STUDIA UNIV. BABEŞ-BOLYAI, INFORMATICA, VOLUME II, NUMBER 1, 1997

# DISTRIBUTED NETWORK MONITORING AND MANAGEMENT

DRAGOS POP SIMONA IURIAN
MARIUS IURIAN CRISTIAN MIHOC

**Abstract.** In order to accomplish their tasks the network managers need a network monitoring and management application which allow them to start and stop network components as needed, to monitor the network, and to extract and analyze information related to network traffic and network performance. NetMon is a distributed framework, specifically designed for the network monitoring and management. The newest version of NetMon extended the support for SNMP in order to be able to interact to critical server applications like Oracle Server or Microsoft SQL Server 6.5. One of the most important features added was the distribution of tasks so that NetMon provides also daemons applications for specific platphorms like Lotus Notes server for Windows NT.

## 1. Introduction

NetMon have had three different stages in its evolution over time:

1. The creation of the kernel of NetMon based on the fundamental notions of Console, Event and Alert and the support for the creation of automated procedures of management and monitoring. In this phase NetMon supported a few basic network protocols for communication and management [Iurian 1995]

2. The second phase was to extent the network protocols as it was the case of IPX/SPX and even the possibility to interact in a native way with Novell Netware servers. Also in this phase it was inserted the support for RMON in order to have greater flexibility and the possibility to manage wider networks and internetworks. The RMON was designed with the following goals in mind:

- Off-line operation, it means that the manager must poll the monitor asynchronously
- Preemptive monitoring,
- Problem detection and reporting,
- Value-added data, which means that the monitor use the collected data in specific analyses,
- Multiple managers.

3. Until this phase NetMon relied in every case on predefined mechanisms existent on the managed servers or components - a Telnet server, an SNMP agent or server, a

Received by the editors: October 30, 1997.
*1991 Mathematics Subject Classification.*68M10, 68M20, 68N15.
*1991 CR Categories and Descriptors.*C.2.3 **[Computer-Communication Networks]**: Network Operations - *Network management, Network monitoring*; C.2.4 **[Computer-Communication Networks]** Distributed Systems - *Distributed applications*

Novell Netware server. The newest version achieve distribution of services over the network because it provide daemons (specialized servers) for applications for which a finer tunned management was necessary. This extends the distribution of services which was first only amongst NetMon stations as presented in figure 1.

All these features make NetMon compliant with ISO (International Organization for Standardization) who defined, as part of its specifications of OSI systems management, the functional requirements of a network management application [Stallings 1993]:

| | |
|---|---|
| Fault management | The facilities that enable the detection, isolation, and correction of abnormal operation of the OSI environment. |
| Accounting management | The facilities that enable charges to be established for the use of managed objects. |
| Configuration and name management | The facilities that exercise control over, identify, collect data from, and provide data to managed objects for assisting the continuous operation of interconnection services. |
| Performance management | The facilities needed to evaluate the behavior of managed objects and the effectiveness of communication activities. |
| Security management | The facilities which address those aspects of OSI security essential to operate OSI network management correctly and to protect managed objects. |

## 2. The structure of NetMon

The application **NetMon** follows the client-server paradigm and has a hierarchical organization. A computer running NetMon can monitor one or more networks and one or more servers UNIX or Novell Netware. If such a system has the option **Server Functions** activated, it means that this system itself can be monitored by a system running a client NetMon. Components of NetMon as the daemon for Lotus Notes Server for Windows NT must be installed to run continously on these machines in order to allow a higher level of manageability.

The internal structure of the NetMon application is presented in Figure 1. Netmon has two important layers: the Kernel layer and the User layer. The Kernel layer is composed of:

- a Thread Manager which can be called by an API very close (as functionality) to Mach's "C Threads" library.
- a TCP/IP kernel which implements IP, ARP, ICMP, UDP and TCP and can be called by BSD Sockets Interface.
- an IPX/SPX protocol stack and the basic interaction with the modules RSPX and REMOTE from the Novell Netware server
- At the lower level this TCP/IP implementation is based on the Packet Driver interface. The use of Packet Driver Interface allow also the concurrent use of TCP/IP and IPX/SPX stacks of protocols.
- An Events and Alerts Manager responsible for the defining, treating and logging of events and alerts in NetMon.

