

INFORMATION SOCIETY TECHNOLOGIES (IST) PROGRAMME



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

D3.2 “Experiment plans and infrastructure setup”

Abstract: This document provides the detailed description of all evaluation experiments to be performed as part of the workpackage WP3 “Experimental Evaluation”. The experiment description follows up the high-level description of experiment plans and infrastructure setup in deliverable D3.1.

Contractual Date of Delivery	28 March 2003
Actual Date of Delivery	
Deliverable Security Class	Internal
Editor	Sven Ubik
Contributors	IMEC, FORTH, LIACS, NETikos, UNINETT, CESNET, FORTHnet

The SCAMPI Consortium consists of:

TERENA	Coordinator	The Netherlands
IMEC	Principal Contractor	Belgium
FORTH	Principal Contractor	Greece
LIACS	Principal Contractor	The Netherlands
NETikos	Principal Contractor	Italy
UNINETT	Principal Contractor	Norway
CESNET	Principal Contractor	Czech Republic
FORTHnet	Principal Contractor	Greece
4Plus	Principal Contractor	Greece
Siemens	Principal Contractor	Germany

Contents

1	Introduction	4
2	Experiment description	4
2.1	Experiment goal	4
2.2	SUT and CUS	4
2.3	Evaluated service	4
2.4	Metrics	5
2.5	Parameters	5
2.6	Factors and levels	5
2.7	Evaluation technique	5
2.8	Setup and operation	5
2.9	Required equipment	5
2.10	Workload	5
3	Packet capture	6
3.1	Experiment goal	6
3.2	SUT and CUS	6
3.3	Evaluated service	6
3.4	Metrics	6
3.5	Parameters	6
3.6	Factors and levels	7
3.7	Evaluation technique	7
3.8	Setup and operation	7
3.9	Required equipment	9
3.10	Workload	9
4	Intrusion and DoS detection	10
4.1	Experiment goal	10
4.2	SUT and CUS	10
4.3	Evaluated service	10
4.4	Metrics	10
4.5	Parameters	10
4.6	Factors and levels	10
4.7	Evaluation technique	10
4.8	Setup and operation	12
4.9	Workload	12
5	QoS monitoring	13
5.1	Experiment goal	13
5.2	SUT and CUS	13
5.3	Evaluated service	13
5.4	Metrics	13
5.5	Parameters	13
5.6	Factors and levels	13
5.7	Evaluation technique	13
5.8	Setup and operation	14
5.9	Required equipment	14
5.10	Workload	15

6	Network flow statistics	16
6.1	Experiment goal	16
6.2	SUT and CUS	16
6.3	Evaluated service	16
6.4	Metrics	16
6.5	Parameters	16
6.6	Factors and levels	16
6.7	Evaluation technique	17
6.8	Setup and operation	17
6.9	Required equipment	17
6.10	Workload	17
7	MAPI, Monitoring Agent and Middleware	18
7.1	Experiment goal	18
7.2	SUT and CUS	18
7.3	Evaluated service	18
7.4	Metrics	18
7.5	Parameters	19
7.6	Factors and levels	19
7.7	Evaluation technique	19
7.8	Setup and operation	19
7.9	Required equipment	19
7.10	Workload	19

1 Introduction

The SCAMPI software architecture consists of several layers communicating with their respective interfaces. By applying test sequences to different interfaces we can evaluate components of the SCAMPI architecture at individual layers. Particularly, we will perform experiments evaluating the following components using the following interfaces:

- Monitoring applications, using their own interfaces (we will test four monitoring applications representing the most important classes of the use of the SCAMPI architecture, see sections 3 to 6)
- MAPI, Monitoring Agent and middleware, using testing sequences of MAPI calls, using the XML-based protocol between MAPI and the Monitoring Agent and using the calls to Middleware framework, such as Click.

2 Experiment description

We adopted a systematic performance evaluation technique proposed in [1]. The description of each experiment follows a template described in this section.

