the perfSONAR data format and the application proprietary data structure.

- Measurement Archives.

  Measurement Archives (MAs) store measurement results in databases (for example SQL or RRD) or in file systems. This data can later be read by clients and further processed or visualised. MAs can also support data aggregation.

- Lookup Services.

  Each network domain that joins the perfSONAR infrastructure should have a Lookup Service (LS). All services in this domain can then register with the Lookup Service. The Lookup Services of multiple network domains communicate with each other to share information about the registered services. Clients just need to know the URL of one Lookup Service to be able to find out which other services are available in local and external domains.

- Authentication Services.

  Network domains that join the perfSONAR infrastructure can use an Authentication Service (AS). Users are then authenticated before they can access services required to retrieve measurement data or to start measurements.

- Visualisation Tools.

  The perfSONAR MDM visualisation tools are user interfaces for querying perfSONAR services and presenting the collected data to the user in front-end applications (e.g. PerfsonarUI). The open-source nature of the visualisation tools means that the presentation of performance data can be adapted to the needs of specific user groups.

2.4 perfSONAR Completeness

perfSONAR is an extensible open system that integrates a set of measurement tools. The integration of new tools and applications is based on:

- A set of common services deployed across the network.

  Measurement Archives, Lookup Services and the Authentication Service are shared building blocks and a new application has to make sure it is compatible with these services. However, Measurement Points are usually application-specific and should be rewritten or tailored to any new application.

- NM-WG schema utilisation.

  Utilising the Network Measurements Working Group (NM-WG [NMWG]) schema as communication protocol is a characteristic feature of the perfSONAR services.

- The use of a visualisation tool.

  The PerfsonarUI visualisation tool is an integral part of the perfSONAR system. Its structure is open and allows user interfaces to be added or changed.

Besides already existing perfSONAR tools, several other tools can be identified as candidates for integration with perfSONAR. These new tools will increase the number of user groups and improve perfSONAR's completeness. Examples of such new tools are:

- NetFlow based tools.

  Many applications that use NetFlow data collected by active network elements exist. These applications are traditionally preferred by Network Operations Center (NOC) staff and network administrators. Combining NetFlow data from different domains will provide a new, multi-domain overview of network traffic.

- SLS verification.

  Many user applications are network-parameter-sensitive and require a specific quality of end-to-end path (for example, IP telephony or video conferencing). Unfortunately, there is lack of measurement tools that can verify the specified Service Level Specification (SLS). Such tools would be very useful for solving performance problems.

- Advanced passive monitoring applications.

  Passive monitoring applications are indispensable for gaining an insight into real network traffic and measuring parameters like available bandwidth. The issue is the need to utilise expensive, specialised, hardware-accelerated devices for high-speed links. Passive monitoring is also required by security applications that are looking for malicious link flows.

2.5 perfSONAR Issues

The perfSONAR system has been developed for the last 4 years in the GN2 project. In its current status it can be described as a complex system with many MPs spread across some NRENs. The start of the GN3 project provides the right point in time for a critical assessment and the specification of new goals for further developments in the GN3 project. Such an assessment has not been completed yet, however, several ideas that should be considered and further discussed can already be pointed out:

- XML schema semantics.

  While the utilised NM-WG schema defines the syntax of the perfSONAR protocol, fine details of the protocol semantics are defined implicitly inside the XML parser. Other formalisms should also be considered (for example NETCONF [RFC4741]) and their potential benefits for perfSONAR analysed.

- Complicated development of new application-specific Measurement Points.

  Several perfSONAR building blocks (e.g. MA, LS) are universal, but an MP is usually application-specific as it also implements XML schema semantics which tend to differ from application to application. Once perfSONAR developers succeed in defining a universal MP skeleton for interpreting the protocol-common semantics, the design and programming of future MPs will be more effective.

- Client required versus scheduled measurement.

  The measurement process of all perfSONAR tools is either automatically scheduled or started upon client request. In the first case, clients can be repeatedly provided with already stored data, while in the second case, the data is unique and refers to recent measurement only. The issue is that there is not yet any systematic solution to how data can be retrieved that was obtained in the past and/or upon request by another client.

- Tools suitable for network administrators.

  Currently implemented perfSONAR tools are often ignored by NOC staff and network administrators, in most NRENs. Originally, NOC staff were assumed to be typical perfSONAR users. This situation should be analysed, and the possibility to design new tools that are more suitable for day-to-day network management should be considered.

- Alternative visualisation tools.

  The universal PerfsonarUI visualisation tool is a typical front-end application which allows to users to specify query parameters and view perfSONAR service results. It is a complex JAVA application and, therefore, requires adequate host system resources, which limits its usage. The possibility to design and develop perfSONAR web servers and a light-weighted, web-based front-end that can be started from any web browser should be considered. Several perfSONAR tools already provide such visualisation tools, but they are not yet part of the complex perfSONAR solution.

2.6 Analysis of MonALISA

perfSONAR has been developed as an intermediate layer between performance measurement tools that are already deployed within NRENs and visualisation tools which present data to the end user. MonALISA (Monitoring Agents in A Large Integrated Services Architecture), is a scalable framework of services for monitoring, managing and optimising the operational performance of networks in real time [ML].

To develop new ideas how perfSONAR could be improved and how the experience of NRENs who use it could be enhanced, it is useful to analyse how MonALISA operates, and if it offers functionality that perfSONAR could emulate. The key aspects that need study are:

- Application design.
- Storage and data handling.
- Communication infrastructure.
- Security.
- Data utilisation and presentation.
- Available services and functionalities.

2.6.1 Application Design

Comparing perfSONAR with similar applications at design level reveals how other applications have been designed, including what technologies and software components have been used and how they were combined into a final product.

The key technology used in MonALISA is JINI [JINI]. JINI technology is a service-oriented architecture that defines a programming model which both exploits and extends Java technology to enable the construction of secure, distributed systems consisting of federations of well-behaved network services and clients. JINI technology can be used to build adaptive network systems that are scalable, evolvable and flexible as typically required in dynamic computing environments.

The MonALISA architecture (see Figure 2.1) has services structured on 4 layers. At the first layer the Lookup Discovery Service (LUS) is located. LUS provides dynamic registration and discovery for all other services. It is similar to the perfSONAR Lookup Service (LS) and provides the same functionalities. The main advantage of LUS over LS is that it is implemented using JINI which brings all the advantages of the distributed systems to LUS.

The second layer of MonALISA consists of services and agents. A service in the MonALISA framework can interact autonomously with other services using one of the following methods:

- Through dynamic proxies (located at the third layer)
- By using agents that use self-describing protocols.

Services and agents are grouped and used together to form a distributed system for gathering and analysing information. Agent groups are known and maintained by the proxies. The groups are dynamically created at runtime.

Proxies, located at the third level, are the only components that can be used for the communication between components from the forth and the second level, after successful registration with a LUS. Since proxies are the only communication point between clients and agents/services, they are also responsible for intelligent multiplexing of the communication, the Authentication, Authorisation and Accounting (AAA) process and Access Control List (ACL) enforcement.

The forth level comprises all the applications that use the data provided by MonALISA to provide global services to the end users.

MonALISA uses the code mobility paradigm (mobile agents), which extends the approaches of remote procedure calls (RPC) or client-servers. The code and the required parameters are downloaded dynamically. This has several advantages:

- Optimised asynchronous communication.
- Disconnected operation.
- Adaptability.
- Dynamic parallel execution.
- Excellent mobility.

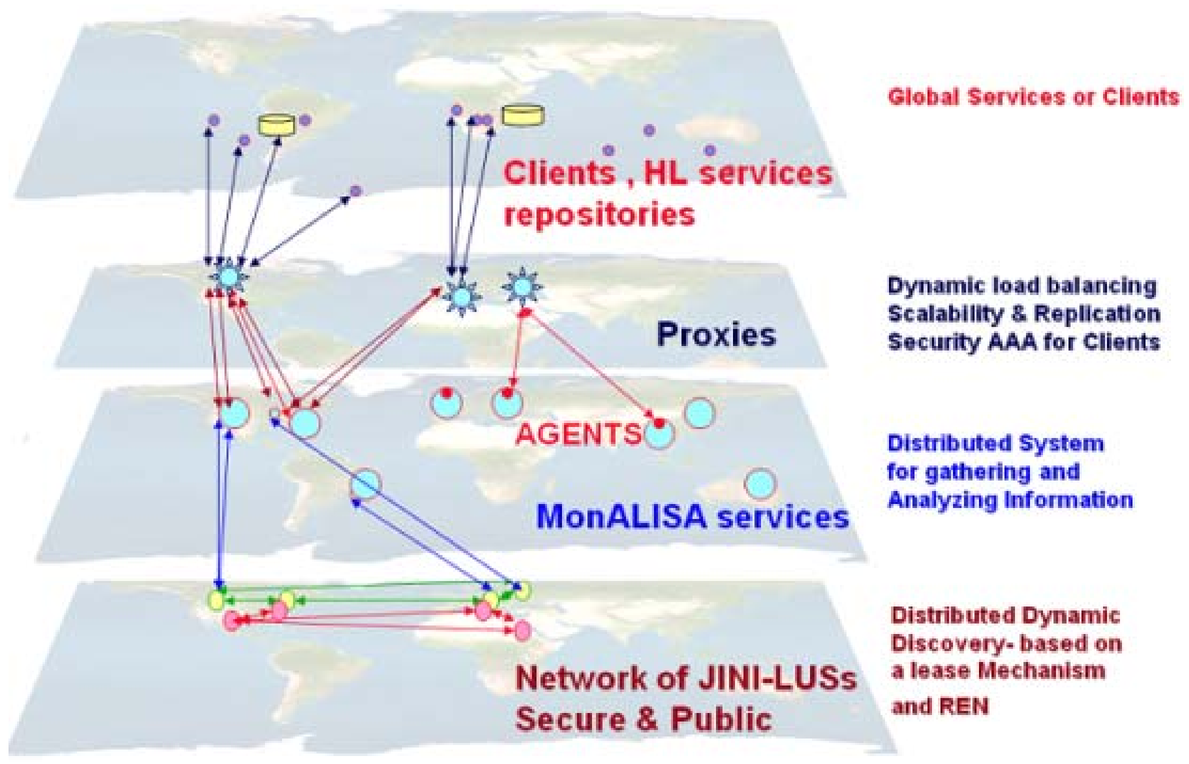


Figure 2.1: MonALISA architecture [MLarch].

2.6.2 Storage and Data Handling

MonALISA services can create local databases which can be used for short of long term data storage. Data is stored using relational databases. The default choice is PostgreSQL [PostgreSQL], but other relational database engines can be used as long as they are supported by Java Database Connectivity (JDBC).

A multi-threaded engine is used to gather parallel and independent data. Fail monitoring (due to blocking I/O or faulty equipments) is automatically removed from the execution queue so it does not interfere with the rest of the system.

A system based on predicates can be used to retrieve data from the local database or to subscribe to new data events. The SQL queries used to retrieve data from the database are built using the predicates specified by the client application. The administrator can choose the timespan for which the data is stored and the data representation from a web interface. The repositories use a dual time resolution system (1 and 100 minutes) for the entire time interval. This structure allows the storage engine to select the best data source in terms of resolution and database interrogation speed. If for example the user requests data from the last hour, the 1 minute resolution data is used but if the user requests data for the last month, the 100 minutes resolution data is used. The efficiency is pushed even further with an adaptive memory buffer which tries to keep as much data in memory as possible, looking into Java Virtual Machine (JVM) to determine the free memory and changing accordingly.

The data used within MonALISA is user-definable, so that any service user can implement their own data type. The data has to be self-describing so that the database engine can store it in a transparent manner. This allows MonALISA to transport and store different types of monitoring data.

Another useful feature of MonALISA is that collected information can either be stored in the local databases or in a central repository (a web repository). The web repositories can store historical and real-time values, statistics and graphical charts.

2.6.3 Communication Infrastructure

Because MonALISA is a distributed framework with agents and services running in very diverse environments, communication between components is essential. The most important parts of the communication system are registration, discovery of services, clients-services interaction and agent cooperation.

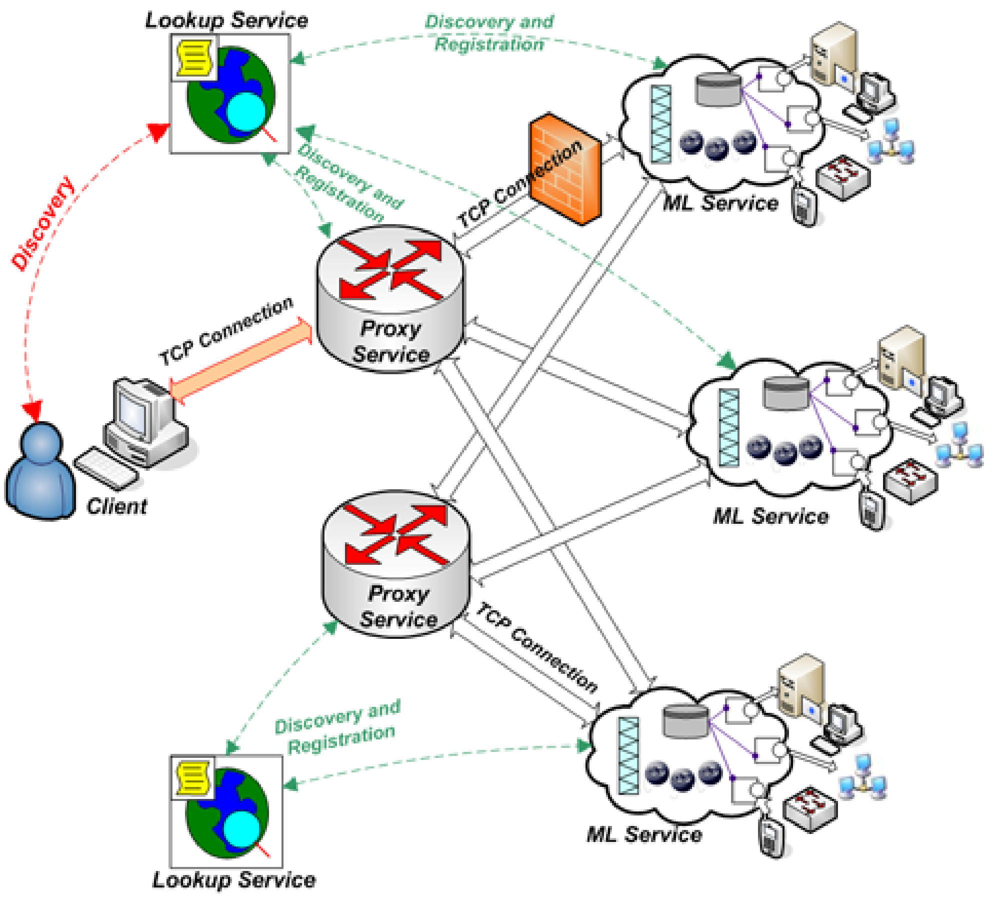


Figure 2.2: MonALISA Communication Infrastructure [MLComInfra].

Where secure communication is needed, the communication between MonALISA components is made using TCP connections combined with Secure Sockets Layer (SSL). To gather and use data from multiple agents written in multiple programming languages, MonALISA uses web services to exchange data between components. The data is used according to the Web Services Description Language (WSDL) for data exchanged between components and SOAP. These two technologies allow non-native (non java) applications to use MonALISA's services.

Easy information exchange is provided through agents which can communicate with each other or services through MonALISA proxies, in a similar way to mail relay. An agent sends a message to the proxy, the proxy verifies that the source is genuine and then tries to deliver the message to the destination. If for some reason the destination is unavailable, the message is stored locally for a period of time and transmissions is attempted again. If the message still cannot be delivered, an error message is sent back to the source.

The agents communicate with each other using three types of messages:

- Unicast.

  Like in networking, the unicast message is a normal message, sent from a single source to a single destination

- Broadcast.

  The broadcast message is sent from a single source to all the agents within the group.

- Information.

  The information message is sent by an agent to the proxy to request information about other agents. An agent can ask a proxy for information about agents within the same group or for information about all agents known to the proxy.

2.6.4 Security

MonALISA's security is largely based on the Globus Security Infrastructure (GSI) but also uses some specific components. Like GSI, the MonALISA security infrastructure is based on existing standards (TLS, PKI and GSS-API) and has 3 main components:

- Authentication.
- Identity certificates.
- Proxy certificates.

Users are authenticated using X.509v3 public key certificates issued by Certificate Authorities. The distinguished name (DN), which is considered a unique identifier, contains information about the user (name, organisational affiliation and email address). The authorisation is based on the DN of the subject contained in the certificate, and this information is used as a unique identifier when defining an ACL.

The advantage of using X.509v3 certificates instead of remote authentication (used by services like Eduroam, EduGAIN, etc) is that it does not have a central point that can fail. Therefore the service can work even in a disconnected environment. The process of a typical session is as follows:

1. Users authenticate to the proxy using their own certificate to prove their identity.
2. Based on the DN extracted from the certificate, the proxy service downloads the access control lists from the AAA service locally. The connection between proxy and AAA service is secured and the proxy itself is authenticated by the AAA service using the X.509 certificate.
3. All communications between the client and MonALISA services are made trough this connection and filtered by the downloaded ACLs.

2.6.5 Data Utilisation and Presentation

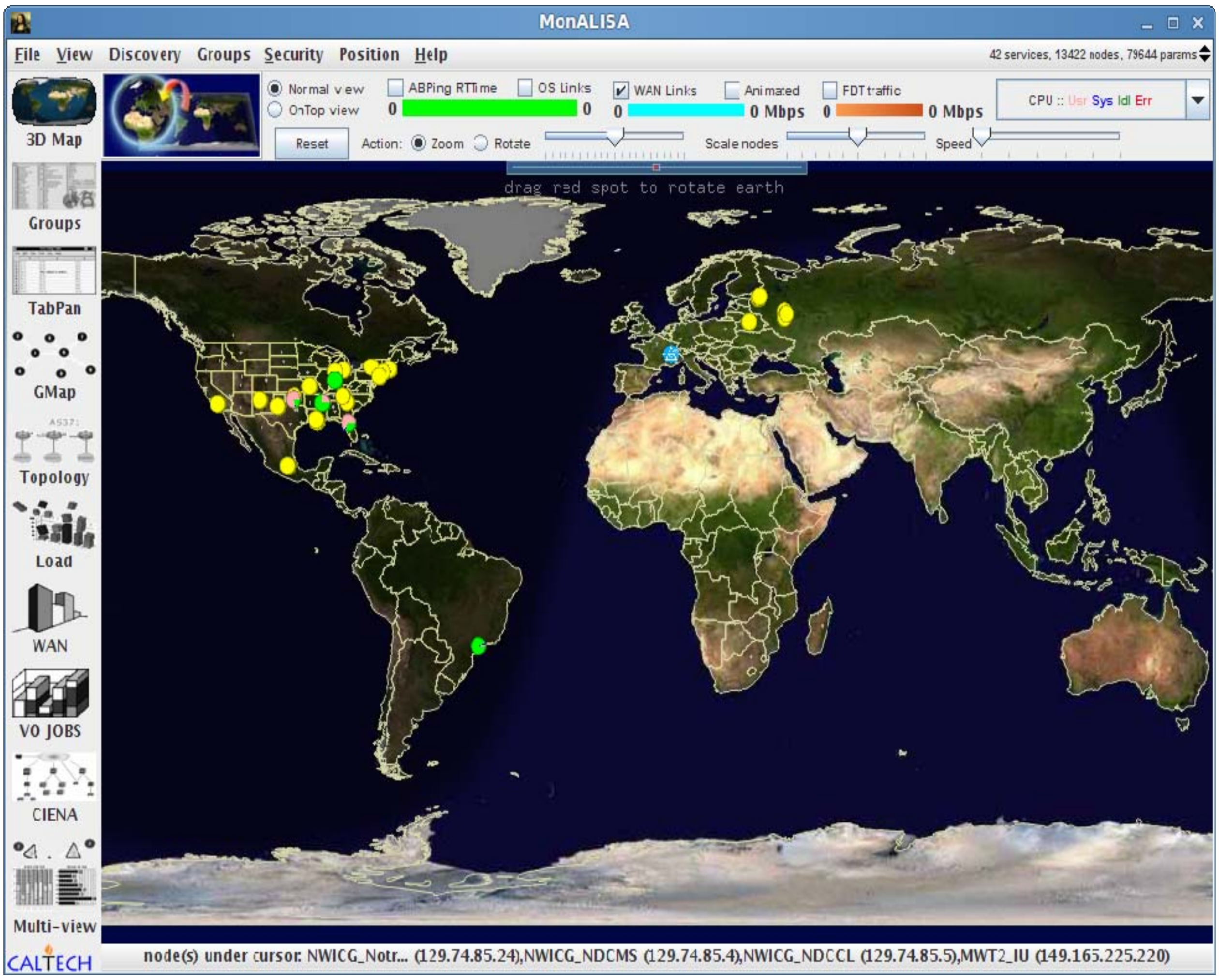
The data available in MonALISA can be accessed by the end user through standard clients. The official applications for accessing MonALISA data are:

- The MonALISA client.
- The EVO client.
- LISA.

All three applications are build in Java and available to the public using Java Web Start technology.

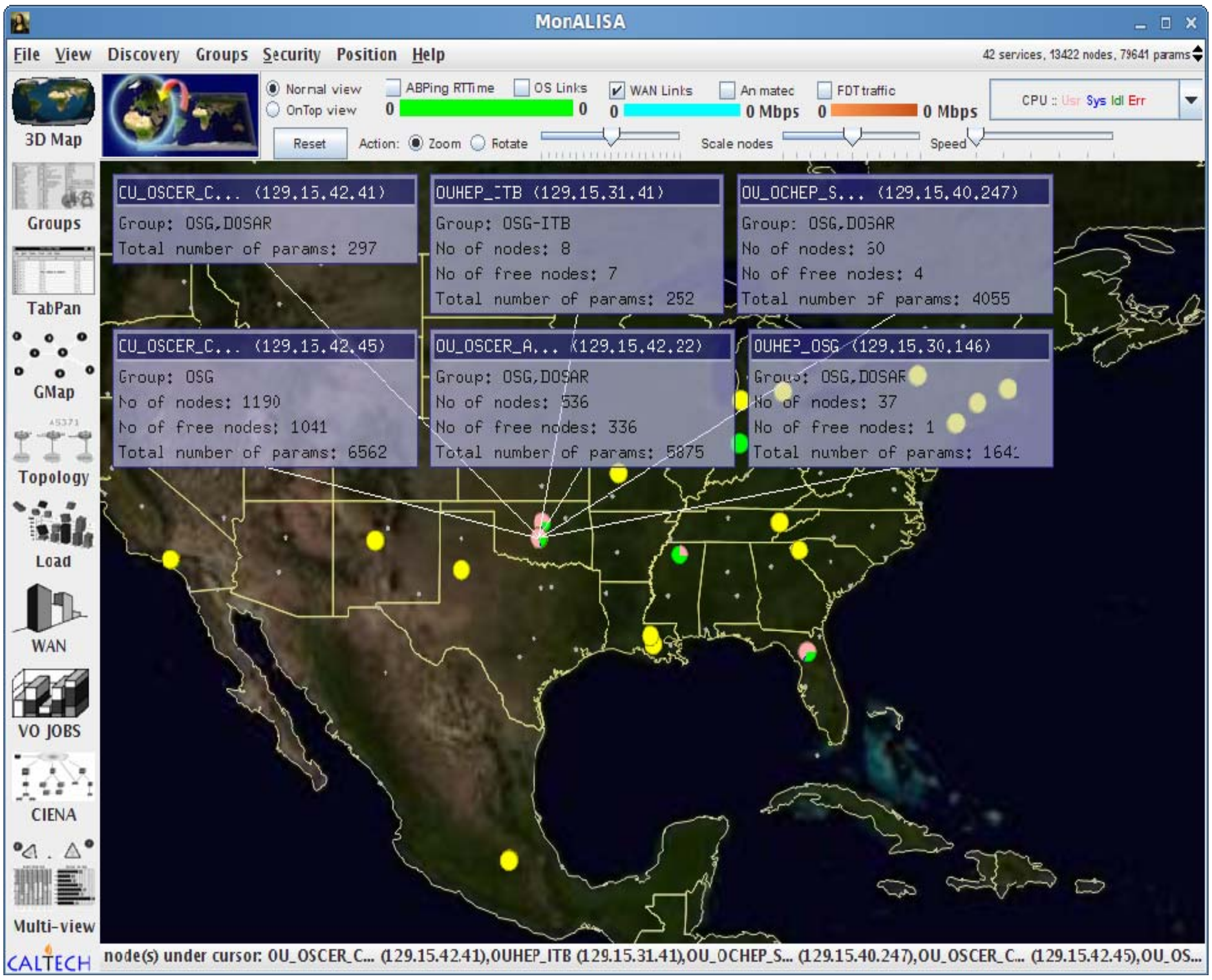
2.6.5.1 *The MonALISA Client*

The MonALISA client is a global monitoring service client. Information (active services, global views of connectivity and traffic, farm usage, farm load, site configuration, parameter values) is shown in real time and all physical locations of the services are shown on a 3D earth map. Because the 3D rendering requires a lot of resource (at least 1GB of memory), two other, lighter versions are available (EVO and LISA).



node(s) under cursor: NWICG_Notr... (129.74.85.24),NWICG_NDCMS (129.74.85.4),NWICG_NDCCL (129.74.85.5),MWT2_IU (149.165.225.220)

Figure 2.3: MonALISA client initial screen.