- An interpreter for procedures written in NSL (NetMon Script Language). These procedures are used by the Events and Alerts Manager.

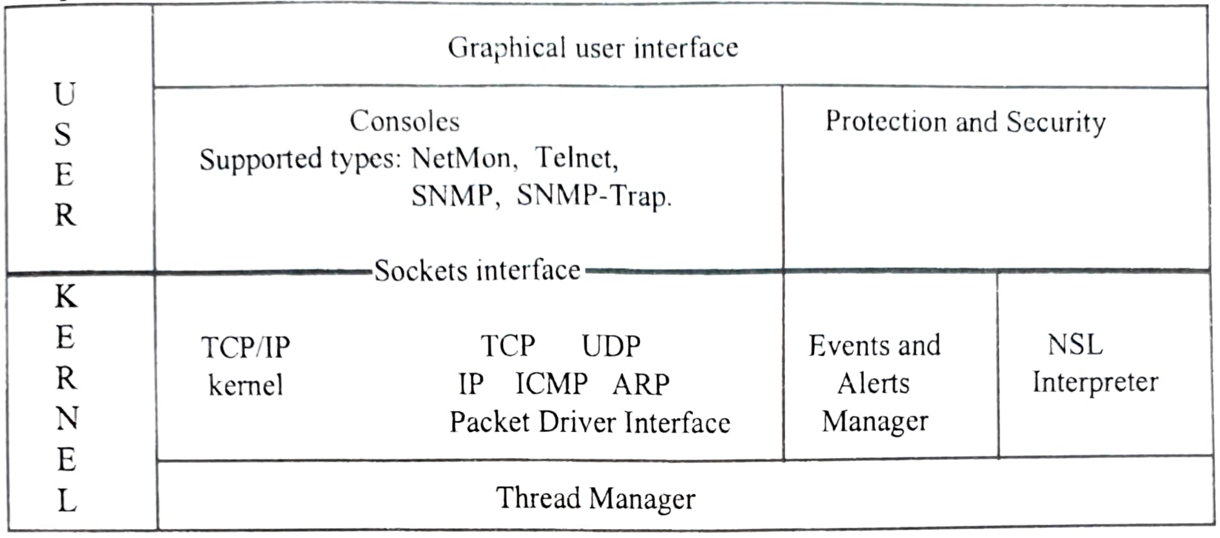

Figure 1. Internal structure of Netmon.

The User layer is composed of:

- a component which manages the Consoles defined in NetMon. The supported console types are: NetMon Client, Telnet, Novell RConsole, Lotus Notes Console or SNMP or SNMP Trap. This is the main component of the User layer, which uses the Thread Manager to create and associate a thread to each console, the TCP/IP kernel to establish connections with the managed servers, and the Events and Alerts Manager to monitor these servers.
- a component which allows the definition of two classes of users and their access rights. This component uses the Events and Alerts Manager to define the access rights of the users.
- the third component is the graphical user interface which is a Windows-like interface.

## 3. The Kernel layer of NetMon.

### *3.1. The Thread Manager*

As a tool for network monitoring and management, NetMon needs to be able to establish connection with several servers and to process data arriving concurrently from them. This is the reason for the implementation of a Thread Manager at the basic level of NetMon. NetMon Threads are "**lightweight threads**", because they are implemented outside the operating system's kernel. The API offered for the use of NetMon Threads consists of the following five calls:

| | |
|---|---|
| Procedure Attach(FarProc: Pointer, m: Integer, FarAddress: Pointer); | Creates a new thread. The first argument is the far address of the procedure whose code will be the thread code, the second is the amount (in bytes) of memory needed for the thread's own stack and the third an address (which can be nil) |

| | |
|---|---|
| | for parameter's passing. |
| Procedure Detach; | Destroys the current thread, releasing the occupied memory. |
| Procedure Suspend; | Suspends the current thread's execution until the next time slice. |
| procedure Tempo(time: integer); | Suspends the current thread's execution for the specified amount of milliseconds. |
| procedure Dispatch; | The current thread is releasing the control of the execution. |

The principle used in NetMon for the thread's execution is that each thread must release the control as soon as possible.