2.1 Experiment goal

Defines what should be the experiment result. For example, functionality of component X verified, processing overhead of monitoring activity Y determined, etc. In general, the goal of each experiment can be classified in one of the following categories:

- Compare alternatives - of underlying HW, that is the specialized adapter vs. commodity adapter vs. intelligent router vs. packet trace (obtained previously by any underlying HW)
- Do performance evaluation - find utilization of NP, host CPU, host RAM or disk for individual monitoring tasks or find limits in system or workload parameters for the correct operation of a particular monitoring activity or SCAMPI component
- Verify usability - check correctness of operation and estimate probability of incorrect response or failure

2.2 SUT and CUS

In general, for most experiments System Under Test (SUT) is the whole SCAMPI architecture, optionally including the monitoring application. System boundary is the adapter interface to the network and the user interface of the monitoring application or the interface to the lower layer of the SCAMPI software architecture.

In some cases, there is one component of SUT, called Component Under Study (CUS), whose alternatives should be evaluated. Usually, CUS will be the underlying HW (specialized adapter, commodity adapter or intelligent router).

2.3 Evaluated service

The service to be evaluated during the experiment. It is the goal of the particular monitoring application or the function of the lower layer of the SCAMPI software architecture. Services to be evaluated should include the most important monitoring applications developed as part of the project and representing different monitoring capabilities of the SCAMPI architecture.

2.4 Metrics

Numeric criteria used to assess performance of the system. For each service, three possible outcomes should be considered: correct response, incorrect response and failure (no response). To cover all possible outcomes, metrics should be related to time, request rate and resource utilization (for correct response), accuracy (classification of incorrect responses and their frequencies) and availability (classification of failure modes and their frequencies). The set of metrics should be complete, nonredundant and should exhibit low variability.

2.5 Parameters

Variables affecting performance as expressed by metrics. Can be divided to system parameters (e.g., CPU speed) and workload parameters (e.g., packet size).

2.6 Factors and levels

A subset of parameters to be studied during the experiment. Considered values (levels) of individual factors should also be specified. Based on the results obtained in the first evaluation pass, the set of factors and levels can be modified.

2.7 Evaluation technique

In general, the performance evaluation experiment can be conducted as measurement, simulation, analytical modeling or a combination of these. Due to the characteristics of the SCAMPI architecture, we decided to use measurements as a primary evaluation technique. Measurements will be performed both on a testbed in a laboratory and on a real network. Where appropriate, analytical modeling will be used to support measurement results.

2.8 Setup and operation

The detailed description of equipment setup to perform the experiment and actions to be taken during the course of the experiment.

2.9 Required equipment

What hardware, software and logistic measures (such as access to a real network) have to be obtained or taken to conduct the experiment.

2.10 Workload

Workload must represent use of the system in real operational conditions in terms of connection or request arrival rate, traffic or request characteristics, total resource demands and resource usage profile (sequence and the amounts at which resources are used).

Workload will be both synthetic and real-world. Synthetic workload will be generated by the SCAMPI architecture in the traffic generator mode and by other network traffic generators. Real-world workload will be taken from selected network connections in the Géant network and in the participating NRENs and the laboratory or operational networks of participating organisations.

3 Packet capture

3.1 Experiment goal

The goal of this experiment is to verify that the packet capture functionality of the SCAMPI architecture works properly and that it can operate at the full speed. The functionality verification should be done with all types of underlying hardware that is supported by the SCAMPI architecture. The performance test must be done with the final version of the specialized adapter, operating at the full speed of 10 Gb/s. For comparison purposes, the performance test can also be done with another hardware option supported.

3.2 SUT and CUS

SUT is the whole SCAMPI architecture. CUS is the underlying hardware.

3.3 Evaluated service

Traffic capture with the following possible outcomes:

Correct response Positive response from the Packet capture monitoring application or from the MAPI calls and captured exactly the required packets and stored exactly the required parts with timestamps of the required precision and accuracy.

Incorrect response Positive response from the Packet capture monitoring application or from the MAPI calls, but some of the required packets are not captured or some unwanted packets are captured or other parts than required are stored or does not provide timestamps of the required precision and accuracy.

Failure Negative response from the Packet capture monitoring application or from the MAPI calls.