CU_OSCER_C... (129.15.42.41)
Group: OSG,DOSAR
Total number of params: 297

OUHEP_ITB (129.15.31.41)
Group: OSG-ITB
No of nodes: 8
No of free nodes: 7
Total number of params: 252

OU_OCHEP_S... (129.15.40.247)
Group: OSG,DOSAR
No of nodes: 50
No of free nodes: 4
Total number of params: 4055

CU_OSCER_C... (129.15.42.45)
Group: OSG
No of nodes: 1190
No of free nodes: 1041
Total number of params: 6562

OU_OSCER_A... (129.15.42.22)
Group: OSG,DOSAR
No of nodes: 536
No of free nodes: 336
Total number of params: 5875

OUHEP_OSG (129.15.30.146)
Group: OSG,DOSAR
No of nodes: 37
No of free nodes: 1
Total number of params: 1641

node(s) under cursor: OU_OSCER_C... (129.15.42.41),OUHEP_ITB (129.15.31.41),OU_OCHEP_S... (129.15.40.247),OU_OSCER_C... (129.15.42.45),OU_OS...

Figure 2.4: MonALISA client - detailed node view.

2.6.5.2 *The EVO Client*

The EVO client provides a good example of how MonALISA can be customised. By using data available in the MonALISA system, the EVO client is able to show traffic values, the number of video conferences, users, etc. in real time.

2.6.5.3 *LISA*

LISA (Localhost Information Service Agent), is an application that can be run on any system (provided Java is installed) to display system information (e.g processor load, number of processes, network traffic). Using LISA allows other services provided by MonALISA to be used (e.g. IPerf).

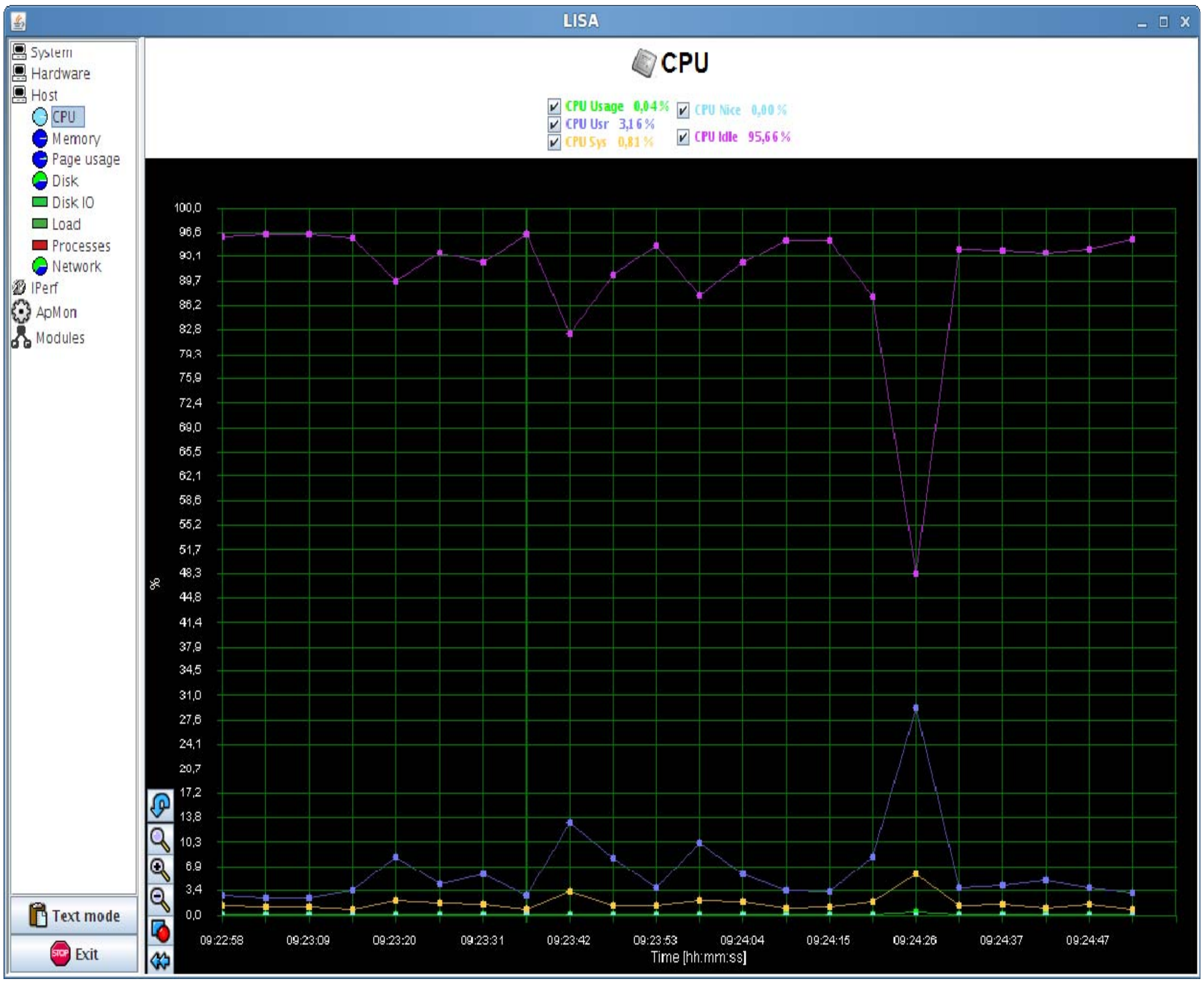


Figure 2.5: LISA CPU graph.

2.6.6 Available Services and Functionalities

MonALISA services are provided by modules. Currently, there are modules for:

- General networking metrics.
- Intrusion detection (agents analyse information generated by SNORT [SNORT]).
- Wide Area Network (WAN) topology.
- Available bandwidth (using path load).
- Dynamic path allocation (dynamically create circuits between two points).

The services implemented in perfSONAR are directly related to network performance monitoring. perfSONAR has been designed with network performance monitoring and GEANT-integration in mind. MonALISA has been designed as a general monitoring framework to be built on and not specific for network monitoring. Therefore, an exact comparison with the services provided by perfSONAR is very difficult.

Both applications provide support for traffic metrics like Round Trip Time (RTT), jitter, link utilisation, packet counters (in, out, error, drops). Both applications support storing data in databases. perfSONAR is easier to deploy in an existing environment and easier to configure to use existing network data (data within RRD files produces by other applications). MonALISA, as a general monitoring framework, cannot easily be configured to use data from other applications. However, using MonALISA, it is easier to build a custom network monitoring solution that can monitor both network and network-dependent applications, or a solution that can execute actions (like circuit reconfiguration or traffic rerouting).

2.7 Use of Protocol Standards OGF NM-WG and OGF NMC-WG versus NETCONF

While perfSONAR provides a good infrastructure for network performance monitoring, its current implementations and protocols have some problems. In this section some of these problems are identified and discussed, and a possible solution based on the IETF protocol NETCONF is proposed.

2.7.1 Problems with Current Implementations and the NM-WG protocol

Most of the problems with the current perfSONAR are related to the NM-WG protocol. There are three main problems that should be addressed:

- Lack of proper separation between information model and communication model.
- Lack of generic information model for MA and MP.
- Lack of proper message validation mechanisms.

2.7.1.1 *Separate Information and Communication Model*

Separating what to transfer (i.e. the information model) from how it is transferred (i.e. the communication model) is a common technique used in many popular network management protocols. As part of the specification of the OSI protocol CMIP, OSI defined what is referred to as the OSI Network Management Model [OSI]. This model divides the specification of network monitoring architectures into several high level models:

- Organisation – defines components and relationships.
- Information – the syntax and semantics of the information shared between components.
- Communication – transfer syntax for sharing the information between components.
- Functional – defines application functions like configuration monitoring, performance measurements etc.

The Simple Network Management Protocol (SNMP), the most commonly used protocol for network management, never explicitly defined the management architecture like OSI, but uses the same structure implicitly [RFC1157] [RFC341x]. The SNMP standard describes the components and relationships between them, and strictly distinguishes between the information and communication model.

The information model in SNMP uses Structure of Managed Information (SMI) [RFC2578] to specify Management Information Bases (MIBs), which can then be transmitted between the manager and agent using the SNMP protocol.

There are many other protocols that also follow the OSI Network Management Model. Some examples are NETCONF, WS-management [DTMF] and WSDM [WSDM]. Many protocols follow this model since there are several advantages of designing a protocol like this, especially when it comes to the separating the information and communication model.

Separating these two models allows both of them to evolve and change relatively independently of each other. One example of this is the SNMP protocol which has two different versions of the information modelling language SMI and three different versions of the protocol.

Another advantage of using a protocol that separates the information and communication models, is that this separation is usually also adopted by applications using the protocol. Again SNMP can be used as an example. Most network management applications that use SNMP only focus on which information they want to retrieve from an agent and do not care how the information is retrieved. An SNMP library does the actual encoding and decoding of messages. Simple utilities allow network operators to create small scripts for managing their network with relatively little effort. Developing small scripts like this is a very common practice among network operators [NM] [SNM] [FNM].

The OGF NM-WG protocol used by perfSONAR does not follow the OSI Management Model and does not properly separate the information and communication model. For the MAs and MPs no proper information model exists. NM-WG defines a base XML schema that can be used for transferring data and meta data between the different services in perfSONAR. For each service like RRD MA, HADES MA, SNMP MP etc. new XML schemas are defined that extends the base schema. This means that in NM-WG the specification of what to transfer is mixed with how it is transferred. This makes the protocol inflexible and current applications are closely coupled with the protocol (for an example, see *Separating perfSONAR Information and Communication Models with NM-WG* on page 71).

The lack of separation between information and communication model is also evident in current perfSONAR implementations. The perfSONAR applications require all the details about how data is transferred between the services and it takes a lot of effort and code to create and decode all the messages. This makes it very difficult to create simple utilities, and it makes it harder for network operators to create their own small scripts.

2.7.1.2 *Generic Information Model*

Separating the information model from the communication model is not enough. perfSONAR should also define a completely generic information model for MA and MP that contains a self-documenting template system which allows user interfaces to display all types of metrics without any code changes. When querying an MA or MP, the template system should specify if the metrics retrieved are bytes, bits per second, IP address, etc. This would allow the user interface to know how to display the metrics appropriately.

In the current perfSONAR different types of MA and MP (like RRD MA, SQL MA, SNMP MP, SSH MP) have different XML schemas. Clients using the services must implement support for each MA and MP that they support. This imposes a lot of restrictions on how the perfSONAR infrastructure can be used.

An example of an application that could benefit from a generic information model for MA and MP is the Advanced Traceroute application (see *Advanced Traceroute Functionality in the perfSONAR System* on page 47). This application does a traceroute to an IP address and checks with an LS for each hop to see if an SNMP MP exists that can provide some performance metrics for the link. Since there is no generic information model for MPs and MAs, the application can only use the SNMP MP, and only a hard coded set of SNMP objects can be retrieved. If another type of MP is available (like a passive monitoring probe) more detailed information about the link can be provided. However, the Advanced Traceroute application is not able to take advantage of this without code being manually added. If a generic self-documenting information model for MPs and MAs existed, the application would be able to display all available statistics for a given link, regardless of the source.

2.7.1.3 *Protocol Inconsistency*

Another problem is protocol inconsistency in the perfSONAR implementations. For example, perfSONAR defines an Echo message that can be used to check if a perfSONAR service (for example, an MA) is operational. If this message is sent to different perfSONAR services, different reply messages may be returned. There is no complete specification of messages which could be used by service developers. Unfortunately, OGF NM-WG is not a final standard supported by a standardisation organisation

Also, in current perfSONAR implementations, all messages are created and parsed manually without any formal XML validation. OGF NM-WG is specified using only Relax NG as the formal specification language. If the Relax NG schemas had been used by all applications for formal message validation, some of the inconsistency would have been avoided. Relax NG does, however, not have the necessary formalism to define all aspects of a proper information model [MLAng] [YANG]. It is therefore not possible to properly validate the XML messages in perfSONAR which can lead to inconsistencies between different implementations.

2.7.2 Possible Solutions

In looking for possible solutions to the problems with current perfSONAR implementations, a step back was taken to look at what perfSONAR is. perfSONAR is an implementation for handling distributed management. Distributed management is something that researchers and standardisation bodies have worked on for many years and many standardised protocols (like WS-Management, WSDM and NETCONF) are available. Instead of trying to fix OGF NM-WG and compete with existing standardised protocols, NETCONF was used as a protocol for transferring data between perfSONAR services. To create an MA information model, the draft specification of the YANG modelling language [YANG1] was used.

To simplify the implementation the Stager [Stager] application was used as a base for the prototype. Stager is a web-based application for presenting and aggregating most types of network statistics. It has a built-in template system that allows the web front-end to display most types of metrics by providing an XML-based template that describes the data.

2.7.2.1 *NETCONF and Yang*

The NETCONF protocol was originally designed for configuring network equipment but it is also well suited for retrieving and storing data in an MA in the perfSONAR framework. NETCONF has several predefined protocol operations but also allows applications to define new operations. Therefore, it is possible for an MA to define new protocol operations for retrieving information about available observation points, time periods etc.

However, this is not a ggod choice, since it means that both the MA and the client retrieving information from the MA need to support these new operations.

In the prototype implementation only standardised features of the NETCONF protocol are used, so that any off-the-shelf NETCONF tool can be used to retrieve information from the MA. The Get protocol operation is used to retrieve information, and the optional XPath support in NETCONF is used to specify from which nodes it should be retrieved.

A client needs to know how the information in an MA is organised. This is specified using the YANG modelling language currently under standardisation in the IETF. With YANG, a proper information model for an MA was defined that included a generic report template that can be used by clients to retrieve information about the measurement data in the MA. This means that a user interface can display all kinds of reports without any code modifications. When retrieving data from the MA, the template system specifies if the data retrieved is a counter for octets, represents a temperature, IP address etc. and the user interface can format the data accordingly.

For a detailed description of the MA information model, see *MA Information Model using NETCONF* on page 72.

2.7.2.2 *Querying the Measurement Archive*

The idea behind using NETCONF and a standardised information model defined in YANG is that all queries to the MA can be done using simple XPath expressions. For example, to retrieve a list of all available data sources in an MA, all that is needed is the following XPath query:

```
/measurementArchive/datasources/source/name
```

This returns a list of names:

```
<name>qstream</name>
<name>wdm</name>
<name>mping</name>
<name>ssmping</name>
<name>qflow</name>
<name>appmon</name>
<name>NetFlow</name>
<name>ssmping (Netconf)</name>
<name>RRD (Netconf)</name>
<name>NetFlow Raw</name>
```

When retrieving data for an actual report, an MA should never return any data for a query that wants to download the entire report subtree. It is important to realise that the XML document described by the YANG model is not a real XML document but a virtual document existing in the MA, which can be very large.

Instead explicit XPath expressions that limit the returned data should be used. A typical XPath query for retrieving data for a report can be:

```
/measurementArchive/datasources/source[name="ssmping"]/timeperiods/
timeperiod[starttime="20090901 12:00" and duration="1 hour"]/
reports/report[id="asmping4' and obspoint='1' and
                transformation='table' and view='Standard'
                and sort=1 and limit=20]/data
```

In this query data from the source "ssmping" is collected for the period that starts at 12:00 on 1st of September 2009 (20090901) and lasts for one hour. The query also states that the report for which data is retrieved is called "asmping4", the observation point is 1, transformation is "table", the view is "Standard", sorting is done by column 1 and the maximum number of rows to retrieve is 20.

The query returns a list of data rows and the user interface presents it using the report template, which it can retrieve through another query.

For a more detailed description of the prototype implementation and additional examples on how to query the MA, see *Implementation of NETCONF in perfSONAR* on page 77.

2.7.3 Conclusions

Several weaknesses in the current perfSONAR implementations and the OGF NM-WG protocol have been identified. Solving these weaknesses will help with the deployment of perfSONAR among network operators.

While it is possible to fix the OGF NM-WG protocol, already standardised protocols are available and the prototype implementation of an MA shows that the NETCONF protocol with the YANG information modelling language is well suited for use in perfSONAR.

NETCONF solves all the addressed weaknesses. It has a strict separation of the information and communication model. The YANG modelling language is well suited to create a generic information model for an MA and as part of the work of standardising YANG, an Internet draft is available that specifies how to validate all NETCONF messages.

The next step in this work is to create proper information models for each perfSONAR service. For MAs and MPs, the Stager template system is a good starting point but should be simplified. A lot of work has gone into NM-WG and NMC-WG and the experiences gained from this work could go into creating formal YANG information models for the various services.

2.8 Lookup Service Testing Functionality

The Lookup Service (LS) is a key element of the measurement framework because it allows every independent service to be a visible part of the system. New services can identify themselves to the community and provide a detailed description of their capabilities. Other services can communicate with the LS to retrieve this Lookup Information.

Services register with the Lookup Service by sending their Lookup information via an LSRegisterRequest. The Lookup information contains data such as access point URL, service name, service type, service description and any other relevant data (for example, a list of stored interfaces, if the service is a Measuring Archive). Services can also de-register (remove) their Lookup information.

The Lookup Service can remove obsolete Lookup information using the LSCleanup module. All Lookup information is valid for a particular period of time (set globally by the LS administrator). Modifications of Lookup information increase the information’s validity. Instead of re-registering the same data over and over again to prevent the Lookup information from becoming obsolete, a service can send an LSKeepaliveRequest which contains just the database key.

Services that are already registered can re-register updated data by sending an LSRegisterRequest which contains the key to already registered data.

The service can be tested from a client application using the EchoRequest. perfSONAR base supports a basic EchoRequest, but the Lookup Service provides more detailed tests mostly for connectivity to the XML database.

Individual LS instances are connected in a federated global system (see Figure 2.6), which consists of well known, global Lookup Service (gLS) instances that form the top level of the hierarchy. Local LS instances (home LS (hLS)) are on the next hierarchy level. They manage the registration of individual services and communicate a summary of information to the upper level. Top-level global Lookup Services manage only the registration of hLS instances but no other perfSONAR service types (MAs and MPs). perfSONAR services can register with one or more gLS to increase reliability and load balancing.

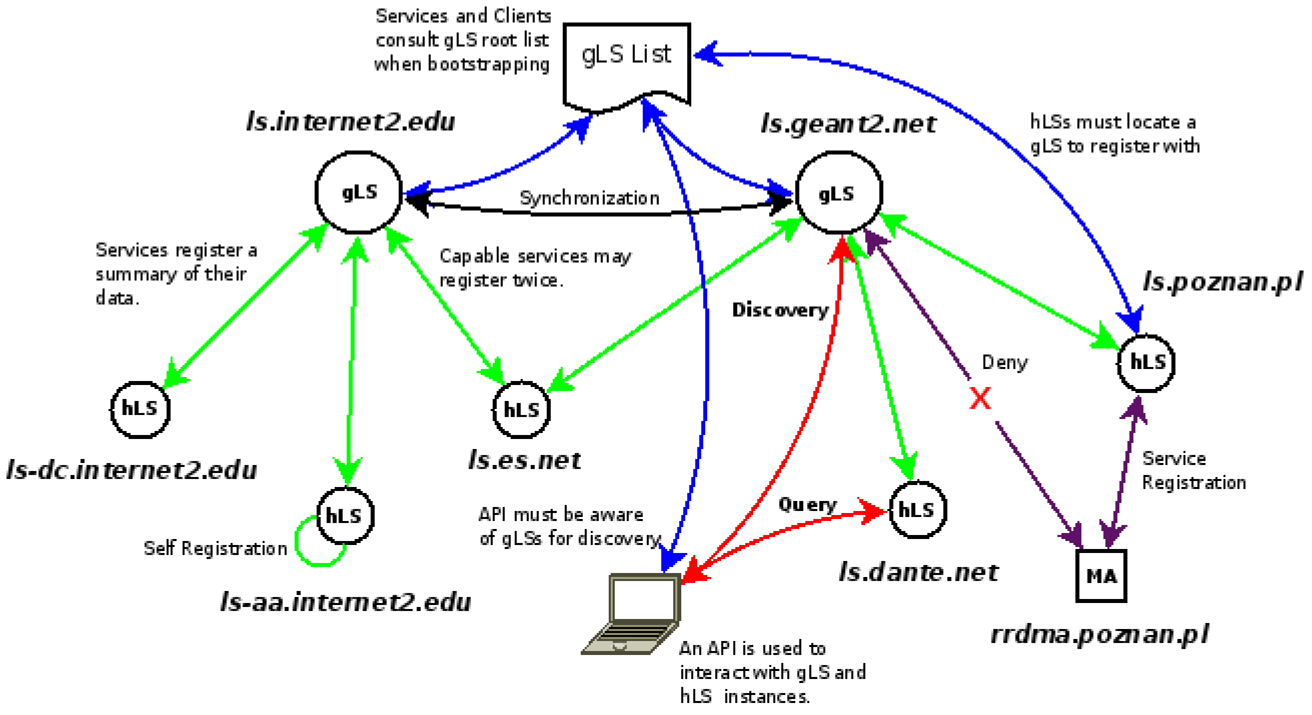

Services and Clients consult gLS root list when bootstrapping
gLS List
ls.internet2.edu
ls.geant2.net
hLSs must locate a gLS to register with
gLS
Synchronization
gLS
Services register a summary of their data.
Capable services may register twice.
Discovery
ls.poznan.pl
Deny
hLS
hLS
hLS
ls-dc.internet2.edu
ls.es.net
Service Registration
hLS
API must be aware of gLSs for discovery
Query
hLS
Self Registration
ls.dante.net
MA
ls-aa.internet2.edu
An API is used to interact with gLS and hLS instances.
rrdma.poznan.pl

Figure 2.6: Architecture of a distributed Lookup Service.

perfSONAR project partners (GÉANT, ESNET, Internet2, RNP) maintain root gLS services, which serve as well-known points, similar to DNS (Domain Name System) root servers. Each root service manages a given set of hLS data. Bootstrap and synchronisation procedures ensure that the information maintained by root servers is consistent. Services and client applications submit unified queries to gLS and hLS to retrieve information that perfSONAR services registered with hLS. There are two types of unified queries:

- Discovery

  Finds the specific LS that has relevant information.

- Query

  Requests, for example, available information for specific metrics and elements of the network topology. The query is sent directly to the LS in question.

Global Lookup Services summarise the data [gLS] and store generalised information of topological endpoints, rather than individual entries, received by hLS.

2.8.1 Tools for Lookup Service Testing

To ensure flawless operation of the complex distributed system comprising gLS, hLS and perfSONAR services, tools to test its most important functionalities are needed. The multi-step nature of querying perfSONAR services (Discovery query, LS query, MA or MP query) complicates testing by requiring query sequences to retrieve information for every single metric and interface. Since PerfsonarUI has already been used to query and test different perfSONAR services, an additional new module for testing the LS was suggested in Gn3. The existing PerfsonarUI Playground module was occasionally used for testing LS services, but required too much knowledge of the specific gLS and hLS queries, as well as the LS infrastructure. The additional Lookup Service Playground module that was proposed for testing the LS therefore includes the following functionalities:

- Simple testing functionality of home LS and Discovery protocol (involves global Lookup Services, listed in a predefined top root.hints file):
  - Ability to change LS federation, by specifying selected root.hints or just selecting specific global LS.
  - Send/receive echo requests/responses.
  - Given a user-selected network element, event types and service type, retrieve and display the following information:
    - global Lookup service(s) involved.
    - home LS(s) involved.
    - MA or MP that serve the specified network element and event types
    - Verify if MA/MP does indeed provide information about the specified network element and event types (troubleshooting summarisation errors).
  - The user should be able to inspect the LS hierarchy relevant to the query network element and identify/report if a problem exists.
  - Report timings of LS interactions.

- More complex scenario (to be considered in further development):

Providing different views of the LS infrastructure.

  - Top down view:

    An LS hierarchy (as in [i2]) with the ability to inspect which network elements are served by the hLS.
  - Bottom up view:

    Starting with (multiple) user-selected network elements and event types, retrieve and display a map of the involved MA/MP, home and global LS troubleshooting.
  - Observing multiple LS services simultaneously to keep an eye on summarisation and LS data exchange in a multi-domain environment (speed, amount of data being exchanged, etc.).