### 3.2. *The TCP/IP kernel.*

At it's basic level, the TCP/IP kernel uses the Packet Driver Interface, which means that it can work with a network driver which follows the Packet Driver Interface Specification. This offers a great portability because the collection of packet drivers from Clarkson University is very large, covering the most used network cards. Using Clarkson's packet drivers collection allowed NetMon to work with Ethernet and Token-Ring (using a modified version of the IBMTOKEN packet driver) cards and also with serial lines (using the SLIP8250 packet driver).

The TCP/IP stack of protocols is implemented following the well-known Internet standards and the API offered for building applications based on this stack is the BSD Sockets interface. There are two versions of this TCP/IP kernel: one that is written as a collection of units used by the DPMI version of NetMon and the other written as a DLL for the Windows version of NetMon. The first version uses intensively the Thread Manager to be able to obtain a very good performance. The second version uses Windows mechanisms for concurrency in order to obtain the same level of performance as the DPMI version.

### 3.3. *The IPX/SPX protocol stack*

The IPX/SPX stack of protocols is implemented following the Novell Netware specifications and by using the Packet Driver Interface Specification it was possible to make IPX/SPX coexist gracefully with TCP/IP. The main difference with NetMon 1.0 is that the support for IPX/SPX is offered only for the DPMI version of NetMon and not for the Windows version. The support for IPX/SPX in the Windows version will be added later.

### 3.4. *The Events and Alerts Manager*

An **Event** for NetMon, or an **awaited event** it is the apparition of a certain message on one of the consoles. An event is in fact an awaited event because is defined as the apparition of a message until a given time (indicated as day of the week, hours, minutes and seconds). There are two possible situations: the awaited event is produced or it is not.

In the first case it is possible to define a **response** to this event, which (in most cases) can be the "execution" of a procedure written in NSL (NetMon Script Language). In fact, the procedure is just interpreted and not executed, NetMon having an interpreter for this script language.

In the second case an **alert** is issued. An alert can be generated not only by the fact that an awaited event is not produced, it can also be generated by errors of the managed networks and servers, and violations of the access rights of the NetMon users, or by other procedures which are currently interpreted.

The Events and Alerts Manager uses the Thread Manager for starting the interpretation of the response procedures and is used by the components of the User layer of NetMon.

## 4. The User layer of NetMon.

### *4.1. Consoles*

The consoles are the main concept in NetMon, because all the monitoring and management activities are defined in relation with them. The internal structure of NetMon as depicted in Figure 2, present the two functional layers of the application. The application NetMon is written in Borland Pascal with Objects and the program structure is built around a hierarchy of classes. For a better understanding of the console's construction and their functionality, a part of the class hierarchy is presented in Figure 2.

