An approach to a methodology and implementation of a Network Monitoring Analysis Tool

Elias Aravantinos

Dr. Petros Ganos

Aristidis Ilias
Research Academic Computer Technology Institute

Riga Feraiou 61

GR- 26221 Patras, Greece
E-mail: eliasara@cti.gr
E-mail: ganos@cti.gr
E-mail: ilsadis@cti.gr

Dr. Christos Bouras
Computer Engineering and Informatics Department
University of Patras
GR-26500, Patras, Greece

Research Academic Computer Technology Institute
E-mail: bouras@cti.gr

KEYWORDS
Network monitoring, IP Based Networks, SNMP, Real Time System, Network resources, Databases.

ABSTRACT

In this paper we describe the design and implementation of a network monitoring analysis tool. The network resources were configured to support SNMP under various operating systems, thresholds definitions and events. The data were collected by a network monitoring system and handled according to the values of the variables or the event type. A real- time report to a database was established via adaptive scripts. A Network Node Manager (NNM) will inform an external, remote database about the network status by sending real time data containing alerts and events. The whole tool is called Network Management Analysis Tool (NMAT).

INTRODUCTION

A fundamental division among network monitoring systems is related to whether the monitoring is done off-line (while only test messages are flowing), on-line (while user traffic is on the network) or both. Techniques for monitoring networks, when they are out of service, involve the generation of test sequences, monitoring and analysis of the results, to determine measurable levels of performance and protocol conformance. Such tests may be implemented in a test bench environment, manufacturing floor, network operations centre or during an outage in an operational network.
The process of on-line monitoring and analysis may be performed on a continuous basis, scheduled at various times of day or invoked only when circumstances (such as load changes or quality concerns) demand a closer scrutiny. Continuous monitoring (intrusive testing) is the most costly alternative in terms of resources and time, but is also the most beneficial from a user's viewpoint.
Monitoring may be divided into two categories according to the layers involved:

- Monitoring and analysis of the physical and network layers referring to the OSI model
- Monitoring and analysis of the hardware platforms and operating systems

A very large number of access points (network nodes) may be required and they would usually be dedicated to the monitoring system. The amount of data that needs to be collected and transported to a local/remote database may also be very high. The benefit, however, is that proactive management and dynamic prediction may be a reality.
Network performance measurement is an extremely broad area and we can only briefly mention some of the more relevant work. (Carter and Crovella 1996) present two tools "bprobe" and "cprobe" to measure the bottleneck link speed and competing traffic respectively, on a path using ICMP ECHO packets. Since these tools do not use TCP, they are not able to capture any TCP related effects that an application might experience. Van Jacobson proposes the tool "pathchar" (Jacobson 1997) that estimates bandwidth on all hops of a path and hence can take a very long time. It also requires root access, making it less desirable for grid environments. The tool "Treno" (Mathis amd Mahdavi 1996) emulates an idealized TCP, which makes the measurements independent of host-specific TCP implementations but not representative of what applications would experience. "Treno" also needs root access. Topology-d (Obraczka 1998) uses "ping" and "netperf" to make measurements between all pairs within a group and then computes a minimum-cost logical topology. The Network Weather Service (NWS) (Wolski 1998) uses TCP to send small, fixed-size probes measured in kilobytes, with a frequency that is tuneable, but typically ranges from tens of seconds to several minutes. Performance measurement systems, such as the National Internet Measurement Infrastructure (NIMI) project (Paxson 1998) are designed for arbitrary Internet hosts. The Cooperative Association complements this work for Internet Data Analysis. The goal is to develop metrics and tools for analyzing traffic across the Internet. Additionally, the data from such tools intend to be used by "higher-level" systems. Lowecamp (Lowecamp 1998) also provides an API whereby applications can pose flow and topology-based queries.

