

INFORMATION SOCIETY TECHNOLOGIES (IST) PROGRAMME



**A Scalable Monitoring Platform for the Internet
Contract No. IST-2001-32404**

D0.2 “Measurement-based application requirements”

Abstract: This document describes the user requirements for the SCAMPI network monitoring system. The requirements are based on the findings of the first deliverable, “Description and analysis of the state-of-the-art”, and on the experience of the various partners.

Contractual Date of Delivery	31 August 2002
Actual Date of Delivery	8 October 2002
Deliverable Security Class	Public
Editors	Arne Øslebø and Jon Kåre Hellan
Contributors	CESNET, FORTHnet, NETIKOS, UNINETT

The SCAMPI Consortium consists of:

TERENA
IMEC
FORTH
LIACS
NETikos
Uninett
CESNET
FORTHnet
4Plus
Siemens

Coordinator
Principal Contractor
Principal Contractor
Principal Contractor
Principal Contractor
Principal Contractor
Principal Contractor
Principal Contractor
Principal Contractor
Principal Contractor

The Netherlands
Belgium
Greece
The Netherlands
Italy
Norway
Czech Republic
Greece
Greece
Germany

Contents

1	Introduction	4
1.1	Terminology	4
1.1.1	user	4
1.1.2	network flow	4
1.1.3	flow record	4
1.1.4	NIC	4
1.1.5	resolution, accuracy and precision	4
2	Applicability	5
2.1	Accounting	5
2.2	DoS Attack detections	5
2.3	Intrusion detection	5
2.4	Network debugging	5
2.5	Traffic profiling/engineering	5
2.6	QoS monitoring	6
2.7	Summary	6
3	Protocol analysis	7
3.1	Physical layers	7
3.2	Link protocols	7
3.3	Network protocols	7
3.4	Packet selection	7
3.4.1	Sampling methods	8
4	Monitoring System requirements	9
4.1	System environment	9
4.1.1	Security	9
4.1.2	Operating systems	9
4.1.3	Management	9
4.2	Application Programming Interfaces	10
4.2.1	Passive Monitoring	10
4.2.2	Active Monitoring	11
4.2.3	Management	11
5	Hardware requirements	12
5.1	Monitoring Platform	12
5.2	Monitoring Hardware Support	12
5.2.1	Timestamp requirements	13
6	Application requirements	14
6.1	NetFlow/IPFIX probe	14
6.1.1	General requirements	14
6.1.2	Flow record formats	14
6.1.3	User requirements	14
6.2	Flow-based reporting	14
6.2.1	Data collection	14
6.2.2	Reports	15
6.2.3	Aggregation	16
6.2.4	User interface	16
6.3	Threshold alerting application	16
6.3.1	General requirements	16
6.3.2	User requirements	16

6.4	Security applications	17
6.5	QoS monitoring	17
6.5.1	General requirements	17
6.5.2	Protocol requirements	17
6.5.3	User requirements	17

1 Introduction

This document describes the user requirements for the SCAMPI network monitoring system. It is based on the reviews made in the product evaluation document [5] and also input from the project partners. This document will be the base for the design phase.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

1.1.1 user

A user is a advanced Internet Service Provider needing both short term operational information on the traffic as well as long term trends for strategic purposes. The user might also be doing traffic research within universities and research institutes.

1.1.2 network flow

The definition of a network flow in this document follows the one defined in the IPFIX requirements document [3]:

A flow is defined as a set of IP packets passing an observation point in the network during a certain time interval. All packets belonging to a particular flow have a set of common properties. Each property is defined as the result of applying a function to the values of:

1. *one or more packet header field (e.g. destination IP address)*
2. *one or more properties of the packet itself (e.g. packet length)*
3. *one or more of fields derived from packet treatment (e.g. the BGP Autonomous System number [RFC1771])*

1.1.3 flow record

The definition of a flow record is also taken from the IPFIX requirements document [3]:

A flow record contains information about a specific flow that was metered at an observation point. A flow record contains measured properties of the flow (e.g. the total number of bytes of all packets of the flow) and usually also characteristic properties of the flow (e.g. source IP address).