3.4 Metrics

- For correct response: packets per second captured, utilization of NP, host CPU, host RAM and disk
- For incorrect response: frequency of error occurrence
- For failure: frequency of failure occurrence

3.5 Parameters

System parameters:

- speed of NP or FPGA
- speed of host CPU
- speed of host RAM
- speed of disk
- size of host RAM
- size of disk
- overhead of host OS
- type of underlying HW (specialized adapter, commodity adapter, intelligent router, packet trace)

Workload parameters:

- speed of network
- size and structure of monitoring requests
- medium packet size and distribution of packet sizes
- other load on NP, host CPU, host RAM and disk (for evaluation of single monitoring application) caused by other concurrent monitoring applications

3.6 Factors and levels

- speed of host PC (commodity PC, high-end PC)
- type of underlying HW (specialized adapter, commodity adapter, intelligent router, packet trace)
- speed of network (1000BASE-SX, 10GBASE-LR)
- number of monitoring tasks (single task, single task + a mixture of other “background” tasks)
- packet size (64 bytes, 1500 bytes, a mixture of different sizes)

3.7 Evaluation technique

Packet capture functionality can be tested by measurement using either the Packet capture monitoring application, if it is available at the time of evaluation, or a testing sequence of MAPI calls, if the complete Packet capture monitoring application is not yet available.

The initial functionality verification tests will be done in a laboratory using a packet generator, such as RUDE/CRUDE [3] or GenSyn [4]. Further functionality verification tests will be done with the real-network traffic on the selected circuits in the Géant network or in the participating NRENs.

The performance test will be probably performed only in a laboratory where we can use required high-rate data streams, which are currently not yet present in real networks.

3.8 Setup and operation

Both for laboratory tests and for real-network tests we will need to tap traffic on the monitored circuit. This can be done by inserting a switch configured for port mirroring or an optical splitter in the monitored circuit. Using a splitter is a simpler and less expensive solution which only slightly changes physical layer characteristics of the monitored circuits and does not change its link layer characteristics. However, if the change of physical layer characteristics (namely increasing the circuit loss) is not acceptable or if there already is a switch with the port mirroring capability installed in the monitored network, using a switch can be a preferred solution. For laboratory tests where interaction of the monitored traffic with the receiving host is not needed it is also possible to replace the receiving host with the monitoring host running the SCAMPI platform. In this case, no switch or splitter is needed. The configuration is illustrated in Fig. 1.

For the performance test we need to generate testing traffic streams at the final operating speed of 10 Gb/s.

One possibility could be to generate these streams directly using a 10 Gb/s network adapter. We are currently working with Intel on procuring 10GBASE-LR adapters for this purpose. This adapter would have to be installed in an appropriately powerful PC, particularly in terms of bus speed and RAM speed. The fastest PC bus currently available is PCI-X 1.0 64-bit/133 MHz, which provides bandwidth of 8 Gb/s. A 512 MHz version has been proposed in PCI-X 2.0 recommendation, which will provide bandwidth of 32 Gb/s. It is likely that the first chipsets supporting this bus will occur at the beginning of year 2004, with first mainboards coming some time during 2004, which is too late according to the

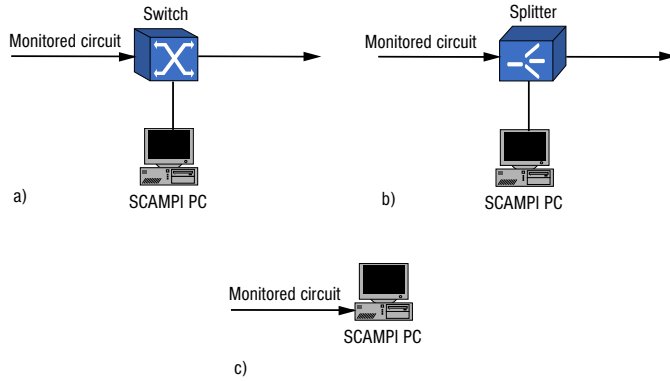


Figure 1: Inserting a SCAMPI platform in the monitored circuit: a) using a switch, b) using a splitter c) with a direct connection