2.8.2 Implementation

An initial implementation of a simple testing scenario in a Lookup Service Playground module for PerfsonarUI is available at [SIMPLETEST].

The interface is intentionally simple (currently no support exists for event and service types). Currently, it is only possible to query for network elements and retrieve the services that are responsible for them (if available).

2.8.3 Testing

The new LS Playground module was tested and discussed via mailing lists by several project partners. Functionality and user interface improvements have been suggested. Several infrastructure issues have been identified:

- The GN3 top-level gLS was not available at the beginning of the tests.
- Incompatibilities between hLS registration messages of MDM hLS instances.
- Issues pertaining to the correct timing of different messages that influence the correctness and completeness of the information, provided by the distributed LS infrastructure.
- Issues, pertaining to the summarisation procedure, resulting in false positives in Discovery/Query results (e.g. a query for a specific IP address returns a pointer to an MA service, which does not provide any information about an endpoint with this IP address).

2.8.4 Conclusions

An LS Playground module with a simple user interface has been developed. The need for this module is illustrated by the fact that several LS infrastructure issues were discovered during the testing procedure. Several improvements to the user interface and functionality that were suggested by project partners are planned to be implemented.

3 Advanced Flow-Monitoring Solutions in the perfSONAR System

This section provides an overview of the new techniques and algorithms for the measurement and monitoring of traffic at the flow level that will be investigated in GN3. This investigation aims to extend the current perfSONAR system with new tools or plugins in order to improve the monitoring, provisioning and operations within the project.

3.1 Traffic Matrix Modelling

With a view to potentially extending perfSONAR with traffic matrix (TM) functionality, research is conducted to provide good models for TMs, with potential applications for traffic demands prediction, anomaly detection, and traffic engineering. Traffic matrices offer a way to express the volume of data exchanged between ingress and egress nodes in a network [ACRUW06]. Although TMs can be expressed at different levels of temporal (from 5 minutes to hours, days, months or years) and spatial aggregation (origin-destination machine, OD prefix, OD router, OD PoP or OD AS), the investigation will focus on cases of interest to GÉANT, and the nodes will be either routers or the PoPs present at each NREN. While there is some literature on the analysis of TM, few efforts have been made in the field of multi-domain TMs.

In the past, a lot of effort has been devoted to the problem of measuring TMs. Although there are tools that can provide TMs directly (e.g. NetFlow), they do not scale well (see *Enhanced Flow-Level Measurements* on page 37), and measurement campaigns are difficult to set up (see [UQLB06] for a detailed description of one of such measurement campaigns, performed in GN2). On the other hand, SNMP data (describing the volume of traffic sent or received in each link) is easy to collect, and almost ubiquitous, as most networking equipment supports the MIB-II. However, SNMP data only provides link load measurements, not TM measurements, i.e., SNMP is able to compute the aggregate traffic transported by the link, but unable to distinguish the specific origin-destination pairs of each flow. The resulting problem of inferring the TM from link measurements is a classic under constrained, linear-inverse problem which needs some sort of side information, such as a Poisson model [Var96] or a Gaussian model for the TM entries, or the use of a previous model based on the gravity relationships among nodes (the so-called “gravity model” [ZRDG03]), among others.

The challenge of finding a good model for TMs is application-dependent, and therefore quite open. The criterion used for defining a TM model is sparsity, i.e., the model should have a small number of parameters. In the case of TMs, a network of N nodes has a TM of N2 entries (N origins x N destinations), and for typical sizes of N in the range of hundreds or even thousands, the TM can become too large to manage easily (for example, when storing TMs, predicting traffic or detecting anomalies). This sparse model will have M parameters, with M << N2.

To develop a sparse model for TMs, the Multi-Resolution Analysis (MRA) framework is used. Wavelet-based MRA techniques have been used for several signal processing procedures, such as denoising (by thresholding the wavelet transform coefficients) or compression (by keeping only the coefficients in the transformed domain that contribute most to the power of the signal) [Mal99]. However, standard 2D MRA analysis with wavelet transforms cannot be directly applied to traffic matrices, since a TM is not an image, and has more complex spatial relationships between its elements.

Therefore, new mathematical developments in the area of graph-based MRA are needed. This is a very new field and the theory is still being developed. A couple of examples of possible mathematical frameworks are provided, however, a different one may be used in the end, since it is possible new tools are developed during the life of the project.

In the first experiments [RRW08] a new technique called Diffusion Wavelets (DW) has been used to analyse the TMs' compressibility. The DW performs multi-resolution analysis of graphs and of functions defined on such graphs. The main idea behind the use of DWs [CM06] is that a diffusion operator "explores" the underlying graph and extracts the possible correlation caused by the spatial relationship in the traffic flows. For instance, traffic flows who share a substantial part of their paths may share characteristics such as their diurnal pattern. The first results [RRW08] are promising but there is still a long way to go to develop a solid, mathematically sound MRA model for TMs. Another promising research line consists of applying compressed sensing-based techniques [Don06] to the TM analysis, as recently reported by [ZRWQ09], in which (under certain conditions) a very reduced set of measurements can provide a very good estimation of the TM.

This research activity will thus consist of the following tasks.

- Identify GN3's needs regarding traffic matrix capture, inference and modelling. The targeted outcome of the work is an analysis of how perfSONAR can help to capture/infer/visualise TMs, either in real or deferred time, and how its existing capabilities can be improved.

- Survey and investigate techniques for visualising traffic demands based on graph multi-resolution techniques. The main goal of this is to explore graph- and time-based multi-resolution techniques in a study of GÉANT traffic matrices. The aim is to visualise trends in traffic demands at several timescales and levels of graph abstraction.

  This is especially important in the case of GÉANT, given its multi-domain nature. For example, at the highest level of abstraction, the tool may help to discover between which domains (NRENs) traffic is exchanged and how such transfers grow over time. At a lower level (nodes, links), with higher granularity, the tool may help in the planning and provisioning of new capacity. The tool may complement or use the measurements provided by the investigation into enhanced flow-level measurements (see *Enhanced Flow-Level Measurements* on page 37).

- Develop multi-resolution TM models and TM prediction techniques. Develop sparse, MRA-based models for TMs, apply them to the prediction of traffic demands, and investigate whether the quality of the predicted demands can improve, if the prediction is fed by measurements at the flow level, such as those that may be proposed by the investigation into enhanced flow-level measurements (see*Enhanced Flow-Level Measurements* on page 37). This aims to provide reliable and mathematically sound predictions of multi-domain traffic demands to network managers. As previously mentioned, TM modelling research has been focused in Intra-AS traffic demands, but few efforts have been made in the field of multi-domain TMs, which might have specific features (a "network of networks" such as GÉANT may have different traffic dynamics to a single-AS network).

3.2 Quality-Oriented Flow Monitoring

Traditionally, flow monitoring has mostly been used for security and accounting purposes. However, making quality measurements per flow is an attractive possibility. Such metrics can be aggregated to provide a high level view of quality of service and to investigate particular problems.

3.2.1 Metrics

The following metrics can be used to monitor flow quality:

- Bit rate

  The bit rate is measured over short time periods. Network traffic is highly variable over short time frames, while bit rates are traditionally measured over many seconds. Over the lifetime of a flow, maximum and minimum bit rates may be recorded, measured over intervals of 1 s, 100 ms, 10 ms and 1 ms. This will reveal episodes where flows are limited by line capacity.

- Packet distances

  Recording the distance between packets for each flow in a histogram reveals capacity utilisation. At full capacity utilisation, packets will be seen to arrive back to back.

- Distribution of packet lengths

  Per flow, maxima, minima and a histogram may be recorded.

- Packet reordering frequency

  Although it does not necessarily indicate that a problem exists, it is interesting to record the frequency of packet reordering.

- Mean, min, max and standard deviation

  Mean, min, max and standard deviation can be recorded to reveal jitter in voice and video.

Although not a quality metric, flow monitoring can also be used for application classification.

3.2.2 Technologies

The industry standard for flow monitoring is Cisco NetFlow version 5. Its flow records carry a fixed and very limited set of metrics. NetFlow v9 added a template mechanism, which allows administrators to configure a probe to report any combination of a large number of metrics.

IP Flow Information Export (IPFIX) is the Internet Engineering Task Force (IETF) standard for flow monitoring, and very similar to NetFlow v9. It includes enterprise-specific metrics, using an IANA registration mechanism to create enterprise numbers. This eliminates the risk of confusing custom metrics from different enterprises with each other. NetFlow v9 also provides a number of unused metric numbers, but it does not include a registration mechanism.

Flow monitoring is usually done at routers, but can also be done using passive monitoring cards and splitters that are connected to optical fibres. During the IST-SCAMPI project, a flow probe for passive monitoring cards was developed. This is part of MAPI, a measurement API which gives multiple measurement applications access to the same monitoring card. UNINETT has extended this flow probe so that it is able to export IPFIX as well as NetFlow versions 5 and 9. It supports the metrics listed above. Together with a corresponding collector based on nfdump, this system is called qflow.

At UNINETT, measurements collected using qflow are visualised using Stager. Stager presents aggregated measurements stored for the long term in a database, whilst the raw flow records are deleted after a few days or weeks.

3.2.3 Challenges

Keeping up with multi gigabit bit rates is challenging. If qflow is chosen, work will have to be done to optimise and parallelise the algorithms.

Measurements collected using qflow may be visualised using Stager (see *Possible Solutions* on page 20). Stager presents aggregated measurements that are stored in a database for a long time, whilst the raw flow records are deleted after few days or weeks. Stager reports can be retrieved as a perfSONAR service.

3.3 Passive Flow-Based Network Traffic Classification Using Machine Learning Techniques

The identification of network traffic flows by application type is challenging because applications evolve continually. Applications that are designed to be undetectable are particularly hard to trace. The diminished effectiveness of port-based identification and the overheads of deep packet inspection approaches have caused these approaches to become less popular and motivated researchers to classify traffic by exploiting distinctive flow characteristics of applications when they communicate on a network.

NRENs have a need for accurate and timely identification of networked applications based on direct observation of associated traffic flows (this is referred to as ‘classification’). Users of perfSONAR can use application identification for trend analysis (estimating capacity demand trends for network planning), adaptive network-based Quality of Service (QoS), marking of traffic, dynamic access control and lawful interception.

3.3.1 Performance Measurements

Machine learning and data mining applications have bottlenecks at memory and CPU consumption for high speed network traffic analysis. Where possible, parallel or concurrent algorithms must therefore be used. A test of several parallel processing libraries conducted within JRA2 Task 3 showed that the most convenient libraries for network traffic analysis are OpenMP and CUDA.

The most popular libraries for parallel processing are:

- OpenMP

  OpenMP is an API for multi-platform shared-memory parallel programming with C/C++ and Fotran. It has a very simple interface to use with pragma directives. [OpenMP]

- CUDA

  CUDA is an API for enabling parallel computation on GPU (Graphics Processing Unit). The CUDA engine is developed for C/C++ programming languages and it is accessible to developers via NVIDIA’s website for several platforms.

- PThreads

  PThreads is a POSIX standard for threading. This standard defines the API for creating and manipulating threads on UNIX-like POSIX systems such as Linux, FreeBSD, Solaris, NetBSD etc. [Hai]

For details of the performance of software libraries for parallelisation, see *Performance of Software Libraries for Parallelisation* on page 80.

3.3.2 Traffic Classification Algorithms

To find a suitable machine learning algorithm for network traffic classification, the following techniques can be considered:

- Unsupervised learning

  In this kind of learning the algorithm analyses data to determine how it is organised. This produces unlabelled sample clusters of data in which the data is grouped according to statistical features of the dataset (e.g.: Gaussian distribution, logistic regression etc.).

- Supervised learning

  A technique for learning a function from training data which consists of labelled samples.

- Semi-supervised learning

  Semi-supervised learning is halfway between supervised and unsupervised learning. In addition to unlabelled data, the algorithm is provided with some supervision information, The advantage of semi-supervised learning is that it can get the benefits of unlabelled samples in the classification phase. When classifying traffic much more unlabelled data than labelled data has to be dealt with. Therefore using semi-supervised learning should produce more accurate results. Several studies showed that semi-supervised learning suits traffic classification tasks, and that it can do near real-time classifications. [SSNTC] [ORTC]

The focus of the research will be on semi-supervised learning algorithms.

3.3.2.1 *Ensemble Learning Methods*

Ensemble learning is primarily used to improve the accuracy of the classification model, and to reduce the likelihood of a poor model being selected. The ways to use ensemble models are investigated based on the idea that using multiple models will provide better predictive performance than using any of the constituent models. [Hai]

The main ensemble learning methods are:

- Bagging

  Also known as bootstrap aggregating. This involves having each model in the ensemble vote with equal weight. To promote model variance, bagging trains each model in the ensemble with a randomly chosen subset of the training set.

- Boosting

  Boosting involves incrementally building an ensemble by training each new model instance to emphasise the training instances that previous models misclassified. Boosting is a general method for improving the accuracy of any given classifier. It is used with weak learners to get better performance by combining the classification results of each learner and to create a single strong learner [Boosting].

3.3.3 Training and Test Data Sets

Creating a credible and accurately labelled training data set for traffic classification is a difficult task in modelling real network traces, and it is impossible to create a training data set by labelling each flow manually. Therefore automated ways of exporting labelled flows are usually preferred. Inaccuracies and incorrect labels in the training data lead to wrong classification results.

An important problem during the training of the algorithm with the data set is overfitting. This occurs if a classifier is too complex for the training data set, so that a model is built which reflects the peculiarities of the training data set so completely that it misses the larger picture.

Creating a synthetic data set that perfectly models the nature of a large network (such as the NRENs) is nearly impossible. Therefore, network traffic flows need to be collected from live servers or clients that are running the applications to be classified.

Fine-grained labels are more informative than coarse-grained labels. But fine-grained labels can be semantically misleading, because they are specific to the applications, and the same types of application share the same traffic patterns. Therefore, if the labels of the VoIP application X are not available, X will be classified as another VoIP application Y, which has labelled traffic traces in the training data set. Thus traffic traces are, therefore, going to be labelled with fine-grained labels and the coarse-grained labels that the fine-grained labels belong to. For example:

VoIP: Skype, Gizmo, Yahoo, etc.

WWW: HTTP, HTTPS, etc.

P2P: EDonkey, DC++, Bittorrent, etc.

3.3.3.1 *Testing and Evaluating Algorithms*

Using training data to teach a classifier or predictor and to estimate the accuracy of the resulting learned model can result in misleading overly optimistic estimates, due to overspecialisation of the learning algorithm to the data. To evaluate the correctness of the classification done by the algorithm, the following classifier accuracy measures are used:

- Sensitivity
- Specificity
- Accuracy

In addition, ROC curves are going to be used for estimating the algorithm’s accuracy. ROC curves are two dimensional curves which are calculated by using the ratio between a True Positive (TP) rate and a False Positive rate, calculated from the labelled dataset. ROC curves are used for calculating and improving the accuracy of the classifiers [ROC03],

3.3.4 Multi Domain Passive Flow-Based Classification System Architecture

A good software architecture will provide reliable and secure application tools. Therefore, it is important to carefully define the software architecture. Netflow v9 is used to implement flow classification and the possible use of IP Flow Information Export (IPFIX) is investigated. The most important advantage of Netflow v9 is its IPv6 support.

As Nguyen et al discussed in their paper there are 3 categories of flows [Surv]:

- Flow or Uni-directional flow

  A series of packets sharing the same five-tuple traffic: source and destination IP addresses, source and destination IP ports and protocol number.

- Bi-directional flow

  A pair of unidirectional flows going in the opposite directions between the same source and destination IP addresses and ports.

- Full-flow

  A bi-directional flow captured over its entire lifetime, from the establishment to the end of the communication connection.

  The ideal flow type for traffic classification depends on how the flow classification system and the traffic classification algorithm are constituted. A typical traffic classification system comprises two major components (see Figure 3.1):

  - Netflow collector

    The basis component of the system for collecting flow records exported from routers. There are already many implementations of flow collectors like nfcapd from nfdump.

  - Traffic classifier

    The core part of the system. It carries out training and classification (see Figure 3.2 and Figure 3.3).

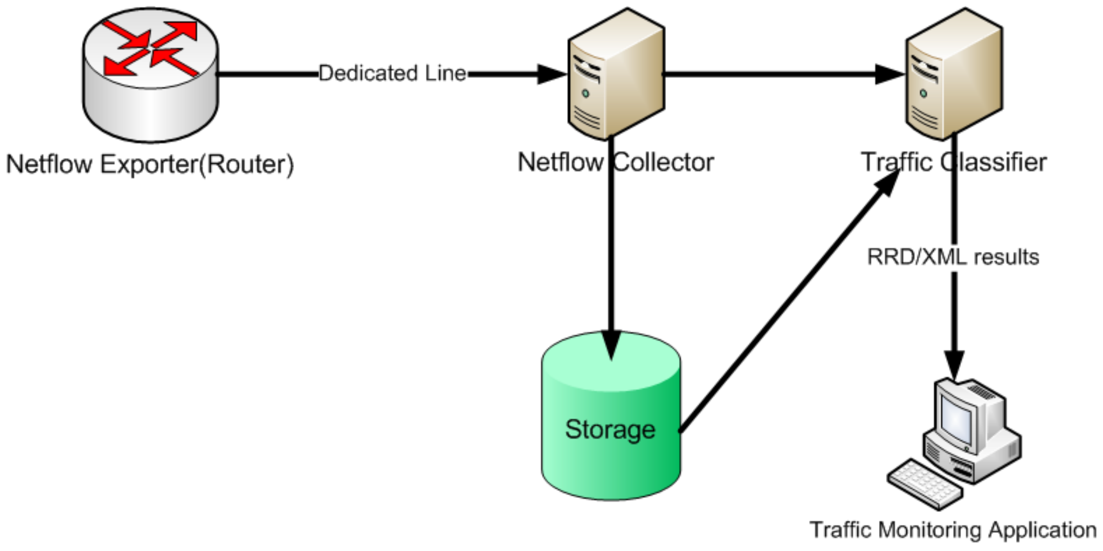


Figure 3.1: A Basic Traffic Classifier System Architecture.

Flows are collected using a Netflow Collector and a Traffic Classifier classifies the flows. The Traffic Monitoring Application generates statistics from the labelled flows for the end user.

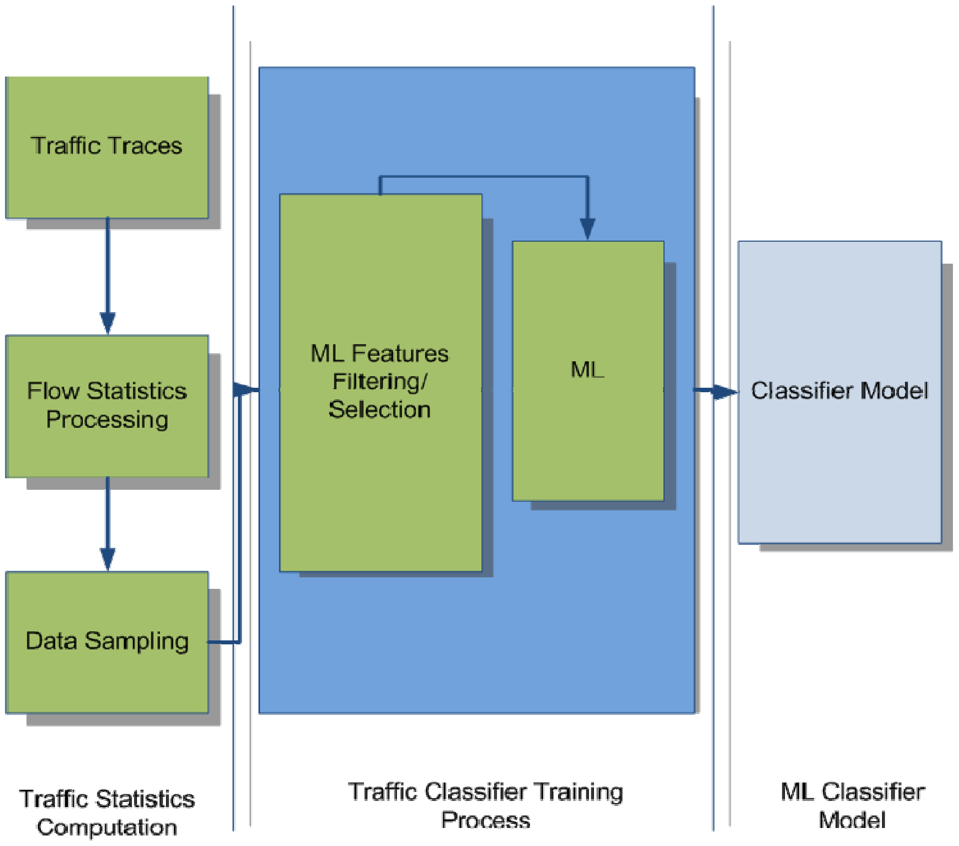

Traffic Traces
Flow Statistics Processing
Data Sampling
ML Features Filtering/ Selection
ML
Classifier Model
Traffic Statistics Computation
Traffic Classifier Training Process
ML Classifier Model

Figure 3.2: Training Supervised ML traffic classifier. [Surv].

As shown in Figure 3.2, after traffic traces are collected, flow statistics are gathered from those traces and then sampling is applied to the data set. Optionally, data sampling can be applied if too much training data exists. The feature selection process follows the sampling, which is the process of selecting the subset of the relevant data set features. The feature selection is applied to the remaining data set, producing a subset of attributes and discarding redundant ones. This yields better classification results and it also improves the performance of the algorithm. To build the classifier model, the machine learning algorithm is trained with the training data set.

In an operational semi-supervised machine learning traffic classifier application (see Figure 3.3), several steps are needed to get the labelled traffic traces. First flow records are captured. Then some statistical features of the flows are found. Next, if there is too much data, data sampling can be applied to the captured flows. The flow records are then fed into the classifier model which classifies the flows and outputs labelled flow data.

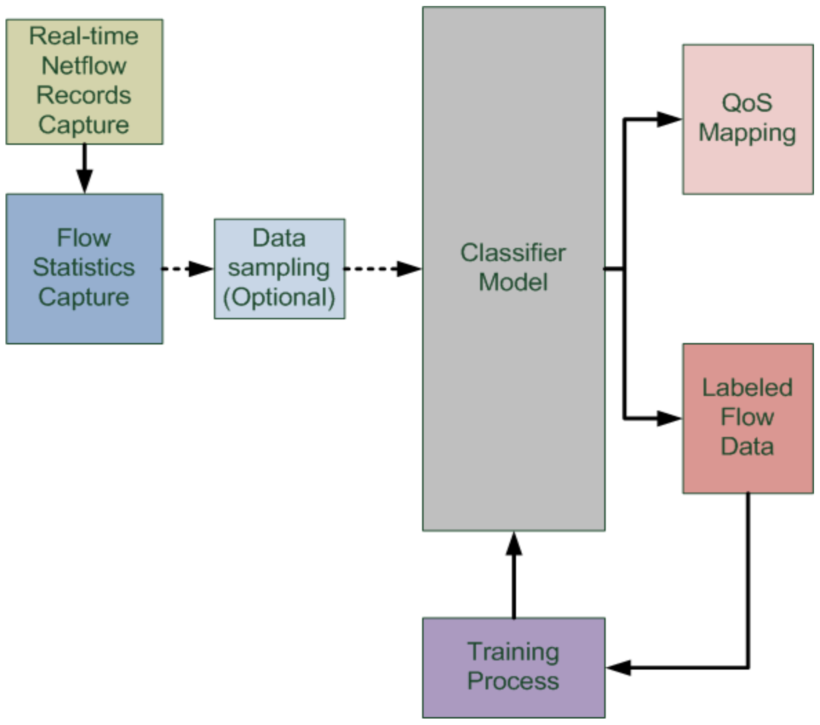


Figure 3.3: Data flow within an operational semi-supervised ML traffic classifier with self-training.

After the flow records are labelled, they can be analysed in several ways. For NRENs two of those analyses are specifically important. Using QoS, NRENs can measure the priority, usage statistics and users of different applications. Using trend analysis, users can collect information and spot patterns or trends in the classified dataset. Using previously labelled traces they can also predict the future usage in the labelled flows.