1.1.4 NIC

Network Interface Card. An expansion card that makes it possible for a computer to perform network input and output.

1.1.5 resolution, accuracy and precision

The terms resolution, accuracy and precision are used as defined in [RFC-2330] with respect to [RFC-1305]:

Resolution (sometime called granularity) is the smallest unit by which the clock is advanced.

Accuracy the offset between local and some other (usually UTC) clock.

Precision the smallest unit in which time is reported.

2 Applicability

The following section describes a selection of applications that will find the SCAMPI platform useful. Many of the requirements in the following sections are derived from the applications described here. The goal of this section is not to provide a full list of applications that can use the SCAMPI platform, but to provide a selection of what is considered the most significant applications.

2.1 Accounting

Accounting is an important application for measurement based systems. The amount of transferred traffic on the network is recorded for each customer and this information is then used for billing. For some uses only the total traffic is counted while others might divide the traffic into categories according to type of traffic, time of day etc. This means that the underlying platform that collects the information must have the capabilities for differentiating between types of traffic and be configurable as to how this is done.

Accounting applications can get all necessary information from flow records and do not need access to each individual packet. Depending on the requirement for accuracy, sampling may or may not be acceptable.

2.2 DoS Attack detections

Detecting Denial of Service (DoS) attacks can be done by analyzing aggregated statistics based on information from flow records. If the total number of flows to individual hosts are reported, DoS attacks can be detected when this number grows unnaturally high. When a host has been identified as being the subject of a DoS attack, individual flow records can be studied in more detail and a defense strategy can be devised.

Packet sampling is sufficient for detecting DoS attacks and there is no need to look into packet payload.

2.3 Intrusion detection

Intrusion detection is the problem of identifying individuals who are using a computer system over a network without authorization. An intrusion attempt often starts with a port scan of the targeted host. This port scan can be detected by just analyzing flow records. However, to get detailed information about the intrusion attempt itself it is necessary to analyze each packet separately and preferably also to look at the contents of packets to search for certain substrings.

Packet sampling can cause problems for intrusion detection since important packets may be left out of the analysis.

2.4 Network debugging

Network debugging is used to detect anomalous behavior on the network. Some problems can be detected from flow records. However, a detailed analysis of the header of each packet is often required. If higher level protocols or tunneled protocols are being debugged, some part of the packet payload also needs analyzing.

Network problems can sometimes be related to timing. To detect this it is important that the underlying monitoring platform provides an accurate timestamp with high resolution.

Sampling can cause problems with network debugging since important packets that can identify problems may be left out by the sampling process.

2.5 Traffic profiling/engineering

Traffic profiling characterizes IP flows according to key parameters of the flows such as duration, volume, time and burstiness. It is heavily used for network planning and dimensioning, trend analysis and creating business models. It is also used by traffic engineering which is concerned with performance optimization of operational networks.

Application	Flow records	Packets	Header	Payload	Aggregated stats	Sampling
Accounting	X	-	X	-	X	(X)
DoS attack	X	-	X	-	X	X
Intrusion detection	-	X	X	X	-	-
Network debugging	-	X	X	(X)	-	-
Traffic profiling/engineering	X	-	X	-	X	X
QoS monitoring	(X)	X	X	-	X	(X)

Table 1: Application requirements

Aggregated statistics based on flow records will usually give all the information that is needed for these kind of applications and sampling is sufficient.

2.6 QoS monitoring

Quality of Service (QoS) monitoring is the passive observation of transmission quality in the network. Often this kind of monitoring requires multiple observation points in the network, and synchronization of the clock is important. Some common QoS parameters that are often monitored are:

- round trip time - requires packet matching techniques to match requests and responses so that the RTT time can be calculated.
- one way delay - requires collection of monitoring data at two observation points.
- packet loss - can be done with either two observations points or by one observation point where sequence numbers in the packets are analyzed.
- one way delay variation - this is calculated by performing one way delay monitoring or hash based trajectory sampling.
- Bandwidth usage and throughput may be measured with flow based statistics.

Many of the QoS parameters that are interesting will have problems with packet sampling, as packets being omitted during sampling will lead to inaccurate results. Some parameters can be monitored using flow records, but analyzing each packet is preferable.