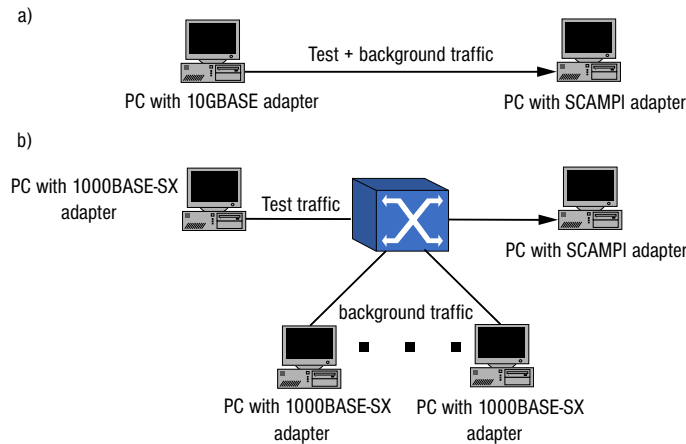


Figure 2: Packet capture stress test with a) 10GBASE-LR adapter b) 1000BASE-SX adapters

project schedule. A new standard PCI Express should deliver even higher speeds, but is likely to appear later. The speed of RAM and host CPU can also be a limiting factor. As we would generate the testing stream on-the-fly, the speed of disk would not be important in our case.

If sufficiently powerful PC for 10-Gigabit Ethernet adapters is not available in time to conduct the experiment, another possibility is to generate a slower testing stream, such as 1 Gb/s, and merge it with a set of background streams, each running at 1 Gb/s, using a switch with one 10 Gb/s port and multiple 1 Gb/s ports. A candidate switch for this purpose is Cisco Catalyst 6500 series with a 10-Gigabit Ethernet adapter. Cesnet plans to obtain two such switches for testing of long-distance optical circuits. The switches should be available in part for the evaluation in the course of the SCAMPI project. The configuration is illustrated in Fig. 2.

All streams (testing and possibly background) will be generated by a software application, such as RUDE tool modified to support required stream characteristics.

If the first pass of the experiment shows difficulty to generate streams at the required speed, we will use a commercial traffic generator. However, it seems that possibilities of specification of required traffic patterns are rather limited on these generators.

3.9 Required equipment

- packet splitter or a switch with mirroring capability
- packet generator, such as RUDE/CRUDE [3] or GenSyn [4]
- access to a real network or a 10-Gigabit Ethernet testbed
- underlying hardware of the SCAMPI architecture that is: a commodity network adapter (100BASE-TX or 1000BASE-TX/SX), an intelligent router (such as Juniper), a DAG card, the specialized adapter
- two PCs, one for a packet generator and one for the SCAMPI architecture
- either a high-performance PC capable to generate 10 Gb/s (if such a hardware becomes available in time to conduct the experiment, which seems to be unlikely case) or a set of PCs with Gigabit Ethernet adapters and a 10-Gigabit / 1-Gigabit ethernet switch, such as Cisco 6500 series Catalyst
- MAPI, Monitoring Agent and middleware
- Packet capture monitoring application or a testing sequence of MAPI calls

3.10 Workload

For laboratory tests, we will use defined flows produced by a packet generator, such as RUDE or GenSyn. For real network tests, we will use real user traffic on the monitored circuit.

For the performance test, we will also consider using a commercial packet generator. The aggregated traffic will consists of two parts: testing streams with defined patterns to be captured by the SCAMPI platform and background streams increasing the aggregated data rate to the level required for the stress test.

4 Intrusion and DoS detection

4.1 Experiment goal

The purpose of this experiment is to check that the SCAMPI architecture correctly detects required intrusion and DoS attempts in a supplied testing traffic.

4.2 SUT and CUS

SUT is the whole SCAMPI architecture. There is no CUS in a sense as defined in [1].

4.3 Evaluated service

Detection of intrusion or DoS attempts with the following possible outcomes:

Correct response Positive response from the Intrusion or DoS detection monitoring application or from the MAPI calls and exactly the required types of intrusion or DoS attempts detected in a supplied testing traffic.

Incorrect response Positive response from the Intrusion or DoS detection monitoring application or from the MAPI calls, but some of the required types of intrusion or DoS attempts are not detected or produces a false alarm.

Failure Negative response from the Intrusion or DoS detection monitoring application or from the MAPI calls.

4.4 Metrics

- For correct response: utilization of NP, host CPU (and optionally also CPU in an intelligent router), host RAM and disk
- For incorrect response: frequency of error occurrence
- For failure: frequency of failure occurrence

4.5 Parameters

Same as for packet capture evaluation described in section 3.