3.3.5 Advantages of Flow Based Traffic Classification Techniques

Studies on traffic classification of flow records are often carried out using data mining and machine learning techniques. The main advantage of these techniques over other approaches (e.g. port-based, DPI etc.) is that they can learn automatically. Flow-based approaches provide highly accurate classification and are, by nature, immune to link conditions, traffic conditions and noises. A flow based approach (FBA) is the most convenient way to classify traffic in high speed networks because it:

- Can adapt to the rapidly changing conditions of the Internet.
- Is immune to noise (unlike host behaviour based approaches). [Blinc]
- Will not fail due to a simple port change (unlike port based approaches (POBA)).[DYN]
- Will not fail for encrypted data (unlike payload based approaches (PABA)).
- Is not as computationally challenging as PABA.
- Relies on flow records (FR) which are extracted from the packet headers. Therefore FBA preserve the privacy of network packets (whereas PABA and host behaviour based approaches are inspecting the packet contents). [Blinc][Surv]

3.3.6 Implementation

The core classification software will be implemented in C as this is an industry-standard programming language with an emphasis on high performance. For the flow collector an open-source solution like nfdump will be used. The target platforms of the application are UNIX- and LINUX-based operating systems. The application will support both CUDA and OpenMP. Users will be able to use either CPU or GPU for parallel processing.

One of the benefit of this implementation is that it will be able to carry out high speed traffic analyses with low-cost hardware, no specific hardware is needed.

After the implementation is completed, unit tests, performance tests, stability tests, and security tests are going to be performed.

The results will be exported to RRD format and/or XML format on an hourly, daily and yearly basis. This will ensure compatibility with perfSONAR and other third party monitoring applications.

3.4 Enhanced Flow-Level Measurements

In addition to new flow-based traffic classification schemes (which may rely on existing flow-based meters such as NetFlow), new techniques for collecting traffic information at the flow level will be devised. The motivation for this is that conducting exhaustive measurements at the flow level is problematic because:

- Monitoring every flow present poses a scalability problem, due to the large number of flows that can be present in a link, and obliges routers to keep extensive states for all flows being tracked. Moreover, enabling flow-level tools at network nodes is known to impact forwarding efficiency.

- The exhaustive approach to flow monitoring requires exporting a large volume of measurement data. If activated at every MP, this poses new challenges in the design of a collection infrastructure. Exporting data for each flow could congest the reporting path to collecting stations, overwhelm the stations with more data than they can handle, and demand great storage capacities in the MA.

The investigation into what new techniques should be devised will concentrate on measurement techniques that can detect the largest traffic aggregates present in a link automatically and in real time. This will make it possible to discover activity patterns that are useful for traffic engineering, network trouble-shooting and network provisioning.

Research will concentrate on new techniques that can scale in terms of speed, memory usage and reporting bandwidth, so that the techniques can be implemented in DRAM (Dynamic RAM) and function at increasing link speeds. To achieve the latter, the techniques will rely on packet sampling methods in order to avoid processing (and memory operations for) every packet present in a link. Scaling in memory will permit a cost-effective deployment, avoid the need to keep large measurement archives, keep the bandwidth consumed for exporting measurement data low, and ease the sharing of measurement data between the domains.

Unlike existing tools, the techniques to be devised will be able to discover big flows without requiring prior knowledge of how flows are defined. This may enable them to report significant pieces of traffic with heterogeneous structure with minimal configuration and without human intervention. For example, they may discover that a single host is sending a high volume of traffic, or that all the hosts of a particular network prefix are producing a significant amount of traffic of a certain type, or that most of the traffic received by a set of nodes originates from a certain domain and is of a certain protocol.

Therefore, the result of this investigation could be the basis for a tool that would allow users to visualise the major trends of the bulk of traffic exchanged between the domains (at different timescales, such as minutes, hours, days or months), identify which domains exchange most of the traffic, or find out which applications are responsible for the bulk of traffic. Such a tool could be useful for several reasons:

- It may simplify the provisioning of new capacity, aiding in understanding who demands more capacity and under which patterns (continuous demand, sporadic demand, periodic, etc.).
- Visualising how much bandwidth is consumed by big traffic aggregates could make it possible to understand their effect on network performance, and whether the service provided meets the capacity demand of applications run by GN3 users.
- By reporting only the largest traffic flows, it could be much easier for the distinct domains to exchange measurement information and correlate measurements taken within each domain to those at inter-domain boundaries. This could open the door to the design of new tools that would aid different domains in collaborating to measure traffic.

If deployed at every link, the look and feel of the output supplied by the tool could be similar to that of a network weathermap. However, instead of visualising only the overall usage of each link (as done currently using SNMP counts), it could display a stacked graph of link usage, specifying the data volume and the properties of the biggest pieces of traffic (e.g. some IP source prefix, protocol, destination prefix, AS number, etc.). This would provide network operators with much richer information about how the networks are being used and by whom, without the need for exhaustive flow-level measurements.

Furthermore, the new techniques may provide network-usage information on a link-by-link basis. Thus, it may be necessary to correlate measurements taken at distinct links to gain a view of the spatial distribution of the big traffic contributors that are identified.

In parallel to the above research, techniques to discover the path or trajectory followed by large traffic aggregates will be investigated. However, compared to prior approaches (such as Trajectory Sampling) the new techniques will not require upgrading the existing network infrastructure with new functionalities and will thus be able to function with legacy equipment. This research will complement the work conducted on techniques for the inference and prediction of traffic matrices.

3.5 Lightweight Application-Layer Classification

To enhance the protocol identification/application-layer classification of the Available Bandwidth (ABW) [ABW] application, a new application-layer classification library will be developed using “blind” classification. This type of classification is based on a flow's statistical features and does not use any private data such as IP addresses, ports or packet payloads. This new library should be able to classify tunnelled, encrypted or anonymised data. It will be included in the DiMAPI middleware [DiMAPI]. Therefore, the functionality will also be available to other DiMAPI-based applications.

Results of network traffic classification will be stored in an RRD database, which will be accessible through RRD MAs. This will allow traffic classification results from other perfSONAR components to be accessed.

3.5.1 Introduction and Problem Description

Packet classification is needed in many monitoring applications. For example, to reduce or distribute traffic to multiple processes, or to compute statistics about specific hosts for traffic accounting. In classification, packets are marked as belonging to classes, which are then treated separately, for statistics or performance monitoring reasons. Routers classify packets to determine which class they belong to. Proper packet classification is the base for traffic shaping and traffic limiting. These are important for Internet service providers to fairly distribute the bandwidth among the users.

Port-based and deep-packet-inspection-based classification (a type of classification that is based on protocol knowledge) is still widely used. However, it is less effective or completely ineffective if tunnelling or encryption is in place, an application uses dynamic port allocation, non-standard port numbers or anonymisation, or if payload removal is used.

The protocol knowledge is represented by a protocol decoding automata or payload signatures, often hardcoded into the classification library for performance reasons. Maintaining such a library can be difficult. Adding a new protocol or changing an existing one requires library rewriting. Adding a proprietary protocol also requires additional work such as protocol reverse engineering.

Classification based on machine learning algorithms offers an alternative method of application classification. Instead of using protocol knowledge, it is based on statistical features such as flow duration, number of packets, packet length inter-arrival times, etc. A classifier based on machine learning is built by learning from a training set that consists of known application data. The built classifier is capable of considering whether the input data belongs to the same class it learnt or is unknown. Any captured data can be used as a training set. Therefore, it is also possible to identify a proprietary or unknown protocol.

Several papers have investigated the possibilities of using machine learning algorithms for packet classification [WilZanArm06, WangZheng09, NguGrenArm07, AngHey, AlsHeyTwo, AlsHeyDiff]. The results showed that it is possible to achieve very good classification accuracy with a detection rate of about 98% and a false positive rate of less than 0.5% even for encrypted traffic. The information presented in the next section is based on these papers.

3.5.2 Implementation Details

The ABW application is based on DiMAPI middleware and the protocol identification/application layer classification is based on the trackflib library. This is one of the DiMAPI libraries that implement monitoring functions for DiMAPI applications. These libraries are easily replaceable. There can be multiple trackflib libraries using different algorithms or different accelerating hardware for traffic classification.

The current trackflib library is based on protocol knowledge. A new library will be using "black box" or "blind" classification, based on a machine learning algorithm, extending the possibilities of the DiMAPI middleware. This should make it possible to classify anonymised, encrypted or tunnelled data.

The following issues need to be solved:

- Choosing a machine learning algorithm.
- Sourcing application tracefiles.
- Selecting flow features.
- Hardware acceleration and software tools.

3.5.2.1 *Choosing a Machine Learning Algorithm*

Many machine learning algorithms have plenty of fine-tuning options and different computational complexity. Choosing a suitable algorithm and its setup is one of the most difficult tasks. Based on information obtained from [WilZanArm06, WangZheng09], the following algorithms are suitable for classification:

- C4.5 Decision Tree (C4.5)
- Bayesian Networks (Bayes Net)
- Naive Bayes Tree (NB Tree)
- Naive Bayes with Discretisation (NBD)

Performance characteristic and classification accuracy were measured in [WilZanArm06] using a 3.4 GHz Pentium 4 workstation with 4GB RAM. The training set consisted of following flows:

- FTP
- Telnet
- SMTP
- DNS
- HTTP
- Half-Life

The results are shown in Table 3.1. The complexity column estimates the effort required to set up and fine-tune the algorithm. The build time is the time required to build a classifier (the length of the learning process).

Algorithm	Classifications per seconds	Build Time (seconds)	Complexity	Mean Accuracy (%)
C4.5	54700	23.77	Moderate	97.8
NB Tree	5974	1266.12	Moderate	97.4
Bayes Net	9767	13.14	Moderate-High	97.2
NBD	27049	10.86	Low	95.4

Table 3.1: Performance and setup complexity (FTP, Telnet, SMTP, DNS, HTTP, Half-Life).

The performance and accuracy of peer-to-peer classification is shown in Table 3.2. The training set consisted of the following flows:

- DNS
- HTTP
- eDonkey
- BitTorrent
- Kazaa

Only a subset of algorithms was tested in [WilZanArm06]. The same testbed was used.

Algorithm	Best Accuracy (%)
Bayes Net	98.98
C4.5	97.88
NBD	97.28

Table 3.2: Peer-to-peer classification performance (DNS, HTTP, eDonkey, BitTorrent, Kazaa).

As shown in Table 3.1, the C4.5 algorithm is the fastest, with a short build time and the best accuracy. For peer-to-peer classification, Bayes Net slightly outperformed C4.5. However, the classification speed of Bayes Net is significantly slower than C4.5.

Accuracy and classification speed are the most important factors in determining the right machine learning algorithm. The build time is also important. However, the algorithms can be built offline (prepared), so this factor has less importance.

Based on the information presented here, the C4.5 [C4.5] algorithm was chosen.

3.5.2.2 *Sourcing Application Tracefiles*

The classified application's tracefiles are used as learning data for the classification algorithm. Properly classified tracefiles imply the classification quality (accuracy). Getting classified tracefiles is not an easy task. Several options exist:

- Using publicly available tracefiles.
- Creating synthetic tracefiles.
- Collecting tracefiles from a network and classifying them.

Every option has its pros and cons. If publicly available tracefiles are used, their quality cannot be guaranteed. They are anonymised and without payload disabling any further classification. Creating synthetic tracefiles is possible, however, it is difficult to properly model the network environment. It is possible that the accuracy of a classifier built with this kind of data will be reduced. Gathering tracefiles from a real network seems to be the best option, but it is difficult to manually classify application flows in the captured traffic.

3.5.2.3 *Selecting Flow features*

Each training data set consists of packet flows that are represented by a set of features (flow attributes). There is a large number of attributes and their information value varies. To build a classifier based on all available attributes is not feasible as a high number of features dramatically increases the build and classification time without notably improving accuracy.

Machine learning algorithms typically use following flow attributes [WilZanArm06, AlsHeyTwo, AlsHeyDiff]:

- Protocol.
- Flow duration.
- Number of packets (forward, backward).
- Number of bytes (forward, backward).
- Packet length (min, max, mean, stdev) x (forward, backward).
- Packet inter-arrival time (min, max, mean, stdev) x (forward, backward).

The best attribute set will be selected during the implementation phase, by measuring the accuracy of different attribute selections.

At some network monitoring places (GN3-NREN boundary links), packet tracefiles are available for one link direction only. There are no forward and backward attributes. This will be investigated and an effort to find the best attribute selection for this case will be made.

3.5.2.4 *Hardware Acceleration and Software Tools*

The following software is expected to be used:

- NetMate [NetMate] – Open source flexible packet classification and filtering.
- Weka [Weka] – Open source machine learning software.
- DiMAPI [DiMapi] – Open source distributed monitoring API.

During the implementation phase the possibility of hardware acceleration will be determined. Various hardware cards and sources of flow records are available:

- DAG [DAG] – Passive monitoring and packet capturing card.
- Combo6 [Combo6] – Passive monitoring card with flow monitoring design capable of direct export of flow attributes.
- MTPP10 [MTPP10] – 10GE Modular Traffic Processing Platform.
- Other – External source of flow records (e.g. routers).

3.5.3 Conclusions

A series of practical experiments will be performed in wich the indicated hardware options, software tools and tracefile sources will be used to determine the dependence of classification accuracy on the used set of flow features. Particular attention will be paid to classification possibilities using traditional uni-directional Netflow records. Depending on the classification results, a hardware option to be used in the GN3 network will be recommended, and the corresponding classification application will be developed.

4 Authentication and Authorisation in the perfSONAR System

The authentication model for the perfSONAR system, which was designed during the GN2 project, centres around the development of the Authentication and Authorisation Service (AS).

Because clients and perfSONAR services are exchanging sensitive data, there is a strong constraint to provide a fully protected environment as soon as possible. To allow services to start taking advantage of the authentication process rather than having to wait for a full release, the development of the AS has been divided into two releases:

- The initial release was carried out in GN2 and covered the authentication process.
- The second release will be carried out in GN3 and implement authorisation.

The authentication web service can receive authentication and authorisation requests. Authorisation requests include authentication, so if the service receives an authorisation request (authR) it automatically authenticates (authN) the sender, too

The main goal of the perfSONAR AS is to allow any perfSONAR service to check whether the sender of a message is valid and has sufficient privileges to access the data they hold. Network domains that join the perfSONAR infrastructure can use the AS to protect resources within their domain from unrestricted access.

By configuring their web services to register with the AS, the domains can specify which request types require authentication before they are executed. This means that only users who have an identity provider (e.g. the GÉANT Identity Provider [GIdP]) account can send messages of the specified types to the domains' web services, while unauthorised users cannot access them. Once users are successfully authenticated, they can access the domain resources that allow them to access measurement data or to carry out measurements.

The AS accepts all identities that are issued by any eduGAIN-connected identity provider without any further checks. Any user can install an identity provider (using the Identity Provider (IdP) software) and get valid identities which are accepted by all AS deployments.

eduGAIN is one of the results of the JRA5 activity in the GN2 project, which sets the basis for interconnecting European academic users with ubiquitous networked services. Its purpose is to facilitate interoperation between different authentication and authorisation infrastructures.

The eduGAIN technology involves a translation of protocols between the ones used in local Authentication and Authorisation Infrastructures (AAIs) and Security Assertion Markup Language (SAML) as well as a mapping of attributes depending on local definitions.

However, although eduGAIN provides an excellent Authentication and Authorisation (AA) framework and meets all of perfSONAR’s AA requirements, the AS requires a specific development as the scenario of perfSONAR would be extremely complex if every service had to act as a Remote Bridging Element (R-BE) and this would make the service development slower. Therefore, the AS acts as the unique R-BE of perfSONAR in the eduGAIN trust model.

In this authentication model, clients must provide identity information (a security token) in every message they send to a perfSONAR service. The service includes the security token into the authentication request sent to its corresponding AS. The security token is built using credentials obtained from an eduGAIN component.

As the exchange of messages is based on SOAP 1.1, security tokens are sent using the Web Services Security (WS-SEC) standard published by OASIS [OASIS]. This avoids the perfSONAR protocol having to be modified, since the security tokens are added without altering the original message.

Three authentication and authorisation scenarios have been identified and implemented:

- Automated Client (AC) profile, for clients without any human interaction, based on X.509 digital certificates issued by the eduGAIN Public Key Infrastructure (PKI).

- User behind a Client (UbC) profile, for non-web-based clients, based on X.509 digital certificates issued by a Simple Authentication and Security Layer Certificate Authority (SASL CA).

- Client in a Web container (WE) profile, for web-based clients, based on SAML assertions issued by an Identity Provider within eduGAIN.

A group within JRA2 T3 will initially focus on finalising the developments started during the GN2 project: creating new versions of the AS and perfSONAR which provide both authentication and authorisation. The authorisation decision will be based on the user’s NREN or Virtual Organisation, information which is sent within the security token.

The following tasks will also be worked on during GN3:

- Definition of a full authorisation scenario, in which the AS retrieves user attributes from an external attribute authority. This task should be carried out in cooperation with the DICE [DICE] group to create a common authorisation model with other organisations.

- Redesign of the User behind a Client profile, since the model based on an online CA is not wanted by most organisations. The new profile will use DAMe (Deploying Authorization Mechanisms for Federated Services in the eduroam architecture), a project developed under GN2, which is able to get a SAML assertion from a Radius deployment.

- Session-based workflow with the AS. Every authentication or authorisation request sent to AS has to be signed, so it costs some seconds extra in the performance. The goal of this task is to provide a session capability to the authentication and authorisation model to improve the performance of the model.

5 Advanced Traceroute Functionality in the perfSONAR System

Many performance problems in the current Internet are caused by misconfiguration, faulty components or exceeding link or node capacity in some specific point in a network path. This section describes an advanced traceroute that helps to identify these problems. Similarly to a standard traceroute, an advanced traceroute traverses an end-to-end path, but it also provides performance information about links and nodes in both directions of a network path. This information is collected from measurement points of the perfSONAR monitoring framework. An advanced traceroute also correlates forward and backward traces and tries to identify asymmetries.

5.1 Problem Description and Requirements

The standard traceroute program provides reachability information and the RTT for each hop along a network path. This, however, is not sufficient to debug performance problems, and both characteristics are sometimes inaccurate. Some hops (routers or switches) do not respond to traceroute queries because the response is disabled on the hop. If such responses are permitted, they are usually provided by a lower priority process. As a result, RTT experiences fluctuations that may not relate to the actual load on the links in a measured network path.

An advanced traceroute is required to:

- Operate in both directions and identify asymmetric parts of the network path.
- Provide characteristics that can be used to locate performance limiting points.
- Allow extensions for measurement, storage and presentation of additional network characteristics.

Considering a typical process to debug end-to-end performance problems, the network characteristics that should be provided by an advanced traceroute for each link or hop in a network path should ideally include the following:

- Installed capacity.
- Currently available or occupied capacity.
- Packet error rate (including reason).
- Queue occupancy and utilisation on the hop before the link.

While installed capacity is constant, other characteristics vary in time and therefore need to be complemented by information about the time period for which they were measured, and about the statistics used to compute them (average, maximum, etc.). Packet loss is an important characteristic, but it is even more useful if the cause why packets are being dropped is known (either some kind of a packet error or queue overflow). Queue occupancy is the number of packets held in a queue (which affects latency), while queue utilisation is the percentage of the queue capacity that is currently occupied.

Some of the currently available implementations of traceroute-like programs that represent various measurement and presentation techniques have been reviewed. For details about each reviewed program, see [Ple06].

5.2 Jtraceroute Architecture

A prototype of the advanced traceroute tool called Jtraceroute (as „Java traceroute“) has been implemented (by CESNET [CESNET]). The Java language was selected for the implementation, because the tool needs to run directly from the user PC, independently of the underlying which can run on various operating systems.

The above stated requirements require support from the network infrastructure. The given characteristics can not be measured from end-hosts only. An obvious candidate for communication with routers is the SNMP protocol. It is a common practice that access to routers by management protocols is restricted to a few IP addresses where management stations are running. This information needs to be accessed from the PC of an unprivileged user.

Therefore, it is necessary to use proxy devices that mediate access to the routers along a network path. One technology based on this paradigm is the perfSONAR framework. For the advanced traceroute an SNMP MP was developed, which uses the SNMP protocol to retrieve data from routers. A common installation is to deploy one SNMP MP in each network, under separate administration, which decides to participate in the advanced traceroute infrastructure. Each SNMP MP can be configured to authorise requests for specified MIB objects on specified routers.

Measurement results are stored in an XML file and can be retrieved and presented later. A format recommended in an Request for Comments (RFC) document [RFC5388] for storing standard traceroute results has been extended to store additional measurement results.

Jtraceroute operates in the following steps (see Figure 5.1):

1. When the SNMP MP in each network starts, it automatically registers with the LS.
2. A user starts Jtraceroute from a web page using Java Web Start technology.
3. Jtraceroute performs a standard traceroute to specified the IP address or hostname.
4. Jtraceroute asks the remote end to perform a backward traceroute. This succeeds if the remote end also runs Jtraceroute (which responds to these requests), otherwise backward traceroute is not done.
5. Jtraceroute tries to correlate the forward and backward traceroute to identify possible asymmetries

6. Jtraceroute (on both ends of a tested network path) asks the LS for the URLs of the SNMP MPs, which serves the IP addresses of all forward and possibly backward router interfaces.
7. Jtraceroute (on both ends of a tested network path) asks all SNMP MPs for selected performance characteristics. This query is sent twice, with the configurable delay, in order to obtain rates per second, such as interface utilisation from two values of transferred bytes.
8. Jtraceroute writes the results to an XML file.
9. Jtraceroute presents the results graphically.
10. Results can be presented again later from a stored XML file.

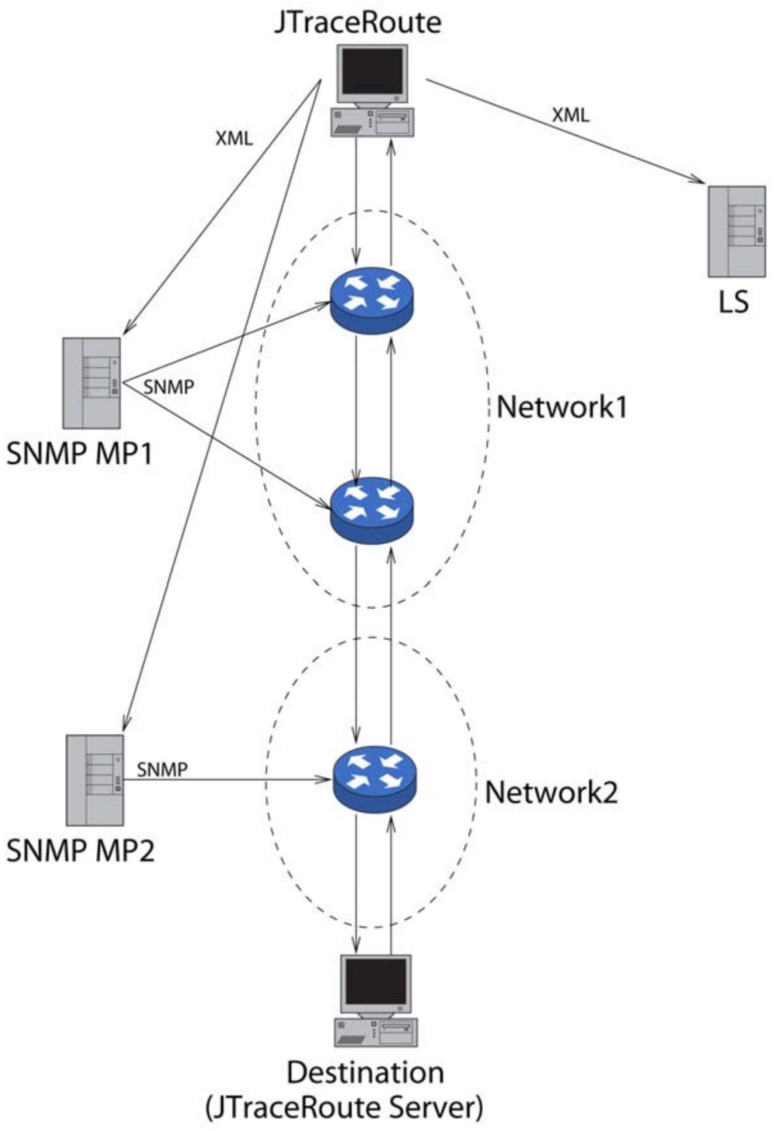


Figure 5.1: Example Figure.

For a usage example, see *Example: Jtraceroute Use* on page 81.

5.3 Conclusions

Jtraceroute is a prototype of an advanced traceroute that can retrieve, store and present performance characteristics of forward and backward interfaces along a network path. The participating network needs to run a SNMP MP which acts as SNMP proxy to the network routers. Otherwise standard traceroute results are presented for the corresponding part of the network path.

Jtraceroute currently provides a subset of performance characteristics identified as useful for performance debugging. Characteristics such as interface queue occupancy are not provided by routers in the CESNET network [CESNET], which was used for testing. Jtraceroute can be extended to retrieve more characteristics specified by MIB object identifiers.

6 Monitoring the AutoBAHN System with perfSONAR

AutoBAHN is a provisioning system that enables on-demand bandwidth reservation for links across multi-domain networks. Each individual network is operated differently, as required by the different technologies that the networks are based on. This diversity has been a challenge in GN2 when aiming to provide an agreed service level at the interdomain level.