2.7 Summary

The description given in the previous sections about the various applications that can take advantage of the SCAMPI platform clearly shows that the needs vary. Some applications can be based on flow records while others need access to each packet in a flow. Some only need the information from the header while others need access to part or all of the the packet payload. Yet other applications only need access to aggregated statistical measurements of the network traffic.

The SCAMPI platform will provide the means for analyzing each packet at high speed and also opens up the possibility of scanning for patterns inside the payload of packets. The SCAMPI platform may also provide an API that allows existing programs that are built using existing APIs, like libpcap, to take advantage of high speed NICs without any change in the programs. This will allow many existing programs to be used on high speed networks without having to change any source code.

Table 1 shows a summary of features that the various applications need.

3 Protocol analysis

This chapter describes need for support for different protocols and media as well as packet selection techniques.

3.1 Physical layers

Both single-mode and multi-mode fiber **MUST** be supported, and for single mode both short range 1310 nm and long range 1530 nm lasers. Both the SONET and SDH framing **MUST** be supported for 2.5 Gbps (OC-48/STM-16) and 10 Gbps (OC-192/STM-64) links. The system **SHOULD** support Ethernet on twisted pair copper.

3.2 Link protocols

- The system **MUST** support Packet over SONET (RFC2615) for 2.5 Gbps and 10 Gbps links. It **SHOULD** support generic HDLC-like encapsulation and it **MUST** support the widely used Cisco HDLC.
- The system **SHOULD** support 10 gigabit/s Ethernet (IEEE 802.3ae).
- The system **MUST** support 1 gigabit/s Ethernet.
- The system **SHOULD** support 10/100 Mbit/s Ethernet.
- MPLS **SHOULD** be supported to an extent where packets are decapsulated to a chosen nesting level.
- The system **MAY** have some level of awareness for ATM and Frame Relay like decapsulation of packets (ATM SAR), but no signalling support is needed.

3.3 Network protocols

- The system **MUST** support IPv4 and IPv6 (IP) basic encapsulation, and **SHOULD** be able to filter based on IPv4 option fields as well as IPv6 header extensions.
- The system **SHOULD** be able to unwind IP tunnels to a chosen level of nesting (GRE, IPIP).
- There **MUST** be generic support for filtering within any protocol header type, like OSPF or IS-IS, but no specific protocol knowledge is required.

3.4 Packet selection

The system **SHOULD** be able to select packets on complex criteria/masks and combination of them. The design **SHOULD** relate to the ongoing work in the IETF groups like PSAMP.

The NIC **SHOULD** be able to match all header fields identifying a flow (ipfix) or a routing equivalence class (with particular TOS field) and protocol handshaking (such as TCP flags). This requires at least 15 bytes for IPv4 or 23 bytes for IPv6 to be matched in IP and TCP/UDP headers. The most general solution would be to match on any field in Ethernet, MPLS, IP, TCP and UDP headers as well as first 8 bytes of data (RTP/UDP).

Mechanisms to define useful and not useful packet parts, i.e. what parts of useful packets should be stored, **MUST** be added. This could reduce amount of data being stored. It **MUST** be possible to choose a number of bytes from the start of the packet. It **SHOULD** be possible to add another part of the packet and it **MAY** be possible to support a list of parts of the packet.

The system **SHOULD** provide provisions for correlating packets captured by two different SCAMPI platforms, such as precise timestamps and packet signatures. This signature **SHOULD** be computed over all parts of a packet that remain the same during packet transport over a network and **SHOULD** be logged along with timestamp for each logged packet. The signature **SHOULD** be computed in the adapter when a

special purpose adapter is used. The original packet and its first fragment **SHOULD** be identified as being the same packet.

The system **SHOULD** be able to handle back to back minimum size packets at the maximum data rate.

3.4.1 Sampling methods

Users will be able to choose among sampling method for reducing the amount of traffic to be processed. The methods **SHOULD** be conformant to interim work of the IETF psamp group. The methods considered are

- Selection of every N packet **MUST** be implemented.
- Selection by hashing on header fields **MUST** be implemented. As a minimum, hashing on addresses and ports **MUST** be supported.
- Selection by user defined hash functions **MAY** be implemented.
- Selection by statistical functions like random **MAY** be implemented.