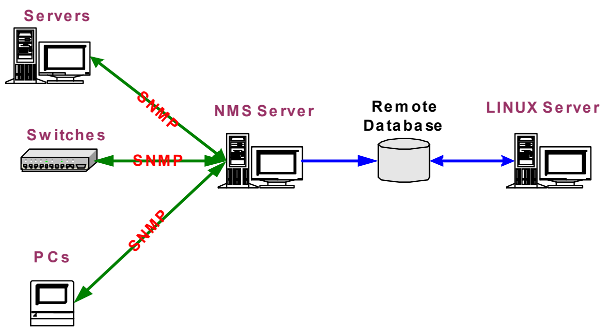
AppLeS (Application Level Scheduler) (Su 1998) uses information from the NWS to schedule distributed resource-intensive applications.
Finally other components will use “Gloperf” information in similar resource discovery and allocation functions. We note (Lee 1999) that “Gloperf” is designed to enhance portability. It makes end-to-end TCP measurements. Storing “Gloperf” data in a directory service provides data discovery and access. However, there are missing some measurements to allow applications to probe network resources and getting fresh data.
The purpose of this paper is to approach a methodology and implementation of a Network Monitoring Analysis Tool (NMAT) during the operation of an intranet network containing hundreds of terminals established in enterprise buildings, airplanes, ships etc. The aim is to collect automatically and in real-time values of monitoring data in a storing schema, a database for instance and finally make decisions at a human level.
This paper is organized as follows: First the tool design is presented in main principles. The next section describes the proposed phases of functionalities, containing techniques, specifications and results of network monitoring. Finally future work and concluding remarks are provided.

ARCHITECTURE

The Network Management System (NMS) used, is HP Open View Network Node Manager (NNM). It is a commercial, well-known and reliable platform that implements the Simple Network Management Protocol (SNMP). The NNM collects by polling in tuneable time intervals the SNMP data from the managed network devices of the intranet and automatically sends the required values of data to a database.
The tables are filled by these values, which could be exported in various metadata formats in order to realize a data statistical analysis. Then the analysis will be assessed and will also be the input to form the predictions of network behaviour and performance. The last step concerns decisions and reconfiguration processes.
The whole process is described as following:

- The NMS is fully configured.
- The general database is created (installation, table creation).
- The network resources are configured to support SNMP (agent) under Linux and Windows operating system.
- Thresholds are defined and events are configured at the NMS.
- Data are collected by the NMS and handled according to the values of the variables or the event type.
- Real-time report is imported to the database via a script language.
- The events are classified by an appropriate value for later data manipulation and analysis.
- Prediction and decision analysis about the network components performance are made.

The tested network is an Intranet, supported by a Fast Ethernet network with bus topology. The Fast Ethernet network cable is Unshielded Twisted Pair category 5 (UTP cat 5). The tested network consists of a certain number of terminals, servers, Local Area Network (LAN) router and LAN switches. Each LAN segment is located in a different physical place in the Intranet. The LAN switches are Fast Ethernet network devices that provide each sender/receiver pair with full 100 Mbps capacity. Each port on the switch gives full bandwidth to a single server or client station.
Furthermore, there is a switch-router connected to the Internet to serve the Intranet. The terminals and the servers are connected to different LAN segments. It was supposed that the tested terminals are located in different physical LANs instead of VLANs. All the terminals are connected to different switches instead of VLANs of the same switch.



Figures 1: System Architecture

Figure 1 describes the tested system architecture. The network monitoring process was based on the SNMP, managing all devices, like servers, switches and PCs. The Remote Database was hosted on a LINUX Server and NMS established a one way communication link, each time that appeared the need of sending data to the database.
The NMS server supported Windows 2000 Server operating system and the terminals Windows 2000 and LINUX.
An SNMP agent resides in each terminal machine in order to be remotely managed by NMS. The managed devices run software (SNMP agent) that enables them to send alerts (traps) when they recognize problems (for example, SNMP link failure). Upon receiving these alerts, management entities are programmed to react by executing one, several or a group of actions (referring to scripts) including event logging, changes in device configuration and system shutdown.
The management entities poll the end stations to check the values of certain variables in Management Information Bases (MIB). A MIB is a SNMP structure that describes the particular device being monitored. Polling can be automatic or user-initiated and agents in the managed devices respond to all polls. Agents are software modules that first compile information about the managed devices in which they reside, then store this information in a management database and finally provide it (proactively or reactively) to management entities within NNM via SNMP.