4.6 Factors and levels

- type of underlying HW (commodity adapter, intelligent router, packet trace)
- number of monitoring tasks (single task, single task + a mixture of other “background” tasks)

4.7 Evaluation technique

Detection of intrusion or DoS attempts can be performed by simulating an attack. There are two main ways to do this:

- Try to violate a system while the SCAMPI-based application is running.
- Use a packet trace from a real attack.

Live Violation Attempt A violation of a system is usually performed using an automatic tool that attempts to break a system using known vulnerabilities. For instance viruses and trojans violate a system by applying the same technique to all (most of) the hosts of the target network. Possible solutions are:

- Select a few trojans/worms/exploits and let them run on the network (rather dangerous).
- Use a security scanner (e.g. Nessus) to perform a penetration test against a host.

Below are listed a few security sites that contain information (and also source code) about exploits and worms:

- <http://neworderer.box.sk/>
- <http://www.incidents.org/>
- <http://www.securityfocus.com/>
- <http://www.nessus.org/>
- <http://www.insecure.com/splloits.html>
- <http://www.honeynet.org/>

Packet Trace Some sites provide packet traces (usually in tcpdump format) captures during a real attack. Using tools such as tcpreplay (<http://tcpreplay.sourceforge.net>) it is possible to inject this traffic onto the network simulating a real live attack.

In both cases, the idea is to write a MAPI-based application that is able to detect an intrusion. As there are thousand of known intrusions, this application should either:

- be restricted to a few of them
- or
- be a MAPI-porting of an open-source IDS that contains sensors for many intrusions.

In order to maximize the scope of this evaluation, the second option has been selected. Snort (<http://www.snort.org>), a widely used open source IDS, will be modified and ported on top of MAPI. Snort's architecture is divided in two main components: packet capture and signature detection. The Snort-MAPI application will inherit the signature detection capabilities from the original Snort, while fully replacing the packet capture code with MAPI.

With an IDS it's possible to detect only service-based DoS. For instance Snort can detect PoD (Packet of Death) that can crash an application making the associated service unavailable. Classic DoS such as mail bombing, or applications (e.g., DDoS applications) that saturate the network, cannot be detected by Snort unless they send packets with a precise signature. For this purpose a new MAPI-based application should be developed. This application should maintain a list of:

- top senders/receivers
- TCP sessions (attempted and established)
- peers that a host has contacted

Given that ntop is already providing the above information, it could be enough to port ntop on top of MAPI.

4.8 Setup and operation

Once the code has been developed, we need a test environment. The easiest setup is identify:

- a PC used as target (guinea pig).
- a PC that will run MAPI-based versions of snort and ntop deployed in a place where it can see the traffic from/to the target PC. This PC will save to disk in raw format the captured packets.
- a PC (attacker) that will run nessus and other trojans

The network speed is not really relevant for these tests as they are functional tests. If the test outcome is positive, we can repeat the test in order to verify the overall performance. In this case, the attacker will non use trojans but the captured packets that will be injected at full speed using tcpreplay (<http://tcpreplay.sourceforge.net>). This solution allows to reproduce the attacks at the maximum speed in a fully automatic way with no human intervention.

4.9 Workload

For laboratory tests, we will generate various testing packets using the tools above mentioned. For real network tests, we will need to install the test applications on a real network. For instance the ntop test plant is based at the University of Pisa uses a Juniper M20 switch to mirror all the campus traffic towards the Linux PC where ntop runs.

5 QoS monitoring

5.1 Experiment goal

The purpose of this experiment is to check that the SCAMPI architecture can monitor required QoS characteristics with the required precision.

5.2 SUT and CUS

The whole SCAMPI architecture can be considered both SUT and CUS.

5.3 Evaluated service

QoS monitoring with the following possible outcomes:

Correct response Positive response from the QoS monitoring application and provides all required network QoS characteristics for the specified flow with the required precision and accuracy.

Incorrect response Positive response from the QoS monitoring application, but does not provide some of the required network QoS characteristics or the required precision or accuracy is not met.

Failure Negative response from the QoS monitoring application or from the MAPI calls.

5.4 Metrics

- For correct response: achieved precision, accuracy, number of bytes sent (for active monitoring) and utilization of NP, host CPU, host RAM and disk
- For incorrect response: frequency of error occurrence
- For failure: frequency of failure occurrence

5.5 Parameters

Same as for packet capture evaluation described in section 3.