AutoBAHN is a solution to this problem. The system provides a common business plane that enables users to easily control the inter-domain connection parameters using the Inter-Domain Manager. This component is responsible for technology-agnostic inter-domain communication. Each domain is operated by a Domain Manager (DM). This component is responsible for operations within a specific domain (for example, resource reservation, signalling, providing topology information etc.). The Domain Manager communicates with the underlying network using the Technology Proxy. This is a web-service-based interface that connects the DM with the control/management plane or provisioning system.

While the AutoBAHN system enables users to perform various management tasks necessary to provide a required service level, it lacks essential monitoring capabilities which could make it a more complete solution for multi-domain connection management. During GN2, a prototype monitoring module for the AutoBAHN system was specified and developed. However, this was tightly integrated with AutoBAHN system components and workflows. In the meantime, the perfSONAR framework has been elevated to a production environment and is gaining approval as a distributed monitoring solution in the GEANT-NREN community. At this stage, it will be beneficial for both systems to join forces. In this way, AutoBAHN will exploit a modular monitoring infrastructure and perfSONAR will be extended from IP monitoring to dynamic (usually L2) circuit monitoring.

The monitoring solution proposed for AutoBAHN during the GN2 project was based on an AutoBAHN Domain Manager module extension that handled monitoring within each domain participating in the inter-domain path. This DM Monitoring module was responsible for operating a Secure Scripting Server (SSS), through which it would access the network elements that the intra-domain path was using - in the case of Ethernet-technology-based domains or the NMS controlling the network elements in SDH-based domains. Technology-specific monitoring data (e.g. lost frames for Ethernet domains, errored seconds for SDH domains) would be received via the SSS and fed to the AutoBAHN DM where association with topology data and dynamic circuits would be made. This detailed per circuit monitoring information was then abstracted (at the technology level) and collapsed (at the topology level, wherever a domain did not wish to reveal detailed monitoring data across its topology) and passed over to the AutoBAHN Inter-Domain Manager module for communication to the neighbouring domains and/or the Visualisation client.

Abstraction and aggregation would be dictated by the need to concatenate heterogeneous technology specific metrics (e.g. frame losses with errored seconds) and hide intra-domain details (when required) at the inter-domain level. Inter-domain monitoring data posts from AutoBAHN IDMs towards the Visualisation client would follow the perfSONAR E2EMON [E2EMON] XML schema to ensure seamless visualisation of AutoBAHN monitoring data according through the E2EMON GUI.   However, the information this system provided during the prototype phase was limited. The E2E monitoring could only present the link status of each virtual link (an aggregated link representing the connection between domain entry and exit point).

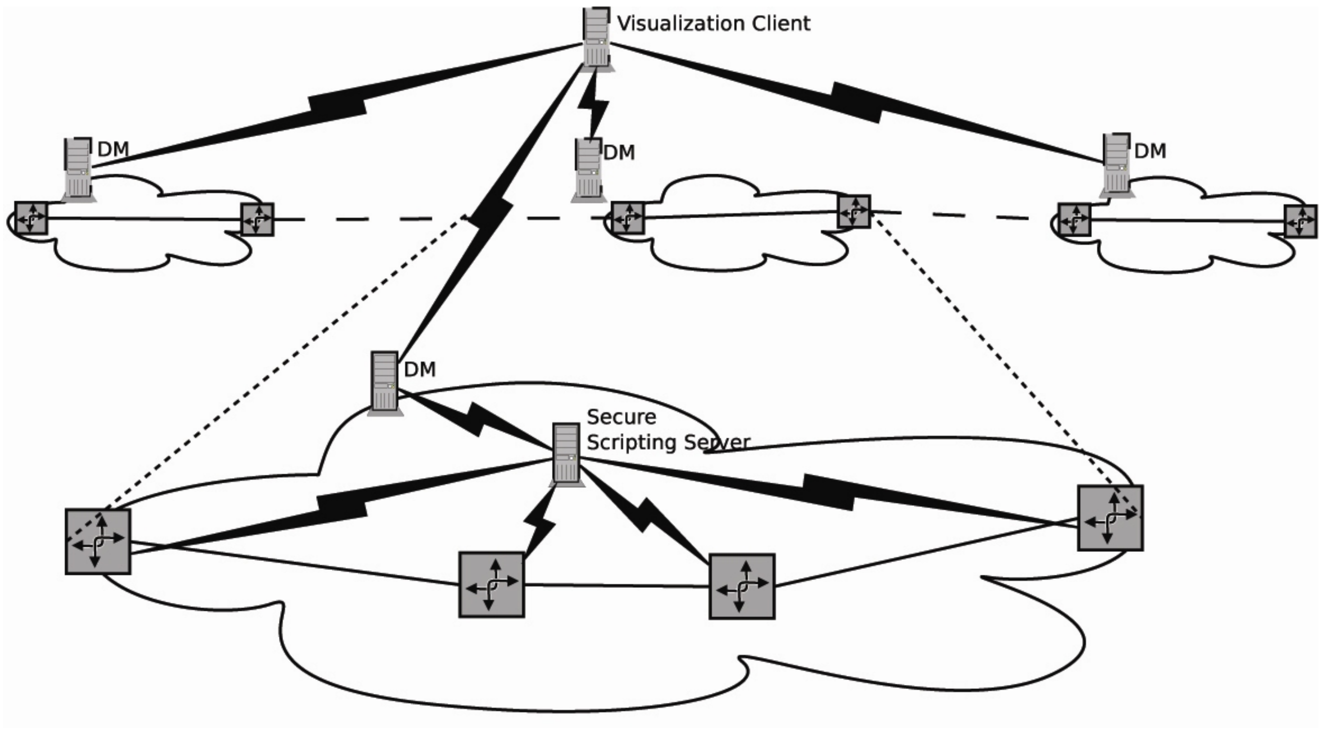


Figure 6.1: Overview of existing AutoBAHN Monitoring module operation principles [AutoBAHNE2Emon].

Internet2 also developed a dynamic circuit monitoring solution which is following the perfSONAR architecture more closely A monitoring agent is introduced to the dynamic circuit provisioning suite (the Inter-Domain Controller module, corresponding to the AutoBAHN IDM) to dispatch monitoring tasks to several Performance Collectors and archives, as new circuits across the domain are being established.

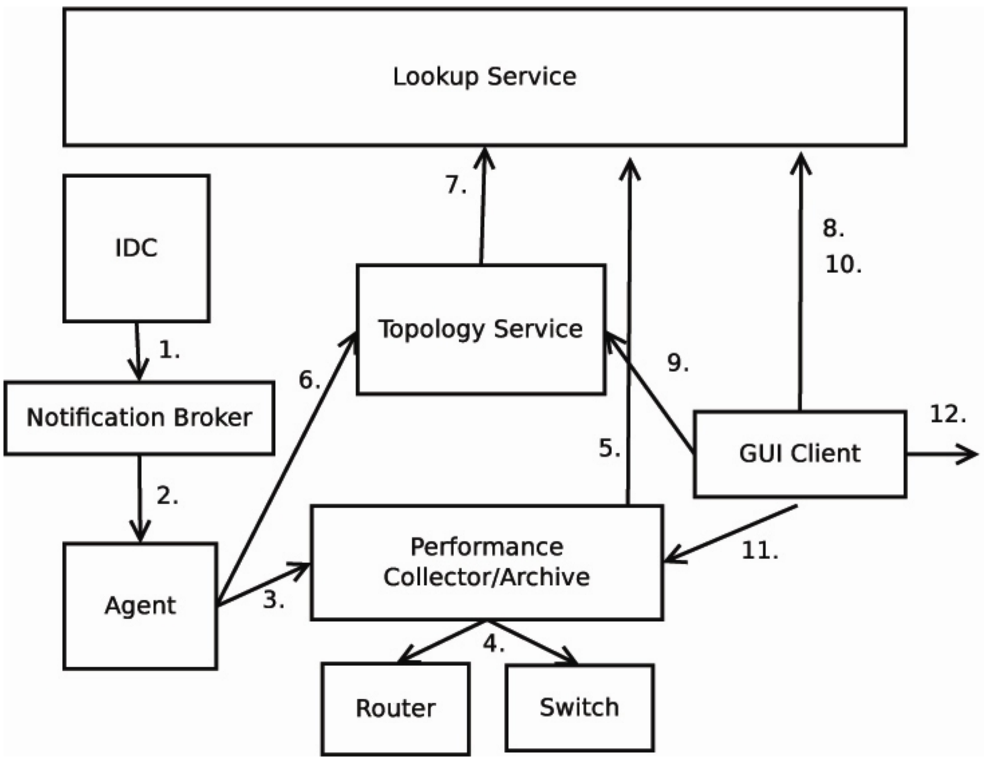


Figure 6.2: Circuit Monitoring Architecture as proposed by Internet2 [CircMon].

The infrastructure built according to the Internet2 concept works as follows:

1. The Inter-Domain Controller sends a message to the Notification Broker when a new circuit is allocated.
2. The Notification Broker passes the information to the Agent.
3. The Agent dispatches the monitoring activities across multiple Performance Collectors.
4. The Performance Collectors start the required monitoring activities.
5. The Measurement Archive registers information about the circuit monitoring data.
6. The Agent registers information about the circuit with the Topology Service.
7. The Topology Service registers information about the current circuit with the Lookup Service.

Monitoring information is gathered from such a system as follows:

8. The client asks the Lookup Service for a Topology Service that has information about the circuit.
9. The client retrieves the circuit information from the relevant Topology Service.
10. The client asks the Lookup Service for Performance Archives from which it can request monitoring data.
11. The client contacts the relevant Performance Archives directly to obtain monitoring data.
12. The client presents the data to the user.

The modular approach to interfacing a dynamic circuit provisioning system with a monitoring platform (as suggested by the Internet2 approach) will be investigated within GN3 through the integration of AutoBAHN with perfSONAR.

It is worth mentioning that both systems are already well-established and have been tested in real operating environments.

The investigation and specification of a modular dynamic circuit monitoring solution that exploits the best of AutoBAHN and perfSONAR will result in a specification document. This will include definition of interfaces for bi-directional communication between the two systems. A number of challenging issues arise when trying to design such a modular approach. For example, decoupling the monitoring functionality from internal functions of the AutoBAHN system means that the monitoring system has to be kept consistent with the topology and metric aggregation functions in AutoBAHN and vice versa. Once such fundamental issues are successfully addressed, the specified solution is also expected to prove useful for more advanced scenarios, e.g. supporting feedback loops, where the network manager takes actions (using AutoBAHN for network (re)configuration) based on information gathered by perfSONAR services. The specification is currently being developed collaboratively by SA2 Task 5 and JRA2 Task 2.

7 Events Handling in the perfSONAR System

Today, a software management solution for monitoring complete networks not only has to offer passive and active traffic measurements, but also scope for events handling (incidents management). The lack of this functionality can be seen as one of the main weakness of the existing perfSONAR implementation as a multi-domain-monitoring infrastructure.

This section compares perfSONAR's functionality with that of some commonly used commercial network management systems like HP OpenView and IBM Tivoli Netcool. Particular attention is paid to how events management is implemented in these systems. Events management is often connected with events correlation and deduplication mechanisms which make it possible to handle a large number of events efficiently, and to perform some actions automatically, triggered by a defined set of events. Mechanisms that are not implemented in perfSONAR are described, and advantages and disadvantages of automatic actions, which can be performed as a consequence of some correlations, are analysed.

With an increasing number of network elements and complexity of network issues, an event management system that is capable of correlating different network events (syslog, trap, log files) may be considered. This architecture behind an event management system is comparable to a Manager of Managers (MOM) system. A well-designed event management system allows personnel in the Network Operations Center (NOC) to be proactive and effective in detecting and diagnosing network issues. Event prioritisation and suppression allow network operation personnel to focus on critical network events, investigate several event management systems, and conduct a feasibility analysis to fully explore the capabilities of such systems [EMS].

The correlation and filtering mechanisms for events are crucial to avoid the pollution of GUIs with storm of events, especially in real telecommunications networks. Such networks can generate more than a hundred thousand events per day (e.g. Embratel's Frame Relay and ATM networks management station receives around 130 000 events daily [FRATATM]). Additionally, users are allowed to configure functions that are executed upon receiving an event or a defined set of events. This can mask some events and generate new ones, which can be more significant than originally received events (for example, if node A has LoS and node B has LoS, then a new, more meaningful event can be generated - for instance a fibre break between A and B), helping the operator to identify the root cause of the problem.

Some functionalities that exist in well-known commercial network monitoring tools (see *Example: Commercial Network Management Tools with Events Handling Functionality* on page 87) are still not covered by the perfSONAR system, especially in the area of events handling and processing by correlations mechanisms. This can be seen as the major weakness of perfSONAR and it may be worth to identify the best method and implement this functionality to fill the gap.

However, first it should be considered whether events handling is really needed in perfSONAR. This depends on whether the aim is to make perfSONAR a better network monitoring tool or to treat it as performance measurement tool only.

To make it easier to find an answer to this, the pros and cons of such an extension of the perfSONAR architecture need to be analysed. The extension of perfSONAR's existing functionality would be an obvious advantage. This could make the tool a good alternative to commercial software for general network monitoring purposes. With the inclusion of an events handling system, perfSONAR could be a free and open source infrastructure not only for network performance monitoring but also for fault management purposes.

A disadvantage of such a solution would be the additional complication of the perfSONAR system. A new service would need to be developed to collect various events from network devices, aggregate, correlate, filter and send them to clients (e.g. PerfsonarUI instances) that are interested in events handling from selected IPs. The client implementations would also need to be properly extended to be able to use the new services functionality. Another disadvantage could be the impact on perfSONAR system performance. The new functionality could decrease the application performance seen by the user, especially because of large amounts of data being sent between network devices and this new service, and between the new service and its clients.

Generally, arguments against events handling that are based on performance impact are outweighed by the obvious advantages that the additional functionality would provide. Therefore, the perfSONAR architecture should be extended with the following proposed events handling functionality:

- New web service

  A new web service should be created for various events that are sent from different network devices. The new service could, for example, be called Events Archive (EA) and one could be installed for each domain. All managed network devices working in one domain should then send various events (e.g. SNMP traps or Syslogs) to this service. After receiving an event from the monitored device, EA should then store it in a local database.

- LS registration

  The EA service, as any other service that forms a part of perfSONAR, should register with the Lookup Service (LS) to make gathered events available to different clients in a multi-domain environment.

- Event accessibility

  Events stored in EA's database should be accessible to perfSONAR clients (e.g. PerfsonarUI) that have been authorised by the Authentication Service (AS). Authorised clients would only get events they are interested in, e.g. from selected devices.

- NMWG protocol

  Events should be sent to clients using perfSONAR's NMWG protocol. A problem that needs to be resolved before implementation is that the NMWG protocol does not support one-way communication (e.g. SNMP Traps). There is always a request and response in this protocol. However, the client should get new events only on demand, after sending a request to the EA service. This means that the protocol has to be extended or changed to enable this.

As an alternative to introducing a new service (EA), another architecture extension should also be considered and analysed. The specialised MP could receive all events form devices and then send them to the appropriate MA service. The MA could store the received events in its database and make them accessible to its clients. This may be a better solution because the typical to perfSONAR functionality distribution between different services can be kept without the need to introduce EA.

Once the different architecture extensions for enabling events handling functionality have been analysed and researched, the best solution will be chosen. The extended perfSONAR architecture will then allow AS authorised clients to access multi-domain services, so they can get events from all devices they want to monitor in the multi-domain network. This new functionality could benefit real networks like GÉANT and the Large Hadron Collider Optical Private Network (LHCOPN [LHCOPN]) by making it possible to monitor not only multi-domain performance issues, but also to perform fault management through handling events in the different NRENs in the Pan-European network.

8 Vertical Cross-Layer Network Monitoring in a Multi-Domain-Monitoring Infrastructure

During the GN2 project the main focus in the development of the perfSONAR system was on providing core components and IP monitoring functionality (a few basic metrics). The scale and complexity of this work did not allow more features to be defined within the time scale of the project. However, the open and expandable architecture of perfSONAR provides the scope to add new features in a flexible way and to continue to enrich the system.

One of the most important tasks that JRA2 T3 has to tackle is to investigate and provide the concept of complete cross-layer monitoring. The planned work should allow to link relevant monitoring results, which represent more than one network layer, in a dynamic and consistent manner. A user (network engineer) should be able to easily observe monitoring parameters in two dimensions: horizontal (multi-domain) and vertical (multi-layer).

The first step to achieve this goal is to propose a detailed and standardised information model of a multi-layer network. Single layer information models are well known, and multiple models exist, usually based on the concept of graph theory. However, cross-layer network description is a new field, and only few models exist.

The JRA2 T3 group will analyse existing cross-layer network descriptions and pick the most promising one. If the analysis shows that features required for cross-layer monitoring are missing, the group will extend the existing model with these features.

Based on an early analysis of existing work, the group will examine the following models, standards and efforts:

- ITU-T Recommendation G.800, “Functional Architecture of Transport Networks”, defines a set of building blocks to describe cross-layer networks. However, it does not define how these building blocks are to be used in an information model and may not be complete.

- The Common Information Model (CIM), defined by the Distributed Management Task Force (DMTF [DMTF]) , describes a data model for an Information, Communications and Technology (ICT) infrastructure, including the network. The model can only describe a limited set of network technologies.

- NETMOD and NETCONF, defined by the IETF are data models to describe the state of a network device. The model is currently not capable of describing links between devices.
- Network Description Language (NDL), developed by the University of Amsterdam is an RDF-based data model to describe multi-layer networks.
- The common Network Information Service (cNIS), defined in GN2, is a relational database to describe multiple layers of the GÉANT network and NRENs wishing to adopt it.
- The Unified Network Information Service (UNIS) is a network model developed for perfSONAR and the Interdomain Controller (IDC). There have been three revisions of the UNIS schema.
- The Network Markup Language (NML) working group in the Open Grid Forum (OGF) is chartered to standardise a multi-domain network model. cNIS, NDL and UNIS developers have contributed to OGF, and several groups have shown commitment to adopt NML as soon as it is ratified.

The JRA2 T3 group has defined criteria (see *Criteria for the Selection of an Information Model* on page 59) for comparing these different efforts, and assess each effort based on these criteria. This assessment will result in a recommendation by the group.

The information model recommended by the JRA2 T3 group will be used to assign network objects (physical and abstract ones) with monitoring data collected and stored in the perfSONAR infrastructure. Domains and layers will be monitored and analysed seamlessly.

8.1 Criteria for the Selection of an Information Model

This section presents a preliminary version of the criteria that will be used to select the information model. The list will be further refined during the remaining course of the project.

In addition to the criteria outlined in this section, weights and grades will be introduced. The weights will emphasise the importance of each criterion. The grades will be assigned to each potential solution, depending on how well it scores for each criteria item. This will result in a weighted mean score for each information model and thus the recommendation.

8.1.1 Scope

The model must be compatible with any choices made by the SA2 activity (Multi-Domain Network services), so that it can be used within any GN3 product (e.g. perfSONAR, AutoBAHN, AMPS, I-SHARe, etc.). Compatibility with other JRA2 tasks (especially JRA2 Task1, Control and Management) is also very important. Furthermore, the chosen model should be able to support monitoring, path finding and inventory management. Other usage such as path provisioning and visualisation are not considered in this evaluation.

Summary of scope criteria:

- Compatibility with SA2 choices.
- Compatibility with other JRA2 tasks' choices.
- Compatibility with other models.
- Support for monitoring.
- Support for path finding.
- Support for inventory management.

8.1.2 Features

The information models will be graded based on the features they support. Models which support multiple technologies (e.g. WDM, OTN, SDH, Ethernet, etc.), and models that can easily be extended or combined with other models will rate higher. Models that support the incorporation of multiple layers and multiple domains are also preferred. Models are furthermore judged according to their complexity. A simple model may be easier to implement, but may have limited use.

Summary of feature criteria:

- Supported technologies.
- Extensibility.
- Complexity.
- Multi-layer coverage.
- Multi-domain coverage (aggregation).

8.1.3 Maturity

The information model should preferably be mature in terms of the standardisation status and should have a stable schema. Furthermore, it should be widely deployed and implemented. A model that can be expected to receive more support from the GN3 SAs and US partners will rate higher.

Summary of maturity criteria:

- Standardisation process.
- Schema stability.
- Number of implementations.
- Stability of implementations.
- Number of deployments
- Expected support/number of maintainers (GN3 SAs, US partners, etc.).

8.1.4 Involvement

The information model's results are more likely to be applicable in the GN3 project if GEANT partners are involved in the development of the standard. To encourage uptake of the chosen model, it is beneficial if it is standardised and available without legal or economic hindrances. For this reason, models that are released under an open source license will receive a higher rating.

Summary of involvement criteria:

- Current involvement of GEANT partners.
- Membership requirements of standardisation body.
- Distribution of standard (can the draft or ratified standard be distributed?).
- Work pace (engagement).

8.2 Existing Information Models for Evaluation

Table 8.1 provides a brief overview of in how far each possible model conforms to the previously detailed criteria. The table does not contain ITU-T G.800 as it is not an actual model, but merely defines functional elements.

Scope						
Criteria	**CIM**	**cNIS**	**NDL**	**NETMOD/ NETCONF**	**NML**	**UNIS**
Choice in SA2	Not in use	In current use	Not in use	In limited use	Not in use	In current use
Choice in other JRA2 tasks	Not in use	In current use	Not in use	Under consideration	Not in use	In use in JRA2 T3 only; not in other tasks
Compatibility with other models	Compatible with DMTF standards	No specific references to other standards	Partly compatible with G.800	Unknown	Partly compatible with G.800; compatible with NSI	No specific references to other standards
Monitoring support	Limited support	Good support	Good support	Very good support	Undefined; Good support expected	Very good support
Path finding support	No support	Good support	Very good support	No support	Undefined; Good support expected	Very limited support
Inventory management support	Very good support	Very limited support	Limited support	Good support	Limited support expected	Very limited support

Features						
Criteria	**CIM**	**cNIS**	**NDL**	**NETMOD/ NETCONF**	**NML**	**UNIS**
Supported technologies	IP, basic Ethernet only	WDM, basic Ethernet, limited IP	Fibre, WDM, Ethernet	Mostly IP, Ethernet, wireless	Undefined yet	Fibre, WDM, basic Ethernet, IP
Extensibility	Extensible through changes only	Extensible through changes only	Extensible and combinable	Extensible with modules	Undefined yet	Extensible and combinable
Complexity	Complex, but limited use	Intermediate complexity and limited use	Intermediate complexity and flexible	Unknown	Undefined yet	Simple, and intermediate flexibility
Multi-layer coverage	Very limited coverage	Good coverage	Very good coverage	Very limited coverage	Very good coverage	Good coverage
Multi-domain coverage	Very limited coverage	Limited coverage	Good coverage	No coverage	Very good coverage	Limited coverage
Maturity						
Criteria	**CIM**	**cNIS**	**NDL**	**NETMOD/ NETCONF**	**NML**	**UNIS**
Standardisation	DMTF Standard	Not standardised	Not standardised	IETF standard	OGF standard	OGF standard (version 2)
Stability	Mostly stable	Limited stability	Limited stability	Stable	In development	Limited stability
Number of Implementations	Unknown	One implementation	Few implementations	Many implementations	No implementations yet	Few implementations
Stability of Implementations	Unknown	Ready for early deployment	Only custom implementations	Unknown	Not applicable	Ready for early deployment
Deployment	Unknown	Deployed at NRENs	One or two sites only	Widely deployed	Not deployed	Deployed at NRENs
Maintainers	Unknown	GEANT supported	No support	Unknown	Not applicable	GEANT and Internet2 supported

Involvement						
Criteria	**CIM**	**cNIS**	**NDL**	**NETMOD/ NETCONF**	**NML**	**UNIS**
GN3 involvement	Not involved	Heavily involved	Involved	Not involved	Heavily involved	Clearly Involved
Membership requirement	Free for academia	Not applicable	Not applicable	Free for all	Free for all	Free for all
Distribution	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited
Engagement	Continuous development	Slow development	Slow developme nt; efforts diverted to NML	Unknown	Commitment from NREN and OGF community	Slow development; efforts diverted to NML

Table 8.1: Information model overview.

8.3 Selection of Information Model

Since the group has not yet defined the weight of each criteria, no selection has yet been made. A preliminary choice is planned to be made by the end of 2009 and to be ratified by other interested consortium partners in early 2010.

8.4 Development of the Data Model

Some information models also define a data model; that is, a syntax and mapping of the model into a language-specific structure (e.g. into XML, RDF, relational database or UML model) or into software. If the chosen model is only an information model, but not a data model, the JRA2 Task 3 team will work with the given standardisation body to turn the information model into a data model.