4 Monitoring System requirements

This section describes requirements as to which services and programming interfaces should be available to applications.

4.1 System environment

It is assumed that a SCAMPI monitoring system will run on a dedicated computer. The computer may be owned and managed by the network operator, or by another party. Running the SCAMPI monitoring system is the only task for this computer.

4.1.1 Security

It may be desirable to use the monitoring system for multiple simultaneous measurement activities. In some cases, the potential for interference may not be well known ahead of time, different legal entities may be involved, or there may be regulatory limitations on what information the system may return to the user.

In other cases, it will be safe to assume that the parties involved will be able to and motivated to coordinate their activities without support from the system. This may permit higher performance from the system.

- It **SHOULD** be possible to configure the system in a way that conflicts between simultaneous activities are prevented and resource limitations enforced.
- It **SHOULD** be possible to turn off security and resource control mechanisms when desired.
- It **SHOULD** be possible to manage users and user privileges. Standard mechanisms of the chosen operating system **SHOULD** be used.

4.1.2 Operating systems

When choosing which operating systems to support, both programming interfaces and driver availability have to be considered. It is good programming practice to program to portable APIs when possible. However, parts of the monitoring platform will have to be implemented at the operating system level. Furthermore, the network adapters will be leading edge, low volume models for which operating system driver support will be uneven. Thus, it will not be practical to implement the system without access to source code for the operating system and drivers.

- Software which is not intended to run as part of the operating system kernel, **SHOULD** be portable between Unix like systems.
- The operating system chosen **SHOULD** have freely available source code.

4.1.3 Management

It **SHOULD** be possible to access usage statistics and information about the status of the SCAMPI platform via SNMP. As a minimum, the following information **MUST** be available:

- host system memory usage
- host system CPU usage
- host system disk usage
- network link status of monitoring hardware
- congestion warning (should let users know when packets have been dropped because of congestion in the SCAMPI platform)

- number of active MAPI network flows
- number of active MAPI filters
- number of active MAPI functions
- total number of packets received by MAPI.

The following information **SHOULD** be available:

- cause of congestion
- number of packets dropped as a result of congestion
- number of packets received by monitoring hardware
- special purpose adapter resource usage.

The following information **MAY** be available:

- number of received packets per active MAPI network flow.

4.2 Application Programming Interfaces

Applications will interact with SCAMPI through its application programming interfaces.

- An API which supports passive monitoring **MUST** be provided.
- An API which supports management of the monitoring system **MUST** be provided.
- An API which supports packet generation **MAY** be provided.
- Legacy libpcap based monitoring applications **MUST** be supported.

4.2.1 Passive Monitoring

Passive monitoring entails giving applications

- access to packets on the monitored link.
- the ability to install filters and code to select and process packets.

MAPI is the SCAMPI API for passive monitoring. The requirements for MAPI are:

- MAPI **MUST** support packet filtering and sampling as specified in section 3.4.1 to define a packet subset of the network traffic.
 - MAPI **MUST** support configuration of filtering and sampling on special purpose hardware when available.
 - MAPI **MAY** support configuration of filtering on intelligent routers.
- MAPI **SHOULD** support dividing the packet subset into further subsets.
- It **MUST** be possible to read all packets in packet subsets.
- It **SHOULD** be possible to reassemble fragmented packets and remove retransmitted packets from packet subsets.
- It **SHOULD** be possible by means of MAPI to apply user specified functions to packet subsets.
- It **MUST** be possible in MAPI to define how separate network flows from packet subsets.

- MAPI **MUST** support reading flow records generated from packet subsets.
- MAPI **SHOULD** support non blocking packet capture via callbacks. This facility **SHOULD** be usable from any main loop which provides timeout and event notification.
- It **MUST** be possible to process with MAPI both data captured in real time and previously captured data read from storage.

4.2.2 Active Monitoring

- An API **SHOULD** be made available for performing trajectory sampling (see [1]).
- An API **MAY** be made available for performing packet generation.

4.2.3 Management

It **SHOULD** be possible to access usage statistics and information about the status of monitoring hardware via an API.