IMPLEMENTATION ISSUES

In our tested network an external database is used instead of an internal NNM database. The reason is to create applicable tables with different relations compared to the compact, complicated and inflexible NNM’s database model. As a result we propose a scheme, which could appear database

platform independency. Moreover the use of an external database provides interoperability, because it is possible to comply with several software tools, like JAVA-based platforms or other experimental tools developed for similar purposes.
The NNM collects by polling in tuneable time intervals the SNMP data from the managed network devices of the intranet and automatically sends the required values of data to a database.
Any corporate network has two types of end users: the organization's typical employee and the network operator who is responsible for maintaining the end-to-end service. The organization's typical end users and the success of the business applications that live on the network are the ultimate beneficiaries of all the various network monitoring activities that are performed within the network. However, the end users are least aware of the infrastructure behind the service. They simply need a quality of service that meets their business needs and achieves certain invisibility. It is the goal of achieving this "invisibility" of consistently good service that makes the added cost of monitoring and analysis tools justifiable.
The MIBs are used to monitor and access network and system information variables. They are either vendor MIBs or experimental MIBs. The vendor MIBs concern mainly the network switches while the experimental ones are used to monitor special parameters like the hostmib.mib which monitors system information (for example CPU load).
The SNMP polling interval was generally configured at 60 seconds. The NMS polls each critical variable every 10 seconds receiving traps and events. This polling interval concerns mostly the servers and the switches of the Intranet. The rest of the nodes are controlled from NNM every 1 minute.
The Network Management Analysis Tool (NMAT) reports all events in the external database with the help of two script-files (Brown 1999) (Events.ovpl and Events_value.ovpl) developed in Perl scripting language. These files are useful for the automatic actions, in order to insert the network variables values polled from NNM into the Database. Both scripts are running with the help of Open View Perl installed with NNM and configured with the help of the Event Configuration module of NNM.
The main problem we had to solve was the handle of consequential events, which are events that occurred exactly the same time. After making several experiments, we noticed that there was a unique variable generated by every event. This variable helped to distinguish the consequent events as discrete instants and store them into a remote database.
The events are separated in two categories:

- Events with value
- Events without a measured value

Events with value

The configuration of the events that do contain value should include the following automatic action command:
OVHIDESHELL cmd.exe /c "Events_value.ovpl $s $ar $N $3 $8"

Through this command the Perl file is called and rest of the symbols present the specific event variables based on NNM's manual.

```
#!/opt/OV/bin/Perl/bin/perl
my $param_1 = $ARGV[0];
my $param_2 = $ARGV[1];
my $param_3 = $ARGV[2];
my $param_5 = $ARGV[4];
my $param_4 = 100;
my $param_7 = 100;
my
$SUMMARY_TEXT="tmp_event_".$param_5.".sql"
;
open(SUMMARY,">$SUMMARY_TEXT");
$command1 = "insert into nms(severity,source,event,value)
values ('$param_1', '$param_2', '$param_3', '$param_4');";
printf SUMMARY ("$command1");
close(SUMMARY);
$command2 = "psql.exe -h 150.140.21.70 -f
$SUMMARY_TEXT system nms";
$command3 = "del $SUMMARY_TEXT";
system($command2);
system($command3);
exit 0;
```

Figures 2: Listing of Events_value.ovpl

Events without a measured value

The configuration of the events that do not contain value should include the following automatic action command:
OVHIDESHELL cmd.exe /c "events.ovpl $s $ar $N 100 $3"

```
#!/opt/OV/bin/Perl/bin/perl
my $param_1 = $ARGV[0];
my $param_2 = $ARGV[1];
my $param_3 = $ARGV[2];
my $param_4 = $ARGV[3];
my $param_5 = $ARGV[4];
my
$SUMMARY_TEXT="tmp_event_".$param_4.".sql"
;
open(SUMMARY,">$SUMMARY_TEXT");
$command1 = "insert into nms(severity,source,event,value)
values ('$param_1', '$param_2', '$param_3', '$param_5');";
printf SUMMARY ("$command1");
close(SUMMARY);
$command2 = "psql.exe -h 150.140.21.70 -f
$SUMMARY_TEXT system nms";
$command3 = "del $SUMMARY_TEXT";
system($command2);
system($command3);
exit 0;
```