Both the information model and data model for vertical cross-layer network monitoring will be used to store, manage and publish measurement data in the perfSONAR system. Details will be investigated but it can be presumed that the key element would be a service of the type Measurement Archive (MA) available in the perfSONAR infrastructure for other services and client (visualisation) applications. Measurement Point (MP) services for network monitoring at layers below IP are also expected.

9 Use of SLA in a Multi-Domain Environment

Service Level Agreements (SLAs) were not introduced in perfSONAR during the GN2 project. The main reason for this was that the work on core functionalities (basic IP monitoring managed and conducted by a dynamic, distributed infrastructure) had higher priority. In GN3, the first development phase of perfSONAR is completed and the second one will be started when new extensions can be added. One of these is SLA functionality.

SLA is a well known concept of mapping and managing the consumer's service quality expectations and the provider's service quality offer. The relationship between the two becomes more complex if a service is more advanced (multiple domains, multiple policies, multiple technologies and standards, etc.). If more than one domain is involved, two kinds of SLAs can be considered:

- Between the service providers (more than one are involved in providing a service to a consumer).
- Between service provider and consumer (the service provider is directly connected to the customer).

To prepare a framework specification for SLA functionality within perfSONAR, the following work needs to be undertaken:

- Investigation of availability and formats (compatibility) of measurement data in the perfSONAR infrastructure.

- Investigation of existing and missing functionalities needed for SLA management in perfSONAR.

- Work on the SLA information flow in the perfSONAR infrastructure (communication between services).

- Encoding of Service Level Objectives (SLOs) and actions for defined events (e.g. violation of parameter thresholds) in the SLA information model.

- Work on perfSONAR schema and protocol extensions responsible for SLA data sharing. (The communication between perfSONAR components will have to be checked and eventually extended to make sure that SLA information can be exchanged).

- Work on management of possible SLA conflicts between domains. (Differences between SLAs in neighbouring domains, which may cause service degradation, should be identified and reported by perfSONAR. The system should manage them to minimise or eliminate negative effects).

- Management of SLA violations. (A series of actions conducted by the system when conditions that have been specified and agreed in the Service Level Agreement cannot be met).

The work on SLA Management in perfSONAR does not have to be based on a clean slate. Where appropriate, Task 3 is going to use existing ideas and experience to design the multi-domain infrastructure. One of the existing solutions that offers promising advantages is the Web Service Level Agreement (WSLA) framework proposed by IBM. This has mainly been designed for web services but its concept is general enough to cover many applications (in this case network management in a highly distributed environment). The WSLA framework consists of an information model that is based on an XML schema and a runtime architecture for performing SLA management. The first is used to encode and share a set of SLA information, the latter shows what components and communication is required between them. The distributed nature of the WSLA fits perfectly into the perfSONAR architecture and the information model may be the right choice for addressing all needs.

Another useful resource is the SLA Management Handbook produced by the TeleManagement Forum [TMF]. It provides recommendations for a general SLA management framework and defines SLA life cycles that can influence component breakdown and work flow. The use of TMF knowledge will help to find an open and flexible solution for SLA management in perfSONAR.

9.1 Using perfSONAR Measurement Data for SLAs

9.1.1 Active Monitoring Data

Active monitoring has been key in perfSONAR activities so far. Measurement Points and Archives for metrics at the IP layer or above have been implemented and deployed. As in most cases the work done so far was not SLA-oriented, refinements will need to be made to adhere to the SLA framework defined, e.g. define throughput measurements per SLO.

9.1.1.1 *Throughput*

As mentioned in *perfSONAR Tools* on page 5, perfSONAR provides throughput measurements via the MDM Bandwidth Controller (BWCTL) service with BWCTL MPs deployed in network PoPs. The introduction of SLAs into perfSONAR is expected to impose requirements for throughput measurements under certain traffic classification criteria as existing bulk throughput measurements between PoPs cannot be useful for SLA verification in the current format. This issue will be investigated and specified as part of JRA2 Task 3 work, following the definition of the SLA framework and SLSs.

9.1.1.2 *Delay and Delay Jitter*

Through HADES MAs, perfSONAR provides one-way delay and delay jitter measurement data between PoPs in NRENs and GEANT. This information is expected to be particularly useful for SLA verification at the IP layer, as these metrics are considered fundamental for the assessment of IP traffic quality.

9.1.2 Passive Monitoring Data

Instead of injecting artificial traffic into the network, in passive monitoring the quality of existing network traffic is analysed directly. Although more costly, passive monitoring can provide many useful network characteristics inherent to real network traffic, which cannot be obtained using other methods. Some of these metrics are expected to be particularly useful for SLAs defined over the perfSONAR infrastructure. At the IP layer the following are primarily to be dealt with:

- Network traffic classification.
- Packet loss.

9.1.2.1 *Network Traffic Classification*

Based on passive monitoring, information on what protocols, applications or applications classes the traffic mix comprises will be made available (including load dynamics and trends). This type of information can be exploited by SLAs for explicit treatment of specific traffic categories.

Advanced network traffic classification based on machine learning methods will be used (see *Passive Flow-Based Network Traffic Classification Using Machine Learning Techniques* on page 30 for more details).

A TC MP (Traffic Classification Measurement Point) will be implemented to provide network classification results within the perfSONAR infrastructure.

9.1.2.2 *Packet Loss*

Packet loss at the IP layer is a key network performance characteristic that significantly affects user experience. It is expected to be a key metric in SLAs defined for IP layer services, as it significantly affects the quality for protocols and applications over an IP network. Experience has shown that it is very difficult to measure realistic packet loss by test packets, due to the volume and dynamics of real traffic. Packet loss is a property inherent to traffic in which it is experienced and thus passive monitoring is necessary.

Using passive monitoring the following packet loss information can be made available:

- The number of lost packets and the packet loss rate for all pairs of network edge points where monitoring probes are deployed (typically G3 – NREN links) depending on the time period.
- The distribution of packet loss into major protocols.

- The detailed list of flows that experienced packet loss to assist in the resolution of performance problems.

The Packetloss application has been deployed as part of the pilot passive monitoring phase in GN2. A new version is planned within the GN3 activity SA2, Task T3.

New features will include a correlation of measurement results inside the application with possible losses detected in monitoring stations to increase result confidence as well as using standard Netflow records to make deployment easier (as opposed to extended flow records generated using packet capture cards).

The results will be stored in MySQL and RRD databases, made available from the Packetloss MPs of the perfSONAR infrastructure.

10 New Features and Maintenance of OGF NM-WG and OGF NMC-WG

The core of the perfSONAR system is its schema (an XML structure for monitoring network information and relations between information objects) and protocol (a set of rules that define how perfSONAR components - services and client applications - communicate). Most of the work on the perfSONAR system was already carried out during the GN2 project and within the OGF. This work centred around basic schema structures for dealing with few well-known IP metrics, and communication between services to accomplish IP monitoring functionality. In GN3, Task 3 of JRA 2 is going to provide functional extensions to perfSONAR, which will require new definitions in the schema and the protocol. Additionally, some changes may have to be considered to improve already existing solutions.

The perfSONAR architecture is open and flexible, so that new monitoring features can be added as new requirements arise from users. In some cases the addition of new elements to perfSONAR requires its schema and protocol to be extended. This is possible as this requirement was already foreseen at the beginning of the standards' design phase.

The JRA2 Task 3 group is actively involved in the work on OGF Network Measurements Working Group (NM-WG) and OGF Network Measurement and Control Working Group (NMC-WG) standards to improve the schema and the protocol. Since these are integral parts of the perfSONAR system, task contributors, who have already been involved in the topic during GN2, decided to assign some effort to the cooperation with US partners (Internet2, ESnet, University of Delaware and other US institutions involved in the perfSONAR project). The general goals of this cooperation are to:

- Control standards development.
- Improve the current form.
- Add any new elements that new features require.
- Support external institutions and users in the adoption process.
- Finalise the standardisation process.

The Task 3 team has also identified specific actions that this cooperation entails:

- Standardisation of bulk-data transfer in the perfSONAR infrastructure.

  So far perfSONAR is not well prepared to transfer high volumes of data between services in a reasonable amount of time in a standardised way. Although some initiatives and ideas were introduced and tested during GN2, no standardised and widely accepted solution was achieved. Based on the experiences made, T3 will investigate the problem and work on a final solution.

- Specification of data format transformation functionality in the perfSONAR infrastructure.

  The initial concept of the Transformation Service (TrS) in the perfSONAR system, which is responsible for data transformation, was already provided during GN2. The functionality of such a service would be very useful in cases where a service or a client application needs some data in a specific format that is different from the one available.

- Supporting Task 3 of the SA2 activity (Multi-Domain Network Services) in creating new schema and protocol elements needed for new functionalities.

  JRA2 T3 will actively support SA2 T3 where perfSONAR services that are maintained by SA2 T3 require schema or protocol changes.

11 Conclusions

JRA2 Task 3 are investigating and implementing a number of new ideas and extensions to enhance the multi-domain network monitoring functionality provided by the perfSONAR system. This will result in a new prototype that will:

- Offer improved multi-domain network monitoring.
- Provide new functionality that meets emerging demands.
- Deliver a stable and robust solution.

The completed work conducted by JRA 2 Task 3 will be passed on to SA2 (Multi-Domain Network Services) Task 3 (Service Monitoring Tools and Performance) for assessment and further development in a production environment.

Appendix A NETCONF in perfSONAR

A.1 Separating perfSONAR Information and Communication Models with NM-WG

An example of this is the mechanism perfSONAR uses to retrieve a list of interfaces from an RRD MA. There are no formal definitions in the NM-WG schemas that specify how to retrieve a list of interfaces, but a de facto method has been defined by the implementations. The method is to query the MA for a specific metric but instead of specifying which interface data is wanted for, the interface tag is left empty:

```
<nmwg:message id="1250065054" type="MetadataKeyRequest"
xmlns:netutil="http://ggf.org/ns/nmwg/characteristic/utilization/2.0/"
xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"
xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"
xmlns:select="http://ggf.org/ns/nmwg/ops/select/2.0/">
  <nmwg:metadata id="meta1">
    <netutil:subject id="iusub1">
      <nmwgt:interface/>
    </netutil:subject>

<nmwg:eventType>http://ggf.org/ns/nmwg/characteristic/utilization/2.0</nmwg:eve
ntType>
  </nmwg:metadata>
  <nmwg:data id="data_1" metadataIdRef="meta1"/>
</nmwg:message>
```

In this message utilisation metrics are requested. The response is quite long returning both information about the available interfaces and utilisation data for each interface. A small subset of the reply is shown below:

```
<nmwg:message id="1250065054_resp" messageIdRef="1250065054"
  type="MetadataKeyResponse" xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/">
  <nmwg:metadata id="meta0">
    <netutil:subject id="subj0"
xmlns:netutil="http://ggf.org/ns/nmwg/characteristic/utilization/2.0/">
      <nmwgt:interface xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/">
```

```
          <nmwgt:hostName>rt1.ams.nl.geant2.net</nmwgt:hostName>
          <nmwgt:ifAddress type="ipv4"/>
          <nmwgt:ifName>e3-0/0/2</nmwgt:ifName>
          <nmwgt:ifDescription/>
          <nmwgt:ifIndex>25</nmwgt:ifIndex>
          <nmwgt:direction>in</nmwgt:direction>
          <nmwgt:capacity>9953000000</nmwgt:capacity>
        </nmwgt:interface>
      </netutil:subject>

  <nmwg:eventType>http://ggf.org/ns/nmwg/characteristic/utilization/2.0</nmwg:eve
  ntType>
      <nmwg:parameters>
        <nmwg:parameter name="keyword">project:MDM</nmwg:parameter>
      </nmwg:parameters>
    </nmwg:metadata>
```

The response shows that all information about the interface is returned. As NM-WG only specifies more or less fixed messages and does not define an information model, it is impossible to request a subset of the information. It is, for example, not possible to request a simple list of names for the available interfaces.

A.2 MA Information Model using NETCONF

This section provides a detailed description of the information model for the MA. The general outline is shown in Figure A.11.1. Some attributes have been left out to simplify the description.

```
measurementArchive
|-sysinfo
|  |-name
|  |-administrator
|-datasources
    source
    |-name
    |-type
    |-selftest
    |  |-name
    |  |-description
    |  |-status
    |  |-details
    |-obspoints
    |  |-obspoint
    |-groups
    |  |-group
    |-template
    |-timeperiodinfo
    |-timeperiods
      |-timeperiod
        |-id
        |-starttime
        |-duration
        |-reports
          |-report
            |-id
            |-obspoint
            |-transformation
            |-view
            |-sort
            |-limit
            |-data
```

Figure A.11.1: General outline of the MA information model.

The following attributes are defined:

- sysinfo - General information about the MA and its operational status.
- name – The name of the MA.
- administrator – The name, email address etc. of the MA's administrator.
- datasources - A single MA can have multiple data sources (for example, NetFlow data, SNMP, Multicast statistics etc.). This is a list of the data sources that are available
- source - Information about a single data source.
- name – The name of the data source.
- type - The data source type (for example, postgressql, RRD, etc.).
- selftest – A list of self tests that report the MA's operational status.
- name – The name of the test.
- description – A detailed description of the test.
- status – The status of the test (passed or failed).
- details - Additional test details. If the test failed, this includes an error description.

- obspoints – A list of available observation points.
- groups – A list of available groups of observation points.
- template – The report template describing the available reports (for more details, see *Report Template* on page 75).
- timeperiodinfo - General information about available time intervals like name, number of seconds, how to navigate to lower/hight interval etc.
- timeperiods – A list of available time periods that contains actual data.
- timeperiod - Information about a single time period.
- id – A unique id of the time period.
- starttime – The start time of the time period.
- duration – The duration of time period in seconds.
- reports – A list of reports with available data for this time period.
- report - A single report with data for this time period.
- id – The ID of the report.
- obspoint – The observation point that this report contains data for.
- transformation – The transformation of the report (for more details, see *Report Template* on page 75)
- view – The view of the report (for more details see *Report Template* on page 75).
- sort - The column by which the report data is sorted.
- limit – The maximum number of data rows.
- data – The data for this report.

The text below shows a small part of the YANG MA model:

```
container selftests {
 list test {
  key "name";
   leaf name {
    mandatory "true";
    type string;
    description "Name of test";
   }
   leaf description {
    mandatory "true";
    type string;
    description "Description of test";
   }
   leaf status {
    mandatory "true";
    type enumeration {
     enum "passed";
     enum "failed";
    }
```

```
       description "Status of test. Can be passed or failed";
      }
      leaf details {
        mandatory "false";
        type string;
        description "Human readable details about the test. If a test fails, the
   reason should be described here";
       }
      }
```

This YANG text defines the self test capability of the MA. It specifies a YANG container that contains a list of tests. Each test has a name, a description, a status and an optional detailed description. The name of the test is defined as the key of the list, which means that the name has to be unique. The name, description and optional details are all text while the status can only have the value passed or failed.

Such short text that is relatively easy to read and understand is all that is needed by a network operator to properly query a MA to retrieve the list of self tests.

A.2.1 Report Template

The report template in the MA information model is currently taken directly from the Stager application without any modifications. The Stager template model is a good starting point, but it contains many advanced features that are application-specific and not necessarily needed in a template system for an MA or MP.

In Stager a report can have different transformations and views. A transformation specifies how data should be presented (for example, if a source/destination report should be presented as a table with one column for source and one for destination, or as a matrix with source in the first row and destination in the first column). A view specifies which data elements should be shown in the transformation, and each transformation can have multiple views (for example, for a Src/Dst IP Netflow report, a matrix report can have different views: one for octets, one for packets and one for flows).

Many reports often share the same data types, for example most Netflow reports displays information about octets, packets and flows and many reports display IP addresses. The template system therefore specifies all awailable data types in one place and then each report references this definition. This makes it easy to make changes on how a specific data type is displayed by the user frontend.

The general outline of a report template is shown in Figure A.1.2.

```
reports
|-dataType
|  |-name
|  |-type
|-headerDef
|  |-id
|  |-topHeader
|     |-name
|     |-subHeader
|        |-name
|        |-data
|-transformationDef
|  |-id
|  |-type
|  |-view
|     |-name
|     |-headerRef
|-report
   |-id
   |-name
   |-descr
   |-transformationRef
```

Figure A.1.2: General outline of the report template.

The following attributes are shown:

- dataType - A list of data types that are used in the various reports.
- name – The name of the data type.
- type – The type of data (for example, octets, temperature, percentage).
- headerDef - Reports often have a common set of data types and the same headers. headerDef makes it possible to define common header groups that can be reused in multiple reports.
- id – The ID of the header definition.
- topHeader - Stager only supports two layers of headers and this specifies the top header.
- name – The name of the top header.
- subHeader - One or more sub headers belonging to the top header.
- name – The name of the sub header.
- data – References the data type specified by the dataType tag and specifies what data should be displayed under this header.
- transformationDef - Some reports have entire transformations in common. These are specified here.
- id – The ID of the transformation definition.
- type – The type of transformation (table, matrix, overview or global).
- view - A view for this transformation. Can be multiple instances.

- name – The name of the view.
- headerRef - References a headerDef. It is also possible to specify headers directly.
- report - A report.
- id – The unique ID of the report.
- name – The name of the report.
- descr – A description of the report.
- transformationRef - References a transformationDef. Can also specify transformations directly.

A.3 Implementation of NETCONF in perfSONAR

Since the prototype uses the Stager template system directly without any modifications, it was relatively easy to turn Stager into both a Measurement Archive and a front-end for a Measurement Archive. The Stager architecture is modular, so the main change was to create a new class that could retrieve information using NETCONF instead of SQL calls. And instead of displaying reports in a web page, the data is converted into NETCONF XML messages. Less than 1000 new lines of code were needed to implement the prototype.

In the prototype all request messages are formally validated using the standardised NETCONF WSDL file, but so far replies are sent back without any validation. Recommendations on how to validate NETCONF reply messages are being worked on by IETF [YANG].

The implementation is done in PHP and the code needed to do a query is relatively short. An example of a query that retrieves the list of self tests is shown below:

```
$msg=array("message-id"=>1,
           "getconfig"=>array('source'=>array('running',""),
                        'filter'=>array('type'=>'xpath',
'select'=>"/measurementArchive/datasources/source[name='ssmping']/selftests/tes
t")));
$return = (array)$client->rpc($msg);
```

In this example, an array is defined that contains the message-id and a getconfig tag specifying that information should be retrieved. The getconfig tag contains another array that specifies that information should be retrieved from the running source and an XPath filter is used to specify that all self tests for the data source ssmping should be retrieved. This array is almost identical for all queries to the MA. The only difference is the message ID and the actual XPath expression that specifies the information to be retrieved.

The second line uses the rpc command defined by the NETCONF WSDL file [RFC4743] to retrieve the requested information. The code does not care about the actual XML message and can easily be converted into a function or stand-alone utility that can be used by network operators to create simple scripts, similar to getsnmp as described earlier.

The actual XML message sent over the network for the above example would be:

```
   <ns1:rpc message-id="1">
     <ns1:getconfig>
       <ns1:source>
         <ns1:running/>
       </ns1:source>
       <ns1:filter type="xpath" select=
   "/measurementArchive/datasources/source[name='ssmping']/selftests/test"/>
     </ns1:getconfig>
   </ns1:rpc>
```

The result sent back from the MA is a simple list of all tests:

```
   <ns1:rpc-reply message-id="1">
         <ns1:data>
           <measurementArchive xmlns="http://stager.uninett.no/stagerMA">
             <test>
               <name>dbConnect</name>
               <description>
                    Test database connection
               </description>
               <status>passed</status>
               <details/>
             </test>
             <test>
               <name>timeperiod</name>
               <description>
                    Test to see if any time periods exists in the database
               </description>
               <status>failed</status>
               <details>
                    No timeperiods exists in the database
               </details>
             </test>
           </measurementArchive>
         </ns1:data>
       </ns1:rpc-reply>
```

This response can be processed as XML by the client or, as in the example above, it can be automatically converted into an associative array. In the latter case all XML processing is handled by the SOAP library. To find out the structure of the associative array, a developer only has to read the YANG information model for the MA. This provides a lot of flexibility and if someone is just interested in the names of available self tests they could use the following query:

```
/measurementArchive/datasources/source[name='ssmping']/selftests/test/name
```

Another example is to only retrieve self tests that failed:

```
/measurementArchive/datasources/source[name='ssmping']/selftests/test[status='f
ailed']
```

To show that the information model for the MA could easily be implemented by other applications than Stager, a simple implementation of an MA based on RRD files was carried out. The implementation consisted of just a few hundred lines of code and, while primitive, it provided the basic functionality of providing a list of available observation points, and retrieving statistics for a single observation point and time period. Since this implementation uses exactly the same information model as the Stager MA, the Stager user interface can also display the data from the RRD MA without any code modifications.

Appendix B Performance of Software Libraries for Parallelisation

To choose the most convenient library to use with the FBA algorithm, several machine learning algorithms were tested (KNN, K-means and DBSCAN algorithms). The Kmeans Bitmap algorithm was used to compare threading libraries and sequential code with CUDA. CUDA was found to be the best performing algorithm, although OpenMP also performed well in the tests. The PThreads library is harder to use and, because it requires considerable threading specific code, most actions like load balancing, handling race conditions, etc. have to be performed manually. OpenMP, however, can carry out most of these actions automatically or by adding simple #pragma statements. OpenMP is a portable library and can work with any of the modern operating systems. But PThreads can only work with POSIX-compliant and *NIX based OSs.

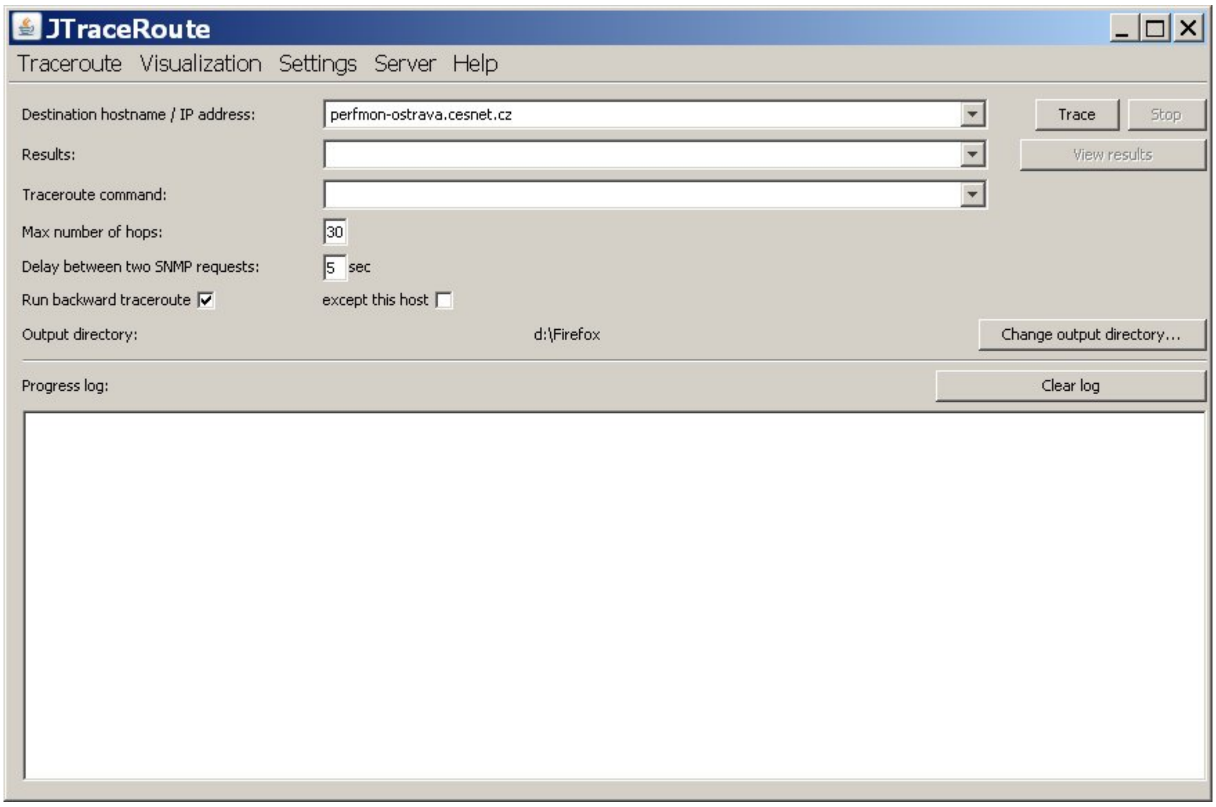
The tests results show that the most reliable parallel processing libraries for implementation of traffic classification tools are CUDA for GPU parallelisation and OpenMP for CPU parallelisation. Therefore, the software implementation of traffic classifier will support both of those libraries, and it will be able to run in either GPU mode or CPU mode.

Appendix C Example: Jtraceroute Use

The following scenario illustrates a typical use of Jtraceroute. When a user starts Jtraceroute, the initial dialog is displayed (see Figure B.1.3). The user can enter the IP address or hostname of the remote host and click Trace to start the traceroute.