5.6 Factors and levels

- type of underlying HW (specialized adapter or commodity adapter)
- number of monitoring tasks (single task, single task + a mixture of other “background” tasks)
- packet size (64 bytes, 1500 bytes, a mixture of different sizes)

5.7 Evaluation technique

Primary network QoS characteristics are one-way delay, one-way packet loss delay variation and throughput. All characteristics can be measured by passive or active monitoring. One-way delay measurement requires precise time synchronization between the monitoring points. We will use GPS receivers for this purpose. For laboratory tests, where both monitoring points are located within the same room or building, we can use one GPS receiver with a distribution unit sending PPS signal to serial ports of both monitoring PCs. We do not plan to measure throughput actively, there are other tools for this purpose. Otherwise we will decide whether to use active or passive monitoring depending on what will be more appropriate for the particular network we will use as a testbed. For active monitoring, we will use one SCAMPI platform as a packet generator (alternatively, other packet generators can be used for comparison) and the other as a packet analyser. For passive monitoring, we will use two SCAMPI platforms

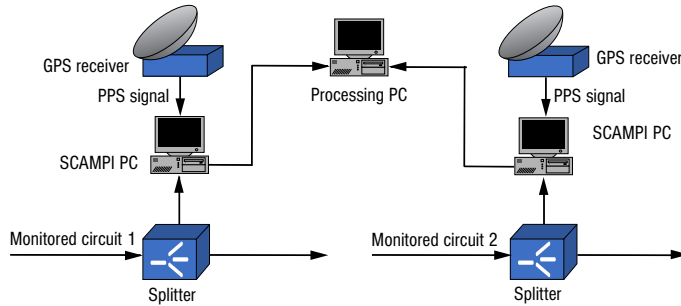


Figure 3: QoS monitoring configuration

as packet analysers. Their measurements can be processed on a monitoring station correlating packets captured by the two monitoring points. Alternatively, the functionality of the remote MAPI can be used where appropriate. For laboratory tests, where both monitoring points are located within the same room or building, we can use one GPS receiver with a distribution unit sending PPS signal to serial ports of both monitoring PCs.

Advanced performance characteristics, for example, for TCP behaviour analysis, such as detection of lost and duplicate TCP segments and monitoring of TCP windows development (rwnd advertised by the receiver and owin used by the sender) can be obtained from one monitoring point only.

At least a minimal QoS monitoring application must be developed prior to starting QoS monitoring evaluation.

5.8 Setup and operation

The initial tests will be done in a laboratory using NIST Net emulator [5] and RUDE/CRUDE [3] and GenSyn [4] packet generators. Further tests will be done over the Géant network or over some of the participating NRENs.

The same technique to tap traffic on a monitored circuit as described in section 3 will be used.

The testing configuration for two-point measurement is illustrated in Fig. 3.

5.9 Required equipment

- two packet splitters or switches with mirroring capability
- access to a real network
- two pieces of the underlying hardware of the SCAMPI architecture that is: a commodity network adapter (100BASE-TX or 1000BASE-TX/SX), an intelligent router (such as Juniper), a specialized adapter
- three PCs: two for the SCAMPI architecture in two measurement points and one for correlating and processing results from the measurement points
- two GPS receivers (one is sufficient for laboratory tests where both measurement points are in one place)
- MAPI, Monitoring Agent and middleware
- at least a minimal QoS monitoring application including a program for correlating and processing results from the measurement points

5.10 Workload

For laboratory tests, we will generate various testing streams using RUDE and GenSyn and use emulation of various QoS characteristics in NIST Net.

For real network tests, we will observe QoS characteristics of flows generated from network performance testing applications, such as iperf and of regular user traffic.

6 Network flow statistics

6.1 Experiment goal

The goal of this experiment is to check the network flow monitoring functionality of the SCAMPI platform and to check that the reported network flow statistics correspond to the characteristics of the monitored network traffic.

The experiment should verify functionality needed for two envisioned monitoring applications: netflow probe and flow-based reporting. The former will just produce records about individual flows detected on a monitored circuit. The latter will produce detailed flow characteristics about the specified flows, such as octets per flow distribution or ToS value distribution.

6.2 SUT and CUS

The whole SCAMPI architecture can be considered both SUT and CUS.