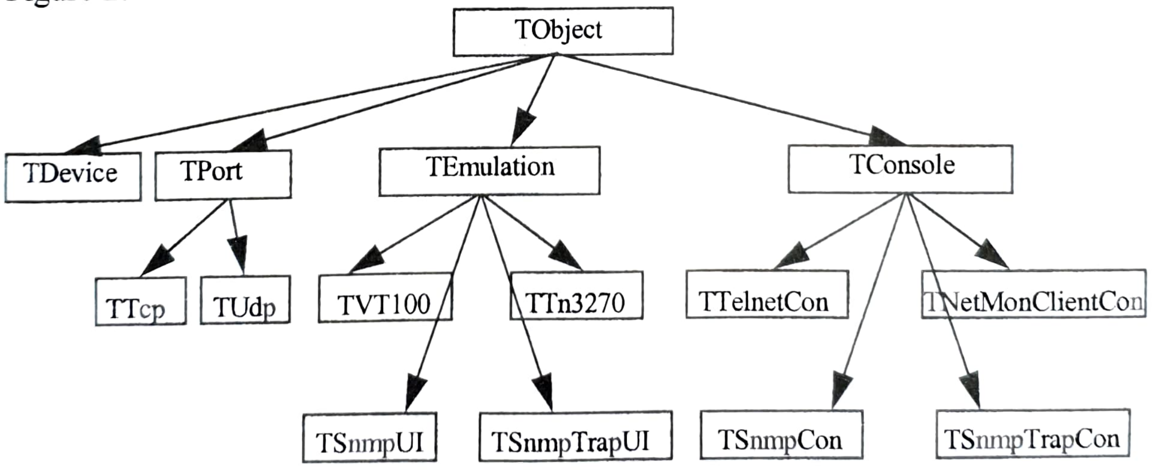

Figure 2. Class Hierarchy in NetMon (partial)

Four classes (and many others which are irrelevant in the following discussion) derive from the root class TObject: TDevice, TPort, TEmulation and TConsole.

The class TDevice represents the devices used for connecting NetMon applications to several types of networks and servers. These devices can be network cards (Ethernet or Token Ring) or serial ports. TDevice has also fields that describe the characteristics and their current settings.

The class TPort represents an abstract class used by inheritance by the classes TTcp and TUdp. Each port of communication has an associated device - an instance of the TDevice class. For example, a TCP port can use an Ethernet network card, but it can also use a serial port (COM2 for example) if it uses the SLIP8250 packet driver. A port has buffers for incoming packets, preinitialized packets to be sent, and other fields necessary for the implementation of the TCP and UDP protocols.

The class TEmulation represents an abstract class used by inheritance by the four classes TVT100, TTn3270, TSnmpUI and TSnmpTrapUI.

There were derived further in this version classes for the emulations VT220 and Tn5250 for connecting better with IBM mainframes or AS/400.

The first two classes have a TCP port (an instance of TTcp) associated and the others have an UDP port associated. Practically, the packets received on a TCP or UDP port are passed forward to a filter that interprets the sequences of characters according to the type of connection and the desired emulation. For example for an instance of TVT100 the packets arrived on the associated TCP port are the input for the finite state machines that describe the TELNET protocol and further for a translation according to the VT100 sequences of characters definition. The instances of TTn3270 are working in exactly the same way. For the TSnmpUI and TSnmpTrapUI instances the things are even simpler: the packets arrived for the associated UDP port are translated from the ASN.1 form to the internal representation.

The classes TTelnetCon, TNetMonClientCon, TSnmpCon and TSnmpTrapCon derive from TConsole to each one being associated an instance of the classes derived from TEmulation. In fact, these classes are the support for the types of consoles supported by NetMon: Telnet, NetMon Client, Snmp and Snmp Trap (see also Figure 2). Each console has a corresponding window in which the incoming data will be displayed. For the Telnet consoles the data supplied by the associated instance of TVT100 are displayed with no modification in the corresponding screen. For Snmp and Snmp Trap consoles the data supplied by the associated instances of TSnmpUI, and respectively TSnmpTrapUI, are displayed in a specific position of the corresponding windows, the specific arrangement on screen being carried out by the methods of the TSnmpCon and TSnmpTrapCon classes. In the newest version were added MIBs for server applications like Oracle server and Microsoft SQL Server.

The protocol for the NetMon consoles is more complicated because a higher perfomance must be obtained. After the connection between a NetMon Client and a NetMon Server (this use a TCP connection) is established, the flow of data will consist of the contents of the open windows of the NetMon Server, at each opening of an administrative window, the needed information being sent. For performance reasons, after a modification in an open window of the server, only the modifications are sent to the client.