Optional settings that can be specified include:

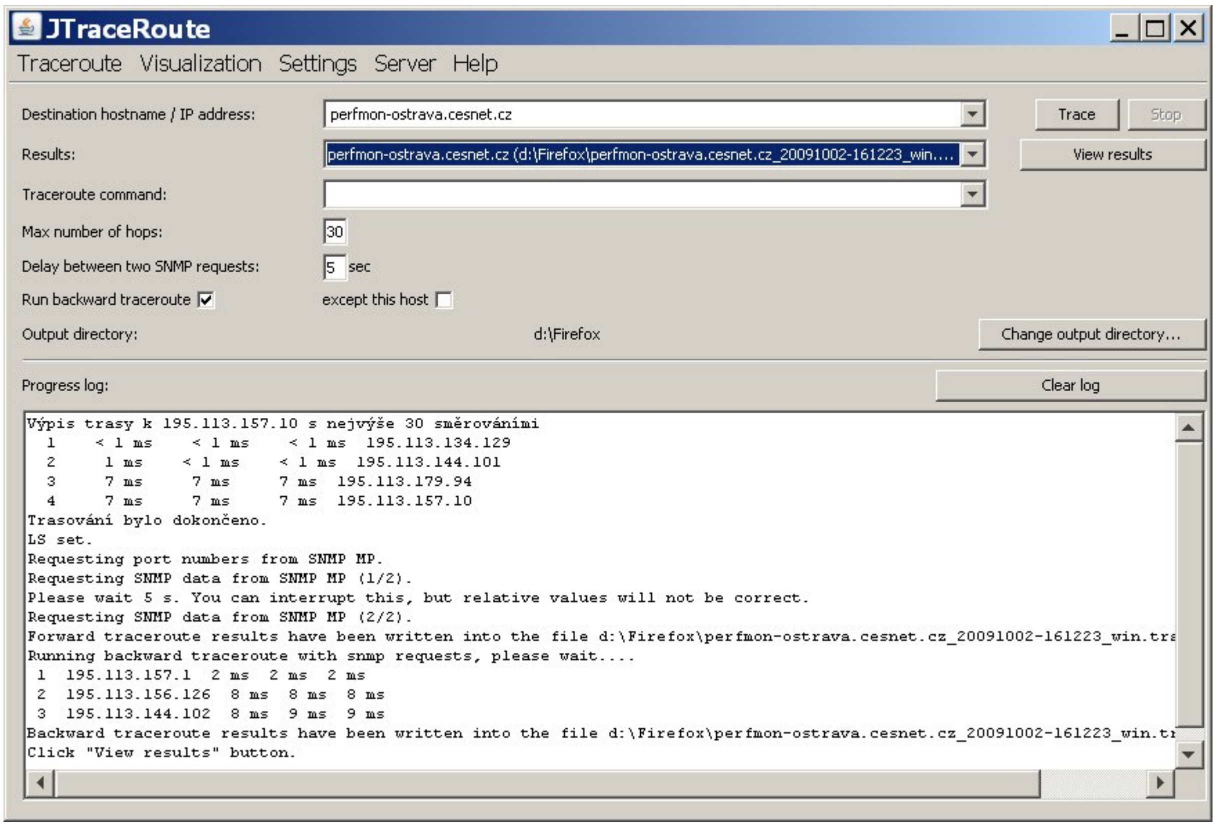
- Whether the backward traceroute should be done
- The maximum number of hops to wait for
- The delay between two requests for performance characteristics.
- The directory to store XML files with results.
- Whether to remove the local host from the backward traceroute (that is not to wait for its response), This option is needed because many Windows installations do not permit traceroute responses by default.
- Whether an alternative command should be used to perform the traceroute (if this is not specified the default traceroute command in the local operating system is used).


JTraceRoute
Traceroute Visualization Settings Server Help
Trace
Stop
View results
Change output directory...
Clear log

Destination hostname / IP address: perfmon-ostrava.cesnet.cz
Results:
Traceroute command:
Max number of hops: 30
Delay between two SNMP requests: 5 sec
Run backward traceroute ✓ except this host ☐
Output directory: d:\Firefox
Progress log:

Figure B.1.3: Jtraceroute initial window.

Figure B.1.4 shows an example of what the main dialog looks like if forward and backward traceroutes are executed and router performance characteristics are retrieved through SNMP MPs.



Destination hostname / IP address: perfmon-ostrava.cesnet.cz

Results: perfmon-ostrava.cesnet.cz (d:\Firefox\perfmon-ostrava.cesnet.cz_20091002-161223_win....

Traceroute command:

Max number of hops: 30

Delay between two SNMP requests: 5 sec

Run backward traceroute ✓ except this host ☐

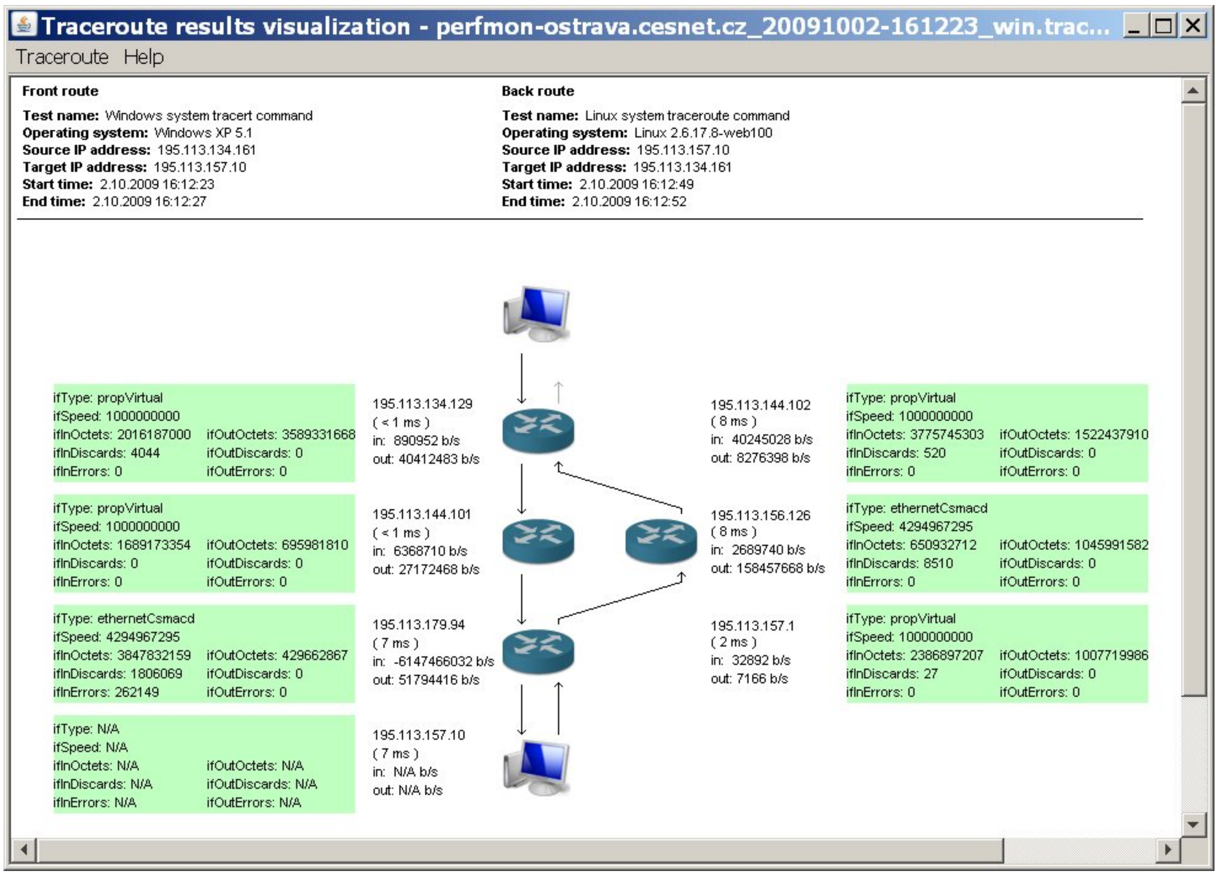
Output directory: d:\Firefox

Progress log:

```
Výpis trasy k 195.113.157.10 s nejvýše 30 směrováními
  1    < 1 ms    < 1 ms    < 1 ms  195.113.134.129
  2     1 ms    < 1 ms    < 1 ms  195.113.144.101
  3     7 ms     7 ms     7 ms  195.113.179.94
  4     7 ms     7 ms     7 ms  195.113.157.10
Trasování bylo dokončeno.
LS set.
Requesting port numbers from SNMP MP.
Requesting SNMP data from SNMP MP (1/2).
Please wait 5 s. You can interrupt this, but relative values will not be correct.
Requesting SNMP data from SNMP MP (2/2).
Forward traceroute results have been written into the file d:\Firefox\perfmon-ostrava.cesnet.cz_20091002-161223_win.tra
Running backward traceroute with snmp requests, please wait....
 1  195.113.157.1  2 ms  2 ms  2 ms
 2  195.113.156.126  8 ms  8 ms  8 ms
 3  195.113.144.102  8 ms  9 ms  9 ms
Backward traceroute results have been written into the file d:\Firefox\perfmon-ostrava.cesnet.cz_20091002-161223_win.tr
Click "View results" button.
```

Figure B.1.4: Jtraceroute window after traceroute and retrieval of performance characteristics.

If the user clicks View results, a graphical presentation is displayed (see Figure B.1.5 for an example). Just as in the standard traceroute, for each forward and backward router interface, the IP address and RTT is listed. However, the current interface utilisation (in bits / second) and values of several other counters in the router MIB are also displayed.


Traceroute results visualization - perfmon-ostrava.cesnet.cz_20091002-161223_win.trac...
Traceroute Help

Front route

Test name: Windows system tracert command
Operating system: Windows XP 5.1
Source IP address: 195.113.134.161
Target IP address: 195.113.157.10
Start time: 2.10.2009 16:12:23
End time: 2.10.2009 16:12:27

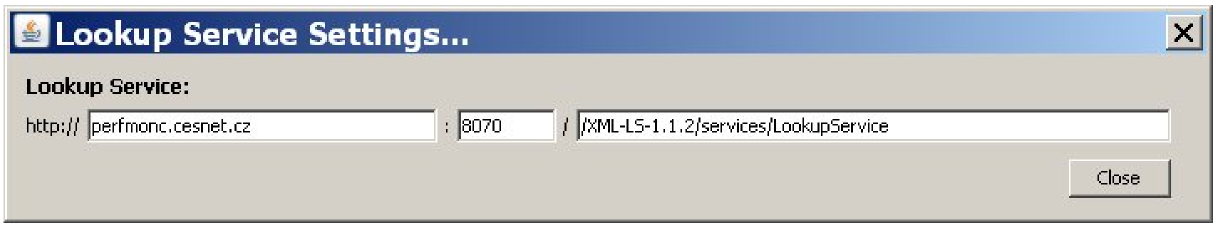
Back route

Test name: Linux system traceroute command
Operating system: Linux 2.6.17.8-web100
Source IP address: 195.113.157.10
Target IP address: 195.113.134.161
Start time: 2.10.2009 16:12:49
End time: 2.10.2009 16:12:52

```
ifType: propVirtual
ifSpeed: 1000000000
ifInOctets: 2016187000    ifOutOctets: 3589331668
ifInDiscards: 4044        ifOutDiscards: 0
ifInErrors: 0             ifOutErrors: 0

195.113.134.129
( < 1 ms )
in:  890952 b/s
out: 40412483 b/s

ifType: propVirtual
ifSpeed: 1000000000
ifInOctets: 1689173354    ifOutOctets: 695981810
ifInDiscards: 0           ifOutDiscards: 0
ifInErrors: 0             ifOutErrors: 0

195.113.144.101
( < 1 ms )
in:  6368710 b/s
out: 27172468 b/s

ifType: ethernetCsmacd
ifSpeed: 4294967295
ifInOctets: 3847832159    ifOutOctets: 429662867
ifInDiscards: 1806069     ifOutDiscards: 0
ifInErrors: 262149        ifOutErrors: 0

195.113.179.94
( 7 ms )
in:  -6147466032 b/s
out: 51794416 b/s

ifType: N/A
ifSpeed: N/A
ifInOctets: N/A           ifOutOctets: N/A
ifInDiscards: N/A         ifOutDiscards: N/A
ifInErrors: N/A           ifOutErrors: N/A

195.113.157.10
( 7 ms )
in:  N/A b/s
out: N/A b/s

195.113.144.102
( 8 ms )
in:  40245028 b/s
out: 8276398 b/s

ifType: propVirtual
ifSpeed: 1000000000
ifInOctets: 3775745303    ifOutOctets: 1522437910
ifInDiscards: 520         ifOutDiscards: 0
ifInErrors: 0             ifOutErrors: 0

195.113.156.126
( 8 ms )
in:  2689740 b/s
out: 158457668 b/s

ifType: ethernetCsmacd
ifSpeed: 4294967295
ifInOctets: 650932712     ifOutOctets: 1045991582
ifInDiscards: 8510        ifOutDiscards: 0
ifInErrors: 0             ifOutErrors: 0

195.113.157.1
( 2 ms )
in:  32892 b/s
out: 7166 b/s

ifType: propVirtual
ifSpeed: 1000000000
ifInOctets: 2386897207    ifOutOctets: 1007719986
ifInDiscards: 27          ifOutDiscards: 0
ifInErrors: 0             ifOutErrors: 0
```

Figure B.1.5: Graphical presentation of Jtraceroute results.

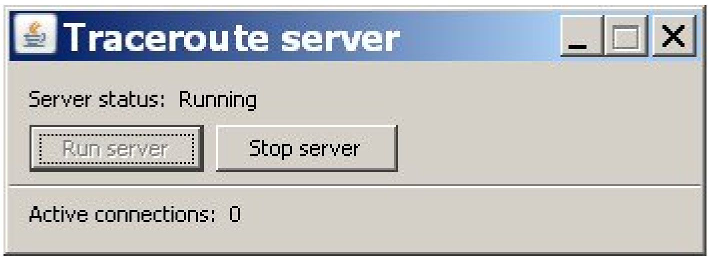
The URL of the LS is preconfigured in Jtraceroute and can be changed by the user using the Lookup Service Settings dialog (see Figure B.1.6). The traceroute server runs by default in Jtraceroute (to respond to requests for backward traceroute from the other end) and can be stopped by the user using the Traceroute server dialog (see Figure B.1.7).



Lookup Service:

http:// perfmonc.cesnet.cz : 8070 / /XML-LS-1.1.2/services/LookupService

Figure B.1.6: Setting the URL of the LS.



Server status: Running

Active connections: 0

Figure B.1.7: Starting or stopping built-in traceroute server.

The following is a fragment of an XML file that includes stored results. You can see various counter values from the router MIB:

```
   <ResultsProbe>
                <Index>11</Index>
                <HopIndex>4</HopIndex>
                <IndexPerHop>3</IndexPerHop>
                <HopAddrType>
                    <probeHopAddrType>ipv4</probeHopAddrType>
                </HopAddrType>
                <HopAddr>
                    <probeHopAddr>
                        <inetAddressIpv4>192.168.1.2</inetAddressIpv4>
                    </probeHopAddr>
                </HopAddr>
                <RoundTripTime>
                    <probeRoundTripTime>7</probeRoundTripTime>
                </RoundTripTime>
                <ResponseStatus>responseReceived</ResponseStatus>
                <Time>2009-10-02T16:12:27</Time>
            </ResultsProbe>
            <ResultEndDateAndTime>
                <dateAndTime>2009-10-02T16:12:27</dateAndTime>
            </ResultEndDateAndTime>
            <DataHop>
                <HopIndex>1</HopIndex>
                <IndexPerHop>1</IndexPerHop>
                <HopAddr>
                    <probeHopAddr>
                        <inetAddressIpv4>195.113.134.129</inetAddressIpv4>
                        <portNum>174</portNum>
                    </probeHopAddr>
                </HopAddr>
                <data name="IF-MIB::ifType.174">propVirtual</data>
                <data name="IF-MIB::ifSpeed.174">1000000000</data>
```

```
        <data name="IF-MIB::ifInOctets.174">2016187000</data>
        <data name="IF-MIB::ifOutOctets.174">3589331668</data>
        <data name="IF-MIB::ifInDiscards.174">4044</data>
        <data name="IF-MIB::ifOutDiscards.174">0</data>
        <data name="IF-MIB::ifInErrors.174">0</data>
        <data name="IF-MIB::ifOutErrors.174">0</data>
    </DataHop>
```

Appendix D Example: Commercial Network Management Tools with Events Handling Functionality

One of the most popular and commonly used software for WAN network management is Open View – Network Node Manager (OV-NNM), a product from Hewlett Packard. It provides the following features:

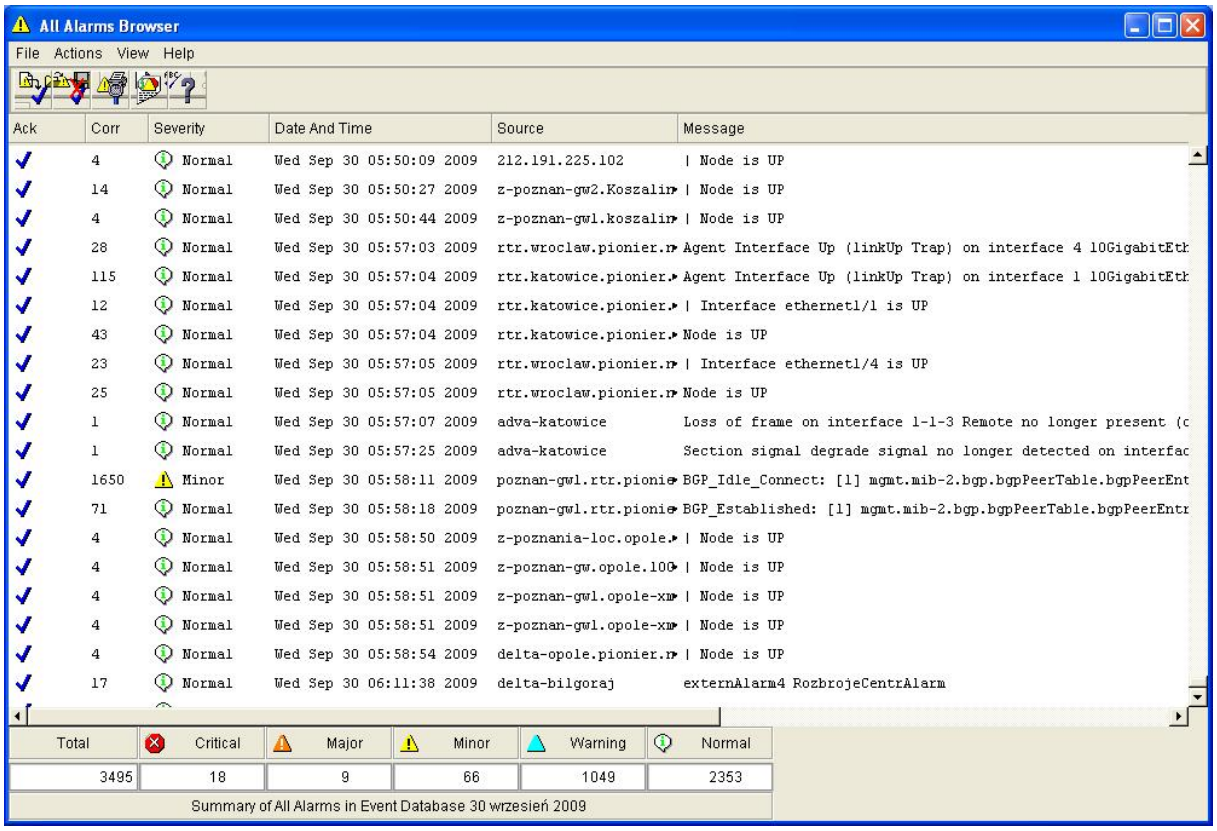
- Broad multi-vendor device coverage.
- Dynamic root cause analysis and diagnostics (e.g. path views and service impact analyses).
- Configurable event correlation.
- Performance thresholding and reporting [SMAPL].

The OV-NNM's events handling and filtering helps operators to identify problems in the monitored network. The HP Network Node Manager Advanced Edition filters and correlates network events, so that only the most important messages are presented to the operator. This allows them to concentrate on the actual problems and find their source [HPNNMAE]. Additionally, HP Network Node Manager Advanced Edition includes the Active Problem Analyzer, a new multi-threaded poller and analysis subsystem. This combines the output of the event-based root-cause correlation with an understanding of physical topology, and augments it as necessary with additional information gathered through targeted polling and data collection in order to determine the root cause of many common network problems.

To handle all received network events NNM offers a set of out-of-the-box correlators, which include the following:

- Event classifier—uniquely classifies and consolidates all Cisco events.
- PairWise events—matches parent and child events.
- Chassis failure—monitors Cisco devices traps for temperature, fan failure and power supply faults.
- Router/switch health—correlates interface status alarms with related router or switch node status alarms.
- Multiple reboot—monitors coldStart and warmStart traps.
- De-duplication—nests duplicate events under the most recent alarm.
- Connector down— using path analysis to zero in on the faulty device when connectivity is lost.
- Scheduled maintenance—excludes events from maintenance activities.

Figure B.1.8 shows an HP-OV NNM Alarms Browser.



Ack	Corr	Severity	Date And Time	Source	Message
✓	4	Normal	Wed Sep 30 05:50:09 2009	212.191.225.102	| Node is UP
✓	14	Normal	Wed Sep 30 05:50:27 2009	z-poznan-gw2.Koszalin	| Node is UP
✓	4	Normal	Wed Sep 30 05:50:44 2009	z-poznan-gw1.koszalin	| Node is UP
✓	28	Normal	Wed Sep 30 05:57:03 2009	rtr.wroclaw.pionier.n	Agent Interface Up (linkUp Trap) on interface 4 10GigabitEth
✓	115	Normal	Wed Sep 30 05:57:04 2009	rtr.katowice.pionier.	Agent Interface Up (linkUp Trap) on interface 1 10GigabitEth
✓	12	Normal	Wed Sep 30 05:57:04 2009	rtr.katowice.pionier.	| Interface ethernet1/1 is UP
✓	43	Normal	Wed Sep 30 05:57:04 2009	rtr.katowice.pionier.	Node is UP
✓	23	Normal	Wed Sep 30 05:57:05 2009	rtr.wroclaw.pionier.n	| Interface ethernet1/4 is UP
✓	25	Normal	Wed Sep 30 05:57:05 2009	rtr.wroclaw.pionier.n	Node is UP
✓	1	Normal	Wed Sep 30 05:57:07 2009	adva-katowice	Loss of frame on interface 1-1-3 Remote no longer present (c
✓	1	Normal	Wed Sep 30 05:57:25 2009	adva-katowice	Section signal degrade signal no longer detected on interfac
✓	1650	Minor	Wed Sep 30 05:58:11 2009	poznan-gw1.rtr.pionie	BGP_Idle_Connect: [1] mgmt.mib-2.bgp.bgpPeerTable.bgpPeerEnt
✓	71	Normal	Wed Sep 30 05:58:18 2009	poznan-gw1.rtr.pionie	BGP_Established: [1] mgmt.mib-2.bgp.bgpPeerTable.bgpPeerEntr
✓	4	Normal	Wed Sep 30 05:58:50 2009	z-poznania-loc.opole.	| Node is UP
✓	4	Normal	Wed Sep 30 05:58:51 2009	z-poznan-gw.opole.100	| Node is UP
✓	4	Normal	Wed Sep 30 05:58:51 2009	z-poznan-gw1.opole-xm	| Node is UP
✓	4	Normal	Wed Sep 30 05:58:51 2009	z-poznan-gw1.opole-xm	| Node is UP
✓	4	Normal	Wed Sep 30 05:58:54 2009	delta-opole.pionier.n	| Node is UP
✓	17	Normal	Wed Sep 30 06:11:38 2009	delta-bilgoraj	externAlarm4 RozbrojeCentrAlarm

Total	Critical	Major	Minor	Warning	Normal
3495	18	9	66	1049	2353

Summary of All Alarms in Event Database 30 wrzesień 2009

Figure B.1.8: An example view of an OV-NNM Alarms Browser window.