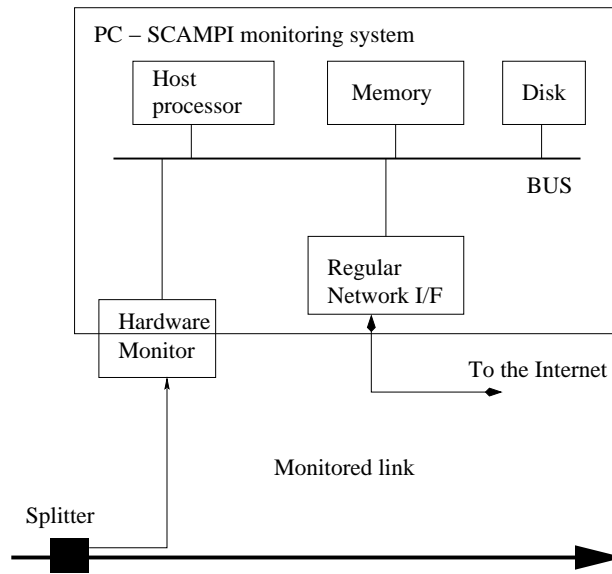


Figure 1: The SCAMPI network monitoring system

5 Hardware requirements

This section describes the hardware components that will be part of SCAMPI.

5.1 Monitoring Platform

The SCAMPI platform **SHOULD** be built on a reasonably priced standard computer and **MUST** be able to handle traffic rates of 10 Gbps and above. Capturing and storing information about all packets at these speeds is not a feasible solution. Even at 2.5 Gbps the standard PCI bus is close to its limits when it comes to throughput. At 10 Gbps, the disk system also becomes a problem as storing information about every packet would need a disk bandwidth of around 250 Mbyte/s [7]. To be able to handle high speeds, traffic must be reduced by using flow records, sampling and/or filtering.

The exact hardware requirements will vary according to network traffic rates and traffic reduction methods used. However, to be able to successfully measure at 10 Gbps and above the hardware **SHOULD** have the following minimum specifications:

- Bus: PCI-X 1.0
- CPU: ≥ 1.8 GHz
- RAM: ≥ 1 GB
- Storage: SCSI Ultra 160 (160 MB/s) (preferably a RAID configuration)

Measuring at lower speeds **SHOULD** be possible with a more reasonably priced configuration.

The overall architecture of the platform's components is shown in figure 1.

5.2 Monitoring Hardware Support

The SCAMPI platform **MUST** support various types of monitoring hardware. For measuring at lower speeds it **SHOULD** support input from all commodity network adapters that can be set to promiscuous mode. This method will also support input from intelligent network routers like Juniper.

For high speed traffic measuring, the SCAMPI platform **MUST** be able to support special purpose adapters that have on board processing capabilities. This processing capability can be used for implementing techniques for reducing the amount of traffic sent to higher software levels. As a minimum, the SCAMPI platform **SHOULD** support techniques like:

- Filtering - the adapter filters out unwanted traffic.
- Sampling - the adapter performs sampling of network packets as described in section 3.4.1.
- Flow records - the adapter analyzes network packets and collects the information in flow records.

Among existing special purpose adapters, support for Endace DAG[6] cards **SHOULD** be implemented, and support for one network processor based adapter **SHOULD** be implemented.

5.2.1 Timestamp requirements

A special purpose adapter **MUST** include an on board precise clock for timestamping. A stable TCX or OCX oscillator is one way to implement this. The resolution of timestamps should be fine enough to measure the interval between two successive packets on the interface. Allowing for what can be realized in practice, the resolution **SHOULD** not be worse than 50 ns, i.e. a 20 MHz clock oscillator. It **MUST** be possible to set up and to continuously fine adjust the on-board clock phase and rate by software running at the host computer. It **MUST** be possible to synchronize the clock by an external PPS (pulse per second) signal, and it **SHOULD** be possible to set up it by GPS receiver via the NMEA serial protocol.

The accuracy of current GPS receivers ranges from 1 microsecond up to 50 ns. It is possible to achieve an accuracy of about 1 millisecond by synchronizing the clock via a level 3 network protocol, i.e. NTP. Therefore, the absolute accuracy of the adapter clock depends on the synchronization method and **SHOULD** be several microsecond when GPS is used.