## 4.2. *The Lotus Notes daemon and Console*

The Lotus Notes daemon was written as an application for Windows NT using Lotus Notes API for C++. This API is a complex hierarchy of objects which permit to access the collections of Lotus Notes Databases. The Console Log of Notes is one special database in which there are stored all the events, alerts, system messages. The daemon retrieves information from this database and on a regular basis or on specific command of an NetMon server sends this information to NetMon.

The Console for Lotus Notes runs as a server accepting connections for multiple Lotus Notes daemons and present the information collected by them. It is possible from the Lotus Notes Console on NetMon to send commands supported by Lotus Notes Server console and to execute them remote or to store them in the special database.

### *4.3. The Protection and Security in NetMon*

In NetMon there are two classes of users: the Supervisor and the Operators. The Supervisor has all the rights in the application, being able to accomplish all the administrative tasks: defining the awaited events, alerts and responses, creating the Operators and assigning them to specific tasks, monitoring all the consoles, and establishing links with other NetMon servers. The Operators are generally assigned to a subset of events and alerts and are allowed to monitor just a subset of consoles.

### *4.4. The Graphical User Interface.*

The interface of NetMon is a Windows-like interface. In fact, the program uses a unique resource file, NETMON.RC, which describes all the windows, dialogs, icons and string tables. This common resource file make the two versions of NetMon (DPMI and Windows) fully compatible from the user's point of view.

NetMon has a window for each console and a special window for displaying the log file NETMON.LOG. Dialog boxes are used for all the administrative tasks (defining events, alerts, responses, access rights, consoles) and for statistics display.

## 5. Conclusions.

NetMon has been developed in two versions, DPMI and Windows, which are fully compatible. It can monitor UNIX servers using Telnet connections (tests have been made with Linux and Sun) and any network component which has a SNMP server or agent (tests have been made with Novell Netware servers with the SNMP component loaded, with Windows NT Advanced Server and with PCs that have a SMC network card with the SNMP PC Agent loaded). The RMON support was tested with a Novell Netware network with NMS product installed on the server. NetMon allow also the direct monitoring of Novell Netware servers with the native Remote Console incorporated. The incorporated Novell RConsole was tested with Novell Netware 3.11, 3.12 and 4.01. The daemon for Lotus Notes was tested on Lotus Notes 4 on Windows NT 4.0 Advanced Server.

## REFERENCES

[Comer 1991] COMER D., STEVENS D., Internetworking with TCP/IP., Prentice Hall International, 1991.

[Iurian 1994] IURIAN M., IURIAN S., MIHOC C., POP D. An object oriented approach in the implementation of the Simple Network Management Protocol (SNMP), Preprint 5/1994, pag 11-20.

[Iurian 1995] IURIAN M., POP D., IURIAN S., MIHOC C., NetMon - a tool for Network Monitoring and Management, Proceedings of ROSE'95, pag 32-39.

[Manson 1991] MANSON C., HAUGHDAL S., Dynamic and Distributed, Byte 3/1991, pag. 167-172.

[Martin 1992] MARTIN J., LEBEN J., DECnet Phase V. An OSI Implementation., Digital Press, 1992.

[Mihoc 1994] MIHOC C., POP D., IURIAN S., IURIAN M., The implementation of the Virtual Terminal Protocol (Telnet) using an object oriented method, Preprint 5/1994, pag 1-10.

[Novell 1993] Novell Netware 3.12. TCP/IP Supervisor's Guide, Novell Inc., 1993

[Ruder 1993] RUDER D., DLL Anatomy - What's a DLL Made Of, MDSG Technical Article, 1993.

[Stallings 1993] STALLINGS W., SNMP, SNMPv2 and CMIP. The practical guide to Network Management Standards, Addison-Wesley Publishing Company, 1993.

Babeş-Bolyai University, Faculty of Mathematics and Informatics, RO 3400 Cluj-Napoca, str. Kogalniceanu 1, România.

*E-mail address*: {dragos,siurian,miurian,cmihoc}@cs.ubbcluj.ro