Another popular and commonly used network management platform is the IBM Tivoli Netcool suite. For network and devices management it includes, for example, such modules as OMNIbus, Webtop, Reporter and Network Manager (formerly Precision IP/TN). The IBM Tivoli Netcool/OMNIbus features are:

- Automated event correlation, network problem isolation and resolution capabilities.
- Consolidates data into Web dashboard views with customisable event and service displays.
- Uses lightweight agents to collect events from more than 1,000 sources in real time
- Provides integration with the IBM Tivoli Monitoring family to measure performance of business applications and to monitor itself—generating alarms based on user-defined thresholds
- Offers operations management software that integrates with the broader Tivoli portfolio for a single view of operations, including cross-domain correlation, and common visualisation, navigation, security and reporting and launch-in-context capabilities. [IBMTiv]

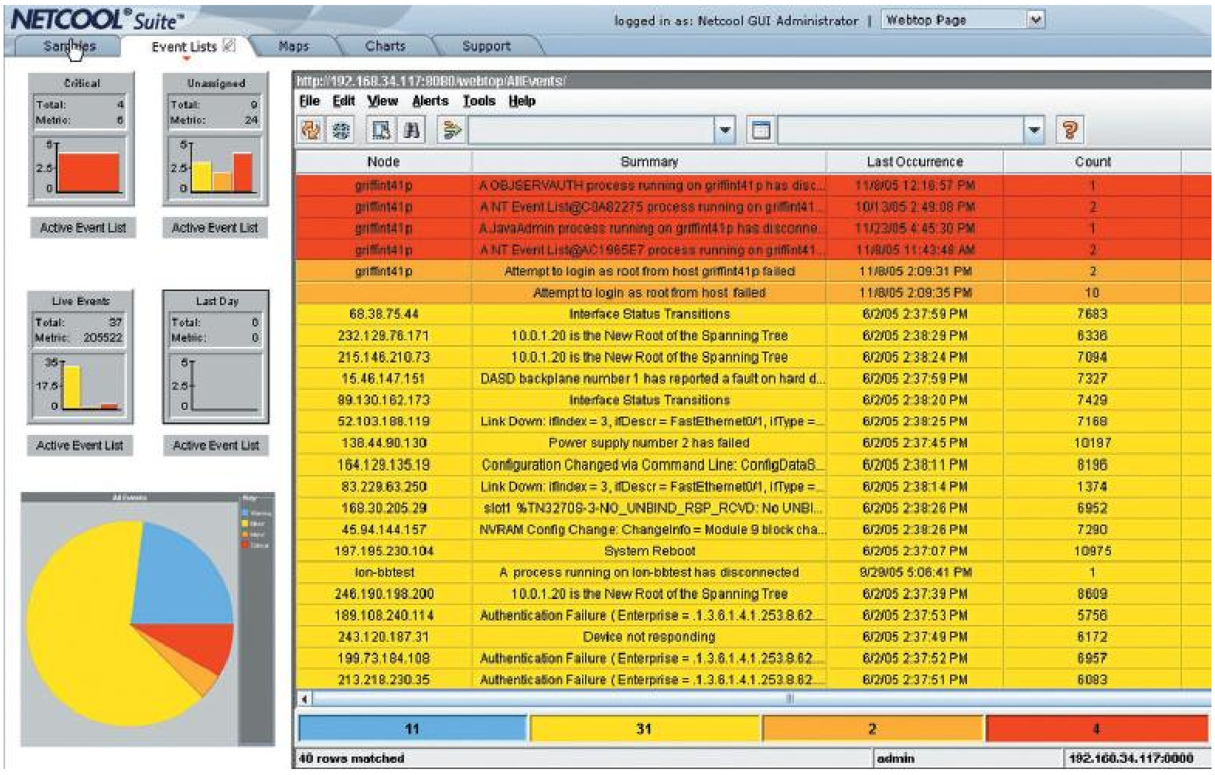
Tivoli Netcool/Webtop “delivers graphical maps, tables and event lists to the remote operator via HTML and Java. […] [It also] provides a Web-[…] interface that enables […] viewing of […] management data from Netcool/OMNIbus [and it] gives operations staff […] access to service status and […] [other] information […] [via] any Java-enabled Web browser” [NetWeb].

Tivoli Netcool/Reporter “provides […] [users] with long-term, retrospective information about the behavior of devices, links, and services within their networks. [It] captures, analyzes, and presents data generated over time into […] [viewable] reports ” [NetRep].

IBM Tivoli Network Manager (formerly Netcool/Precision) provides “real-time network discovery, topology visualisation and root cause analysis for layer 1, 2 and 3 networks, including IP, Ethernet, and multiprotocol label switching (MPLS)” [TNM].

The main application from the aforementioned modules which performs the events handling functionality is the Netcool/OMNIbus. After collecting events from the monitored infrastructure in real time, the Netcool/OMNIbus “eliminates duplicate events and filters […] [them to enable operators to see] the most critical problems and […] [also partially] automate the isolation and resolution of those problems” [NetOM].

Figure B.1.9 shows an IBM Tivoli Netcool event list.



Critical	
Total:	4
Metric:	6

Unassigned	
Total:	9
Metric:	24

Live Events	
Total:	37
Metric:	205522

Last Day	
Total:	0
Metric:	0

Node	Summary	Last Occurrence	Count
griffint41p	A OBJSERVAUTH process running on griffint41p has disc...	11/8/05 12:16:57 PM	1
griffint41p	A NT Event List@C0A82275 process running on griffint41...	10/13/05 2:49:08 PM	2
griffint41p	A JavaAdmin process running on griffint41p has disconne...	11/23/05 4:45:30 PM	1
griffint41p	A NT Event List@AC1965E7 process running on griffint41...	11/8/05 11:43:48 AM	2
griffint41p	Attempt to login as root from host griffint41p failed	11/8/05 2:09:31 PM	2
	Attempt to login as root from host failed	11/8/05 2:09:35 PM	10
68.38.75.44	Interface Status Transitions	6/2/05 2:37:59 PM	7683
232.129.76.171	10.0.1.20 is the New Root of the Spanning Tree	6/2/05 2:38:29 PM	6336
215.146.210.73	10.0.1.20 is the New Root of the Spanning Tree	6/2/05 2:38:24 PM	7094
15.46.147.151	DASD backplane number 1 has reported a fault on hard d...	6/2/05 2:37:59 PM	7327
89.130.162.173	Interface Status Transitions	6/2/05 2:38:20 PM	7429
52.103.188.119	Link Down: ifIndex = 3, ifDescr = FastEthernet0/1, ifType =...	6/2/05 2:38:25 PM	7168
138.44.90.130	Power supply number 2 has failed	6/2/05 2:37:45 PM	10197
164.129.135.19	Configuration Changed via Command Line: ConfigDataS...	6/2/05 2:38:11 PM	8196
83.229.63.250	Link Down: ifIndex = 3, ifDescr = FastEthernet0/1, ifType =...	6/2/05 2:38:14 PM	1374
168.30.205.29	slot1 %TN3270S-3-NO_UNBIND_RSP_RCVD: No UNBI...	6/2/05 2:38:26 PM	6952
45.94.144.157	NVRAM Config Change: ChangeInfo = Module 9 block cha...	6/2/05 2:38:26 PM	7290
197.195.230.104	System Reboot	6/2/05 2:37:07 PM	10975
lon-bbtest	A process running on lon-bbtest has disconnected	9/29/05 5:06:41 PM	1
246.190.198.200	10.0.1.20 is the New Root of the Spanning Tree	6/2/05 2:37:39 PM	8609
189.108.240.114	Authentication Failure (Enterprise = .1.3.6.1.4.1.253.8.62...	6/2/05 2:37:53 PM	5756
243.120.187.31	Device not responding	6/2/05 2:37:49 PM	6172
199.73.184.108	Authentication Failure (Enterprise = .1.3.6.1.4.1.253.8.62...	6/2/05 2:37:52 PM	6957
213.218.230.35	Authentication Failure (Enterprise = .1.3.6.1.4.1.253.8.62...	6/2/05 2:37:51 PM	6083

🟦	🟨	🟧	🟥
11	31	2	4

40 rows matched | admin | 192.168.34.117:0000

Figure B.1.9: An example view of Netcool/Webtop event list page.

The HP Open View and IBM Tivoli Netcool commercial management platforms offer similar functionality for events handling. However, IBM’s product offers a more configurable events list view and correlation mechanisms which enable users to aggregate and organise alerts for quicker problem identification.

References

[ABW]	Ubik S, Oslebo A., Antoniades D., ABW - Short-timescale passive bandwidth monitoring, CESNET Technical Report 3/2007 http://www.cesnet.cz/doc/techzpravy/2007/abw/abw.pdf
[ACRUW06]	D. L. Alderson, H. Chang, M. Roughan, S. Uhlig, and W. Willinger. The many facets of Internet topology and traffic. Networks and Heterogeneous Media, 1(4):569-600, December 2006. http://www.net.t-labs.tu-berlin.de/~steve/papers/nhm-paper.pdf
[AlsHeyDiff]	Alshammari R., Zincir-Heywood A. N., "Investigating Two Different Approaches for Encrypted Traffic Classification," Privacy, Security and Trust, Annual Conference on, pp. 156-166, 2008 Sixth Annual Conference on Privacy, Security and Trust, 2008.
[AlsHeyTwo]	Alshammari R., Zincir-Heywood A. N., "A Preliminary Performance Comparison of Two Feature Sets for Encrypted Traffic Classification", IEEE Computational Intelligence in Security for Information Systems CISIS 2008, October 2008.
[AngHey]	Angevine D., Zincir-Heywood A. N., "A Preliminary Investigation of Skype Traffic Classification Using a Minimalist Feature Set," Availability, Reliability and Security, International Conference on, pp. 1075-1079, 2008 Third International Conference on Availability, Reliability and Security, 2008.
[AutoBAHNE2Emon]	http://wiki.geant2.net/bin/view/JRA3/JRA3MonitoringDocumentation
[Blinc]	T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: Multilevel traffic classification in the dark. In ACM SIGCOMM, August 2005. http://conferences.sigcomm.org/sigcomm/2005/paper-KarPap.pdf
[Boosting]	Robert E. Schapire : The Boosting Approach to Machine Learning An Overview, Nonlinear Estimation and Classification, Springer, 2003.
[C4.5]	Kohavi R. and Quinlan J. R., Data mining tasks and methods: Classification: decision-tree discovery, pp. 267-276, 2002. http://portal.acm.org/citation.cfm?id=778254
[CDWY00]	J. Cao, D. Davis, S. V. Wiel, and B. Yu. Time-varying network tomography. Journal of the American Statistical Association, 95(452):1063-1075, 2000. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=866369
[CESNET]	http://www.ces.net/
[CircMon]	http://code.google.com/p/perfsonar-ps/wiki/CircuitMonitoringMoreDetails
[CM06]	R. R. Coifman and M. Maggioni. Diffusion Wavelets. Applied and Computational Harmonic Analysis, 21(1):53-94, July 2006.
[Combo6]	Combo6 Monitoring Card http://www.liberouter.org/card_combo6.php
[CREDENTIALS]	http://wiki.geant2.net/bin/view/JRA1/Jra1WorkingArea

[DAG]	DAG Network Monitoring Cards http://www.endace.com/dag-network-monitoring-cards.html
[DICE]	http://www.geant2.net/server/show/conWebDoc.1308
[DiMAPI]	Trimintzios P., Polychronakis M., Papadogiannakis A., Foukarakis M., Markatos E. P. & Oslebo A. (2006), DiMAPI: An Application Programming Interface for Distributed Network Monitoring., in Joseph L. Hellerstein & Burkhard Stiller, ed., 'NOMS' , IEEE, , pp. 382-393 http://www.ics.forth.gr/dcs/Activities/papers/dimapi.noms06.pdf
[Don06]	D. Donoho. Compressed Sensing. IEEE Transactions on Information Theory, vol 52, issue 4, pp 1289-1306, 2006. http://www.stanford.edu/~mlustig/CSMRI.pdf
[DMTF]	http://www.dmtf.org/home
[DTMF]	WS-management standard, DTMF http://www.dmtf.org/standards/wsman/
[DYN]	H. Dreger, A. Feldmann, M. Mai, V. Paxson, and R. R. Sommer. Dynamic application-layer protocol analysis for network intrusion detection. In USENIX Security Symposium, July 2006. http://www.icir.org/robin/papers/usenix06.pdf
[E2EMON]	https://wiki.man.poznan.pl/perfsonar-mdm/index.php/PerfSONAR_support_for_E2E_Link_Monitoring
[EMS]	Network Management System: Best Practices White Paper (Document ID: 15114) http://www.cisco.com/en/US/tech/tk869/tk769/technologies_white_paper09186a00800aea9c.shtml
[FNM]	Managing IP Networks with Free Software, Joe Abley and Stephen Stuart, NANOG 26
[FRATATM]	Kundan Misra - OSS for telecom networks: an introduction to network management
[GIdP]	http://gidp.geant2.net/
[gLS]	Hierarchically Federated Registration and Lookup within the perfSONAR Framework Zurawski, J. Boote, J. Boyd, E. Glowiak, M. Hanemann, A. Swany, M. Trocha, S. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4258584 http://anonsvn.internet2.edu/svn/nmwg/trunk/nmwg/doc/dLS/dLS_spec_1.html
[Hai]	HaiTao He, XiaoNan Luo, FeiTeng Ma, ChunHui Che1 and JianMin Wang, Network traffic classification based on ensemble learning and co-training, Science in China Series F: Information Sciences Springer Verlag, 2009.
[HPNNMAE]	HP Network Node Manager Advanced Edition 7.53 software Data sheet – PDF version https://h10078.www1.hp.com/cda/hpdc/display/main/download_pdf_unprotected.jsp?zn=bto&cp=54_4000_100
[i2]	https://dc211.internet2.edu/cgi-bin/perfAdmin/tree.cgi
[IBMTiv]	http://www-01.ibm.com/software/tivoli/products/netcool-omnibus/index.html
[JINI]	http://www.jini.org
[Jtraceroute]	http://perfmon.cesnet.cz/jtraceroute
LHCOPN	http://lhcopn.web.cern.ch/lhcopn/
[Mal99]	S. Mallat. A Wavelet Tour of Signal Processing, Academic Press, 1999
[ML]	http://monalisa.caltech.edu/
[MLAng]	Why we need a NETCONF-Specific Modeling Language, B. Lengyel, Internet Draft
[MLarch]	http://monalisa.caltech.edu/monalisa__System_Design.htm
[MLComInfra]	http://monalisa.caltech.edu/monalisa__System_Design__communication_infrastructure.html

[MTPP10]	Halak, J. and Ubik, S. 2009. MTPP - Modular Traffic Processing Platform. In Proceedings of the 2009 12th international Symposium on Design and Diagnostics of Electronic Circuits&Systems - Volume 00 (April 15 - 17, 2009). DDECS. IEEE Computer Society, Washington, DC, 170-173. http://dx.doi.org/10.1109/DDECS.2009.5012121
[NetMate]	NetMate - Network Measurement and Accounting System http://www.ip-measurement.org/
[NetOM]	Consolidated fault monitoring for real-time service management - Netcool/OMNIbus Data sheet – PDF version http://www.orb-data.com/orbcms/files/downloads/omnibus/Netcool%20Omnibus%20Data%20Sheet.pdf
[NetRep]	http://www-01.ibm.com/software/tivoli/products/netcool-reporter/index.html
[NetWeb]	http://www-01.ibm.com/software/tivoli/products/netcool-webtop/index.html
[NguGrenArm07]	T. T. T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. Communications Surveys & Tutorials, IEEE, 10(4):56–76, 2008.
[NM]	Network management requirements/recommendations, Vidar Faltinsen, Network monitoring workshop for GN3/NA3/T4
[NMWG]	http://nmwg.internet2.edu/
[OASIS]	http://www.oasis-open.org
[OpenMP]	http://www.openmp.org/
[ORTC]	Jeffrey Erman, Anirban Mahanti, Martin Arlitt, Ira Cohen, Carey Williamson , Offline/Realtime Traffic Classification Using Semi-Supervised Learning , HP Laboratories Palo Alto , July 13, 2007. http://www.cse.iitd.ac.in/~mahanti/papers/sigmetrics07.pdf
[OSI]	Network And Distributed Systems Management, Morris Sloman, 1994
[Ple06]	Pleva L. Advanced Traceroute, BS Thesis, Czech Technical University, 2006. http://www.cesnet.cz/doc/techzpravy/2007/advanced-traceroute/
[PostgreSQL]	http://www.postgresql.org/
[REGISTER]	http://perfmon.cesnet.cz/jtraceroute/request
[ROC03]	Nicolas Lachiche, Peter Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.
[RFC1157]	A Simple Network Management Protocol, IETF http://www.ietf.org/rfc/rfc1157.txt
[RFC2578]	Structure of Management Information Version 2 (SMIv2), IETF http://www.rfc-editor.org/rfc/rfc2578.txt
[RFC2679]	A One-way Delay Metric for IPPM, IETF http://www.ietf.org/rfc/rfc2679.txt
[RFC2680]	A One-way Packet Loss Metric for IPPM, IETF http://www.ietf.org/rfc/rfc2680.txt
[RFC3393]	IP Packet Delay Variation Metric for IP Performance Metrics (IPPM), IETF http://www.ietf.org/rfc/rfc3393.txt
[RFC341x]	RFC3410-RFC3418 SNMPv3 RFCs, IETF http://www.rfc-editor.org/rfc/rfc3410.txt http://www.ietf.org/rfc/rfc3411.txt

	http://www.rfc-editor.org/rfc/rfc3412.txt http://www.rfc-editor.org/rfc/rfc3413.txt http://www.rfc-editor.org/rfc/rfc3414.txt http://tools.ietf.org/html/rfc3415 http://www.rfc-editor.org/rfc/rfc3416.txt http://www.rfc-editor.org/rfc/rfc3417.txt http://www.rfc-editor.org/rfc/rfc3418.txt
[RFC4741]	NETCONF Configuration Protocol, IETF http://www.rfc-editor.org/rfc/rfc4741.txt
[RFC4743]	Using NETCONF over the Simple Object Access Protocol (SOAP), IETF http://www.ietf.org/rfc/rfc4743.txt
[RFC5388]	Information Model and XML Data Model for Traceroute Measurements. RFC5388, IETF http://www.ietf.org/rfc/rfc5388.txt
[RRW08]	D. Rincón, M. Roughan, and W. Willinger. Towards a menaingful MRA of traffic matrices. Internet Measurement Conference 2008, pp 331-336, 2008. http://delivery.acm.org/10.1145/1460000/1452559/p331-rincon.pdf?key1=1452559&key2=6771257521&coll=GUIDE&dl=GUIDE&CFID=60240960&CFTOKEN=85875635
[SIMPLETEST]	http://perfsonar.acad.bg/psui_beta/perfsonar.jnlp
[SMAPL]	https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-15-119^1155_4000_100__
[SNM]	Scaling network management tools, Olav Kvittem, NANOG 29
[SNMP]	Net-SNMP software http://www.net-snmp.net/
[SNORT]	http://www.snort.org
[SSNTC]	Jeffrey Erman, Martin Arlitt, Martin Arlitt, Semi-supervised network traffic classification, Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems ,2007.
[Stager]	http://software.uninett.no/stager/
[Surv]	Thuy T.T. Nguyen, Grenville Armitage , A Survey of Techniques for Internet Traffic Classification using Machine Learning, IEEE Communicatiibs Surveys and Tutorials, 2008 http://caia.swin.edu.au/cv/garmitage/things/Nguyen_Armitage_SurveysAndTutorials2008.pdf
[TMF]	http://www.tmforum.org/browse.aspx
[TNM]	http://www-01.ibm.com/software/tivoli/products/network-mgrproductline/
[UQLB06]	S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon. Providing public intradomain traffic matrices to the research community. SIGCOMM Computer Communication Review, 36(1):83-86, 2006. http://alumni.info.ucl.ac.be/suh//papers/traffic-matrices.pdf
[Var96]	Network tomography: estimating source-destination traffic intensities from link data. J. of the Am. Statistical Association, 91:365-377, 1996.
[WangZheng09]	WANG Y., YU S.. Supervised Learning Real-time Traffic Classifiers. Journal of Networks, North America, 4, sep. 2009. http://www.academypublisher.com/ojs/index.php/jnw/article/view/0407622629
[Weka]	Weka - Data Mining Software in Java http://www.cs.waikato.ac.nz/ml/weka/

[WilZanArm06]	Williams N., Zander S., Armitage G.. Evaluating machine learning algorithms for automated network application identification. Swinburne University of Technology; 2006 http://caia.swin.edu.au/reports/060410B/CAIA-TR-060410B.pdf
[WSDM]	Web Services Distributed Management: Management Using Web Services, OASIS, wsdm-muws1-1.1-spec-os-01
[WSLA]	http://www.research.ibm.com/wsla/
[YANG]	Mapping YANG to Document Schema Definition Languages and Validating, L. Lhotka, R. Mahy and S. Chisholm, Internet Draft
[YANG1]	YANG - A data modeling language for NETCONF, M. Bjorklund, draft-ietf-netmod-yang-08
[ZRDG03]	Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale IP traffic matrices from link loads. In ACM SIGMETRICS, pages 206-217, San Diego, California, June 2003. http://www.cs.utexas.edu/~yzhang/papers/tomogravity-sigm03.pdf
[ZRWQ09]	Y. Zhang, M. Roughan, W. Willinger, L. Qiu. Spatio-temporal compressive sensing and internet traffic matrices. SIGCOMM Computer Communication Review, 39(4):267-268, 2009. http://www.cs.utexas.edu/~yzhang/papers/sensing-sigc09.pdf

Glossary

AA	Authentication and Authorisation
AAA	Authentication, Authorisation and Accounting
AAI	Authentication and Authorisation Infrastructure
ABW	Available Bandwidth
AC	Automated Client
ACL	Access Control List
API	Application Programming Interface
AS	Authentication and Authorisation Service
Bayes Net	Bayesian Networks
BWCTL	Bandwidth Controller
CA	Certificate Authority
CIM	Common Information Model
cNIS	common Network Information Service
CPU	Central Processing Unit
DAMe	Deploying Authorization Mechanisms for Federated Services in the eduroam architecture
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DM	Domain Manager
DMTF	Distributed Management Task Force
DN	Distinguished Name
DNS	Domain Name System
DPI	Dots Per inch
DRAM	Dynamic Random Access Memory
DW	Diffusion Wavelets
FBA	Flow Based Approach
FR	Flow Records
FTP	File Transfer Protocol
GB	Gigabyte
GHz	Gigahertz
GIdP	GÉANT Identity Provider
gLS	global Lookup Service
GPU	Graphics Processing Unit
GSI	Globus Security Infrastructure
GSS	Generic Security Services
GUI	Graphical User Interface
hLS	home Lookup Service

HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICT	Information, Communications and Technology
IDC	Interdomain Controller
IdP	Identity Provider
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPFIX	IP Flow Information Export
JDBC	Java Database Connectivity
JRA	Joint Research Activity
JVM	Java Virtual Machine
KNN	K-Nearest Neighbor
LHCOPN	Large Hadron Collider Optical Private Network
LS	Lookup Service
LUS	Lookup Discovery Service
MDM	Multi-Domain Monitoring
MA	Measurement Archive
MIB	Management Information Base
ML	Machine Learning
MOM	Manager of Managers
MP	Measurement Point
MPLS	Multiprotocol Label Switching
MRA	Multi-Resolution Analysis
NB Tree	Naive Bayes Tree
NBD	Naive Bayes with Discretisation
NDL	Network Description Language
NMC-WG	Network Measurement and Control Working Group
NML	Network Markup Language
NM-WG	Network Measurements Working Group
NOC	Network Operations Center
NREN	National Research and Education Network
OD	Origin-Destination
OGF	Open Grid Forum
OS	Operating System
OSI	Open Systems Interconnection
OV-NNM	Open View – Network Node Manager
P2P	Peer-to-Peer
PABA	Payload Based Approach
PC	Personal Computer
perfSONAR	Performance Service-Oriented Network Monitoring Architecture
PHP	PHP: Hypertext Preprocessor
PKI	Public Key Infrastructure
POBA	Port Based Approach
PoP	Point of Presence
POSIX	Portable Operating System Interface

QoS	Quality of Service
RAM	Random-Access Memory
R-BE	Remote Bridging Element
RFC	Request for Comments
ROC	Receiver Operating Characteristic
RPC	Remote Procedure Calls
RRD	Round Robin Database
RTT	Round Trip Time
SA	Service Activity
SAML	Security Assertion Markup Language
SASL	Simple Authentication and Security Layer
SDK	Software Development Kit
SLA	Service Level Agreement
SLO	Service Level Objective
SLS	Service Level Specification
SMI	Structure of Managed Information
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SRAM	Static Random Access Memory
SSL	Secure Sockets Layer
T	Task
TCP	Transmission Control Protocol
Telnet	Teletype Network
TLS	Transport Layer Security
TM	Traffic Matrix
TrS	Transformation Service
UbC	User behind a Client
UNIS	Unified Network Information Service
URL	Uniform Resource Locator
VoIP	Voice over IP
WAN	Wide Area Network
WE	Client in a Web container
WSDL	Web Services Description Language
WS-SEC	Web Services Security
XML	Extensible Markup Language