To be compatible with future needs and current standards (NTP time representation and 'struct timespec' used in Unix like systems), the software **SHOULD** support a timestamp precision of 1 ns. The timestamp format **SHOULD** allow unique timestamping for a minimum of 30 years before the value "wraps" back to zero.

6 Application requirements

The following are requirements for the applications that use the SCAMPI platform and will be implemented as part of the SCAMPI project.

6.1 NetFlow/IPFIX probe

This is a small application that uses the SCAMPI platform to export flow records according to the NetFlow and IPFIX standard.

6.1.1 General requirements

The application **MUST** comply to the IPFIX standard.

6.1.2 Flow record formats

The application **MUST** be able to support several formats for exporting flow records. As a minimum it **MUST** support:

- IPFIX
- NetFlow version 5

It **MAY** support:

- NetFlow version 8

6.1.3 User requirements

- A user **MUST** be able to configure:
 - type of format that is used for exporting flow records.
 - sampling parameters for sampling of packets.
 - IPFIX configuration as specified in the IPFIX standard (e.g. templates, flow distinguishing).
- Configuration of the application through a configuration file **MUST** be supported.
- Configuration of the application through through a WEB interface **MAY** be supported.

6.2 Flow-based reporting

This application will generate flow based reports based on input from NetFlow/IPFIX records as well as the SCAMPI MAPI. The application should take advantage of the fact that the SCAMPI platform can analyze each packet to generate novel reports not available by other applications.

6.2.1 Data collection

The application **MUST** be able to accept data from:

- NetFlow version 5 flow records
- IPFIX flow records
- SCAMPI MAPI

The application **MAY** be able to accept data from:

- NetFlow version 8 flow records
- NetFlow version 7 flow records

6.2.2 Reports

The application **MUST** support a multitude of reports and it **MUST** be easy to add new reports in the future. Some reports will only be available when using MAPI as input. As a minimum the following reports **MUST** be supported:

- Average packet size distribution - shows how many flows and packets that have the same average packet size.
- Packets per flow distribution - shows how many flows that contain the same number of packets.
- Octets per flow distribution - shows how many flows that contain the same number of octets.
- Flow duration distribution - shows how many flows that have the same duration.
- UDP/TCP destination port - shows how many flows, packets and octets that have the same UDP/TCP destination port.
- UDP/TCP source port - shows how many flows, packets and octets that have the same UDP/TCP source port.
- Source IP - shows how many flows, packets and octets that have the same source IP address.
- Destination IP - shows how many flows, packets and octets that have the same destination IP address.
- Source/Destination IP - shows how many flows, packets and octets that have been sent between pairs of source and destination IP addresses.
- Source AS - shows how many flows, packets and octets that have the same source AS.
- Destination AS - shows how many flows, packets and octets that have the same destination AS.
- Source/Destination AS - shows how many flows, packet and octets that have been sent between pairs of source and destination AS.
- IP ToS - shows how many flows, octets and packets that have the same ToS value.
- Source prefix - shows how many flows, packets and octets that have the same source prefix.
- Destination prefix - shows how many flows, packets and octets that have the same destination prefix.
- Source/Destination prefix - shows how many flows, packets and octets that have been sent between pairs of source and destination prefix.

The following reports **SHOULD** be supported:

- Distribution of average time interval between packets - shows how many flows that have the same average time interval between packets.

The following reports **MAY** be supported:

- Packet size distribution per flow - shows the packet size distribution in each flow.
- Time interval distribution per flow - shows the distribution per flow of time interval between packets.
- Bandwidth utilization - shows the bandwidth utilization for the monitored links.
- Retransmitted packets - shows the number of retransmitted packets for flows
- Time interval - shows the average time interval between packets in a flow and the standard deviation.

6.2.3 Aggregation

- Aggregation of reports over hours, days, weeks, months and years **MUST** be supported.
- Aggregated data **SHOULD** be stored in an SQL database.

6.2.4 User interface

There **SHOULD** be both a text based command line (CLI) interface and a web based interface.