6.3 Evaluated service

Getting the network flow statistics with the following possible outcomes:

Correct response Positive response from the Netflow monitoring application or from the MAPI calls and retrieved network flow statistics corresponding to the characteristics of the monitored network traffic.

Incorrect response Positive response from the Netflow monitoring application or from the MAPI calls and retrieved network flow statistics that do not exactly correspond to the characteristics of the monitored network traffic (including the case of no network flow statistics retrieved).

Failure Negative response from the Netflow monitoring application or from the MAPI calls.

6.4 Metrics

- For correct response: flows identified, utilization of NP, host CPU (and optionally also CPU in an intelligent router), host RAM and disk
- For incorrect response: frequency of error occurrence
- For failure: frequency of failure occurrence

6.5 Parameters

Same as for packet capture evaluation described in section 3.

6.6 Factors and levels

- type of underlying HW (specialized adapter or commodity adapter)
- number of monitoring tasks (single task, single task + a mixture of other “background” tasks)
- packet size (64 bytes, 1500 bytes, a mixture of different sizes)
- number of flows in a network (low number for functional tests, high number for stress tests)

6.7 Evaluation technique

The initial tests will be done in a laboratory to check netflow monitoring functionality. Further tests will be done on a real network to check the output produced by the SCAMPI platform against the netflow information produced by GTDMS II [6] netflow monitoring application which gathers data about flows from routers.

The network flow statistics can be either taken from the Netflow monitoring application, if it is available at the time of evaluation, or obtained using a testing sequence of MAPI calls, if the complete Netflow monitoring application is not yet available.

6.8 Setup and operation

The same technique to tap traffic on a monitored circuit as described in section 3 will be used.

A testing sequence of MAPI calls to dynamically generate flows will be used.

6.9 Required equipment

- packet splitter or a switch with mirroring capability
- packet generator producing testing flows
- access to a real network
- underlying hardware of the SCAMPI architecture that is: a commodity network adapter (100BASE-TX or 1000BASE-TX/SX), an intelligent router (such as Juniper), a specialized adapter
- two PCs, one for a packet generator and one for the SCAMPI architecture
- MAPI, Monitoring Agent and middleware
- Netflow monitoring application or a testing sequence of MAPI calls

6.10 Workload

For laboratory tests, we will use a defined mixture of flows produced by a packet generator, such as RUDE or GenSyn. For real network tests, we will use real user traffic on a monitored circuit.

7 MAPI, Monitoring Agent and Middleware

7.1 Experiment goal

The purpose of this experiment is to check that the implementation of MAPI, Monitoring Agent and middleware works properly. This includes testing of MAPI itself, Module Organizer, installation of filters, active monitoring specification, resource control performance and middleware tests, such as verification of proper function of individual Click elements.

7.2 SUT and CUS

SUT is the implementation of MAPI, Monitoring Agent and middleware + underlying hardware or packet traces. CUS is a particular software part such as Module Organizer or Click elements.

7.3 Evaluated service

The evaluated service depends on the particular component being tested.

For example, when testing the MAPI implementation, the evaluated services and its possible outcomes are as follows:

Correct response Positive response from MAPI after submitting the calls, the correct graph of modules is requested from the Monitoring Agent

Incorrect response Positive response from MAPI after submitting the calls, but incorrect or no graph of modules is requested from the Monitoring Agent

Failure Negative response from MAPI, calls are not accepted

When testing the Monitoring Agent:

Correct response Positive response from the Monitoring Agent after submitting the calls, the correct graph of modules is requested from middleware, such as the Click framework

Incorrect response Positive response from the Monitoring Agent after submitting the calls, but incorrect or no graph of modules is requested from middleware, such as the Click framework

Failure Negative response from the Monitoring Agent, calls are not accepted

Or, when testing Click modules:

Correct response Positive response to calls to `/proc` interface, click elements are instantiated as requested

Incorrect response Positive response to calls to `/proc` interface, but click elements are not instantiated as requested

Failure Negative response to calls to `/proc` interface, calls are not accepted

7.4 Metrics

- For correct response: for example, number of modules requested from the Monitoring agent as a result of module optimization when testing MAPI and Module Organizer or response time when testing resource control performance. For some tested components, such as when testing the Monitoring Agent or Click modules, it is difficult to measure the level of success in the case of correct response. Therefore, we will just register if the response was correct or not.
- For incorrect response: frequency of error occurrence
- For failure: frequency of failure occurrence