Figures 3: Listing of Events.ovpl

The scripts use some NNM variables that are generated with the event, open a connection with the Postgres database using

the “psql” client and insert data to the NMS table. By the time of an event, a trap occurs, an automatic action occurs, the appropriate script is executed reporting the status of the nodes (up or down) and certain data are imported to the NMS table of the database. The data could be event description, severity of the event (normal, major) and special values such as CPU load and temperature. When the event does not contain value like node status events, Events.ovpl is running. When the event does contain value like CPU load Events_value.ovpl is running. Additionally, PSQL client helps to connect to the database by using the IP of the LINUX server instead of the DNS to avoid a possible failure of DNS server.
Then, the events should be activated (this is an easy procedure). Once the configuration is created, everything is included in the trapd.conf file. The events and the specifications are:

Table 1: Events Configuration

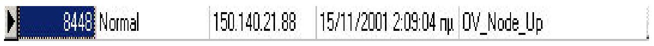
Specifications	**Name of the event**
Polled by the SNMP manager every 60 sec	OV_NODE_UP OV_NODE_DOWN
Checked on all machines every 60 sec OVER the limit in case load > 70% REARM the limit in case load < 60%	OV_CPU_OVER OV_CPU_REARM
OVER if T° > 41°C REARM if T° <= 40°C	OV_TEMP_OVER OV_TEMP_REARM
Link is DOWN if connection of cable is removed	SNMP_LINK_UP SNMP_LINK_DOWN

The thresholds concern the control of CPU load (OVER in case load > 70%, REARM in case load < 60%) of all monitored nodes and also the control of CPU temperature (OVER if T° > 41°C, REARM if T° <= 40°C) at terminals side. When the threshold value is exceeded, NNM reports to the database through an automatic action the new value. Everything is included in the SnmpCol.conf file.
In order to keep NMS CPU level in low level, a task is added in Windows task manager that restarts the “SNMPcollect” procedure every 5 minutes, running cpuNMS.bat file. This task is necessary because the CPU is overloaded due to the short SNMP polling intervals. The suggested regular NNM polling interval from HP is 5 min.
The external database could be Postgres, MySQL etc. The database management system selected was a Postgres, hosted on a LINUX server. The configuration of DBMS concerned only trusted “users”, specifically only the IPs of NMS server represented a trusted machine, which could communicate with the appropriate port of the LINUX server and with the database. The above option was selected to ensure the security of database and its data.
NNM is able to send and store data to the central database. The interface uses the Postgres client to connect to the database and some SQL commands to insert data into the NMS table. All the modules of this interface are stored in a temporary folder of hard disk.

Table 2: NMS Table Architecture

ID	Event Identifier
Event	Short description
Source	The IP Address of the Node
Severity	Event Importance
Value	Variable value of measured events

A different script is executed according to the type of the event that occurs, in order to store CPU load, node status, temperature etc. into the database table. The script is embedded in the Events configuration module and exports several variable values of NNM. All actions are automated and occur in real time. The following table shows a sample record in the NMS table:



8448	Normal	150.140.21.88	15/11/2001 2:09:04 μμ	OV_Node_Up

Figures 4: Sample record

The first field is an auto-counter, next the severity importance, next the source indicating the IP address, next a field that contains the local date-time and last a field that contains a short description of the event. Postgres generates automatically the field date-time. In the field “Value” is stored data from measured events like temperature and CPU load.
Other variables that NNM monitors and measures are:
- Interface Utilization of TCP/IP port
- Interface errors
- Bandwidth
- Hardware fails of the monitored devices
- CPU load
- Temperature value of the terminal
- Node status
- Thresholds to critical values
- SNMP traffic, etc.