Web based interface

- Reports **MUST** be available as both tables and graphs.
- It **MUST** be easy for users to navigate between different time periods and time resolutions.
- It **MUST** be easy for users to navigate between various reports.
- It **MUST** possible to sort the report according to the various fields in the report.

CLI based interface

- It **MUST** be easy to specify time period and resolution for reports
- Reports **MUST** be presented in simple ASCII format for easy import to other programs for further processing.
- It **MUST** possible to sort the report according to the various fields in the report.

6.3 Threshold alerting application

This application will raise real time alerts when various thresholds, defined by the users, are violated.

6.3.1 General requirements

- Create flow alarm notifications based on thresholds violations:
 - on packet loss,
 - inter-packet delay variance,
 - the number of packets of a specific source,
 - a specific packet size,
 - the number of the same protocol packets (e.g ICMP, UDP), etc.
- Create flow alarm notification when an intrusion is detected.
- Create flow alarm notification when packet corruption threshold is exceeded.
- Create flow alarm notification when packet re-transmission threshold is exceeded.
- Create flow alarm notification when abnormal traffic patterns to or from specific IPs are detected.

6.3.2 User requirements

- There **SHOULD** be a WEB based interface for configuring alarm notifications.
- Users **MUST** be able to receive alarms through email.
- There **SHOULD** be a web page that shows alarm history.

6.4 Security applications

An existing application for detecting intrusions and denial of service - most likely Snort [8] - **SHOULD** be adapted to make use of SCAMPI services. This entails:

- A facility for loading rules from the application to a special purpose adapter **SHOULD** be added to the application.
- The application **MAY** be modified to make use of MAPI as well as libpcap.

See also 4.2 - bullet point 4, and 4.2.1 - final bullet point.

6.5 QoS monitoring

This application monitors primary network QoS characteristics including throughput, packet loss rate, one-way delay and jitter of a specified flow. A flow, as specified, may represent aggregated traffic at different levels of aggregation.

6.5.1 General requirements

This application **SHOULD** relate to the ongoing work in the IPPM IETF working group.

Monitoring of some QoS characteristics (such as packet loss rate) requires correlating data from two measurement points. Therefore, the SCAMPI architecture **MUST** provide provisions for correlating packets captured by two different SCAMPI platforms, such as precise timestamps and packet signatures for passive measurements or sequence numbers for active measurements. Based on these provisions, the QoS monitoring application **MUST** be able to correlate packets from two different SCAMPI platforms.

The application **SHOULD** support both real-time measurements and measurements using previously captured packets.

6.5.2 Protocol requirements

If active measurement is used to monitor some QoS characteristics (such as one-way delay), the application **SHOULD** support widely accepted measurement protocol, such as the one-way active measurement protocol proposed by the IPPM IETF working group.

The application **MUST** have built-in knowledge of IPv4, IPv6, UDP, TCP and RTP protocols needed to perform monitoring of QoS characteristics related to these network protocols. Other network protocols **MUST** be supported in a generic way, by specifying and comparing byte offsets and patterns without built-in specific knowledge of these protocols.

The application **SHOULD** allow to make performance analysis in order to identify performance limiting factors.

6.5.3 User requirements

A user **MUST** be able to configure:

- Rules to specify a flow to be monitored
- Location of measurement points in case of two-point measurement
- QoS characteristics to be monitored including parameters such as precision, time granularity, etc.

References

- [1] "Trajectory sampling for direct traffic observation", Duffield and Grossglauser, SIGCOMM'00
- [2] "IP Flow Information Export", IETF working group
- [3] Requirements for IP flow Information Export. Internet draft draft-ietf-ipfix-reqs-02.txt.
- [4] "Hash-based IP traceback", Snoren, Partridge &al. BBN, SIGCOMM'01
- [5] "Description and analysis of the state-of-the-art", SCAMPI deliverable D0.1
- [6] Endace Measurement Systems, <http://www.endace.com>
- [7] G. Iannaccone, C. Diot, I. Graham, and N. McKeown, "Monitoring very high speed links," ACM Internet Measurement Workshop
- [8] Snort - Lightweight Intrusion Detection for Networks. Martin Roesch. USENIX LISA '99 Conference