7.5 Parameters

System parameters:

- level of resource control enforcement

Workload parameters:

- number, types and arguments of calls to MAPI, Monitoring Agent or middleware
- calls to MAPI, Monitoring Agent or middleware submitted by other concurrent monitoring applications

7.6 Factors and levels

- number, types and arguments calls to MAPI, Monitoring Agent or middleware
- number of call sequences (testing only vs. testing + “background” sequence)
- resource control state (on, off)

7.7 Evaluation technique

This experiment will be conducted only in a laboratory. It will be one of the firsts experiments which should be done prior we start to develop monitoring applications.

We will use testing sequences of calls to MAPI, Monitoring Agent or middleware, such as the Click framework to verify software functionality. Result of each call must be carefully examined. The testing environment is illustrated in Fig. 4. When submitting calls to MAPI, the produced graph of modules can be checked observing messages of the XML-based protocol between MAPI and the Monitoring Agent. When submitting testing sequences to the Monitoring agent the resulting graph of modules can be checked by observing calls to middleware, such as through the `/proc` interface to the Click framework. And when testing the Click modules, the configuration must be specified in the router configuration language and written to `/proc/click/config` file. Errors should be checked in `/proc/click/errors` file. Other relevant `/proc` filesystem files will also be checked, such as `/proc/click/list`, `/proc/click/meminfo` and individual element files in `/proc/click/xxx` directories.

7.8 Setup and operation

The test will be conducted on one laboratory PC. Software test can be performed with a commodity network adapter, the specialized adapter is not needed. However, if also lower-layer hardware-dependant software is to be tested, all types of underlying hardware supported by the SCAMPI architecture have to be available. No further network infrastructure is required.

7.9 Required equipment

- one PC with a commodity network adapter
- MAPI and the Monitoring Agent
- optionally other types of the underlying hardware that is an intelligent router (such as Juniper, in addition to a commodity network adapter), a specialized adapter

7.10 Workload

Example testing sequence of calls to MAPI, Monitoring Agent and Click framework will be used.

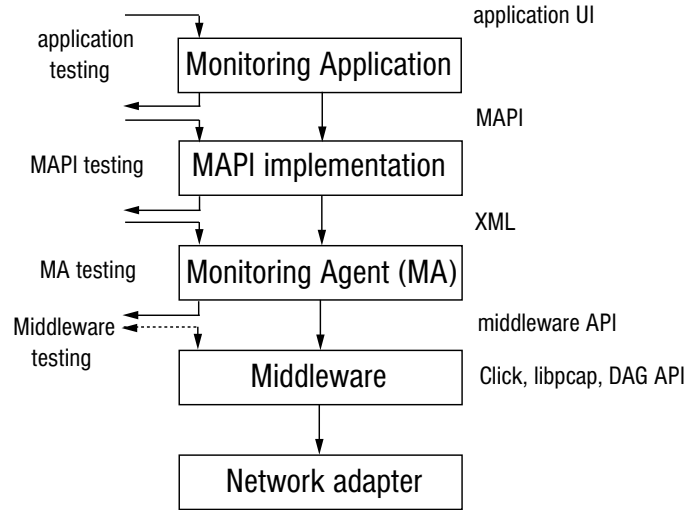


Figure 4: Testing individual software components

References

- [1] Raj Jain. *The Art Of Computer System Performance Analysis*, Wiley, 1991.
- [2] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, M. Frans Kaashoek. *The Click Modular Router*, <http://www.pdos.lcs.mit.edu/click>.
- [3] *RUDE/CRUDE utilities*, <http://rude.sourceforge.net>.
- [4] Poul E. Heegaard. *GenSyn - A generator of synthetic Internet traffic used in QoS experiments*, 15th Nordic Teletraffic Seminar, August 2000, Lund, Sweden, <http://www.item.ntnu.no/poulh/GenSyn/gensyn.html>.
- [5] *NIST Net network emulation package*, NIST Internetworking Technology Group (ITG), <http://snad.ncsl.nist.gov/itg/nistnet>.
- [6] Tom Košnar. *GTDMS II - Grafical and Textual Data Measurement System*, CESNET, 1999-2002.