Table 3 presents some samples of database records during a certain time period of about three months. It is obvious that the desired network and system variables were collected according to the scenario. The Event with ID 11298 that was stored in the table referred to a very important, major situation affecting the CPU of the machine with IP 150.140.21.63. The CPU load of the machine at the specific date time was over the specific limit, about 80%. Then the Event with ID 11302 refers to a situation where the CPU operates normally according to pre-configured limits. The network administrator checks the table and finds out which are the ‘weak’ points of the network. Then he makes decisions about network and system improvements, in order to avoid some similar situations or combination that already occurred. The main and critical issue is to take decisions about the whole information system containing network expansion, bandwidth issues, devices with hardware and software specifications etc. concluding sometimes to a brand new system ‘refresh’, after taking into account the different parameters.

Table 3: Sample of experimental results

ID	EVENT	SOURCE	SEVERITY	VALUE	DATE AND TIME
8356	OV_Node_Up	150.140.21.39	Normal		27/08/2002 01:34:32
11298	OV_Cpu_Over	150.140.21.63	Major	80	17/10/2002 01:41:15
11299	OV_Node_Down	150.140.21.30	Normal		17/10/2002 01:41:43
11302	OV_Cpu_Rearm	150.140.21.63	Normal	20	17/10/2002 01:51:01
13825	SNMP_Link_Down	150.140.21.38	Major		20/11/2002 10:35:08

FUTURE WORK

The future work concerns the extension of the monitored SNMP variables in order to cover all the possible network events. The final expectation is to focus on the real network status, show its weak points and improve the network performance.
The prediction and decision analysis of the data collection could become another issue of our future work. We intend to select some standard models related to data analysis and do further work close to this problem. This work will enhance the decision analysis and prediction parts of our tool.
Another future goal is to apply this tool to several networks and assess the variables based on their criticality.

CONCLUSIONS

For network operators, network monitoring and analysis provides the means to become proactive (i.e. to detect faults prior to receiving a user's complaint). It also allows them to manage service level contracts, to be assured of day-to-day operations and to validate system changes.
The result of our work is a methodology and an implementation of a network monitoring analysis tool, which could improve the network performance. This could be achieved via the prediction and decision analysis of the data collection.

REFERENCES

Brown, M. C., 1999, "The Complete Reference Perl", Osborne/McGraw-Hill.

Carter, R. and M. Crovella. 1996. "Dynamic server selection using bandwidth probing in wide-area networks", *Technical report*, *Boston U.*, TR-96-007.

Carter, R. and M. Crovella. 1996. "Measuring bottleneck link speed in packet-switched networks", *Technical report*, *Boston U.*, TR-96-006.

Jacobson, V. 1997. "A tool to infer characteristics of Internet paths", *Technical report*, Lawrence Berkeley Lab.

Lee, C. A., J. Stepanek, R. Wolski, C. Kesselman and I. Foster, November 1999, "A Network Performance Tool for Grid Environments", *Technical Paper presented at Super Computing '99*, Portland, Oregon, USA.

Lowecamp, B. et al., August 1998, "A resource query interface for network-aware applications", *7th IEEE Symposium on High Performance Distributed Computing*, pages 189–196.

Mathis, M. and J. Mahdavi, 1996, "Diagnosing Internet congestion with a transport layer performance tool", *Proc. INET '96*.

Obraczka, K. and G. Gheorghiu. August 1998. "The performance of a service for network-aware applications", *2nd Sigmetrics conference on parallel and distributed tools*.

Paxson, V., J. Mahdavi, A. Adams, and M. Mathis. August 1998. "An architecture for large-scale internet measurement", *IEEE Communications*, 36(8): 48–54.

Su, A., F. Berman, R. Wolski and M. M. Strout, November 1998, "Using apples to schedule a distributed visualization on the computational grid".

Wolski, R., N. Spring, and H. Hayes. 1998. "The network weather service: A distributed resource performance forecasting service for meta-computing", *Future Generation Computing Systems*.