A Method for the Remote LAN Management

Tihomir Babić, Vedran Šalinović, Alen Bažant

University of Zagreb
Faculty of Electrical Engineering and Computing
Department of Telecommunications
Unska 3, HR-10000 Zagreb

e-mail: tihomir@godina.hr, v_e_d_r_a_n@hotmail.com, alen.bazant@fer.hr

Abstract

Although the network management concept is not new, efficient network management applications have still been developed. In this paper we propose a method for the efficient remote LAN management. In many network management scenarios management application runs on a station that is connected to the managed local network through the public network, e.g. Internet. The problem with the remote network performance monitoring is how to minimize the amount of management data travelling through the Internet. In a classical scenario management station calls management agents in a local network using SNMP protocol and management data are transferred through the Internet all the time. Doing that way the period of data collection must be long enough to lower the amount of management information travelling through the Internet. Therefore we propose the solution in which management application is implemented in the local network being managed. Accordingly, period of management data collection can be decreased and collected data become more significant for the network performance. The management application periodically collects data and generates web pages. A remote management station connects to the local management application via Internet and receives performance graphs depicting network layer throughput and transmission rates per network interfaces.

Introduction

Network management is one of the most important issues in modern communication networks. Although the concept is not new, there are still many unsolved issues. The majority of network management systems (NMS) use simple network management protocol (SNMP) standardized by IETF [3]. According to the TCP/IP reference protocol model SNMP is an application layer protocol. It relies on transport protocol UDP, and as its name says, it is a simple protocol. The main idea is that SNMP messages must be short and they should not saturate the network. Almost all network management systems today have a simple centralized architecture shown in figure 1 [3].

Network management station runs network management application (NMA) and is called a manager [3]. All other networked devices, such as PCs, workstations, routers, switches etc, are agents, running network management entity (NME) responsible for collecting management data. Manager polls agents and collects the data. SNMP supports three different commands: Get, Set and Trap. Using Get command manager collects data from agents. Set command enables to a manager to change values of different variables in agent's management information base (MIB). Finally, agents use Trap command to warn a manager that something important is going on. Accordingly, a manager should do some action using Get and Set.

Hence, on the protocol level everything seems quite arranged and well defined. But, on the network management application level things are different. There are many products on the market, but it is really hard to find software that is very well suited to network management needs. One of the most important NMAs is Open View [4]. It is huge, powerful software, full of functionalities, but it is also very complex for use and maintenance.

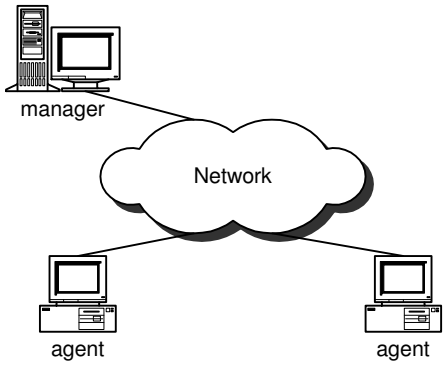

Figure 1. Generic model of a centralized network management system

1. Network Performance Monitoring

In many situations stuff that manages some network needs only a small software packet for network monitoring, without advanced features, like NMA user administration, network topology discovery etc. A typical example of network monitoring tool is MRTG [1, 4]. It simply collects network management data and produces desired graphs presenting network performances. MRTG is a good management tool but in some situations its performances could be quite limited.

Let us presume that a user wants to manage a local area network (LAN). If MRTG is installed on a workstation in a local network, everything looks fine. But if a user wants to access to a local network via Internet, things change. There are many examples where a person monitors several LANs on different locations. In that situation he is forced to access these networks via Internet. The only way for monitoring application to collect data is to permanently send Get commands to these LANs. And that is where the problem arises.

The more LANs have to be monitored the more Get commands should be sent to different locations. Accordingly, the period between two consecutive Get commands sent to the same LAN is bigger and bigger. Second problem is tied to a response time. Sometimes Internet can become quite saturated. NMS sends Get commands periodically in regular time intervals. But there is no guarantee that these commands will reach a target in the same order. Furthermore, responses from managed network devices travel back to NMS. It is quite possible that some UDP packets carrying SNMP data can be lost or suffer from high delays.

2. New concept for a remote network monitoring

All these facts point to a conclusion that a remote LAN management needs a more efficient approach. In this paper we propose a new network management architecture shown in figure 2. Our network management application, called **RPM** (Remote Performance Meter), is based on SNMP protocol and is oriented on both, network monitoring and management. The main focus of the application is network performance monitoring. The application consists of several independent software components that are mutually connected. It is written for the operating system Windows 2000 and can be used on a personal computer or a workstation

supporting Windows 2000. Of course, using the principles described in the sequel of the paper it is possible to build the same application on any other operating system, like Solaris, Linux, etc.

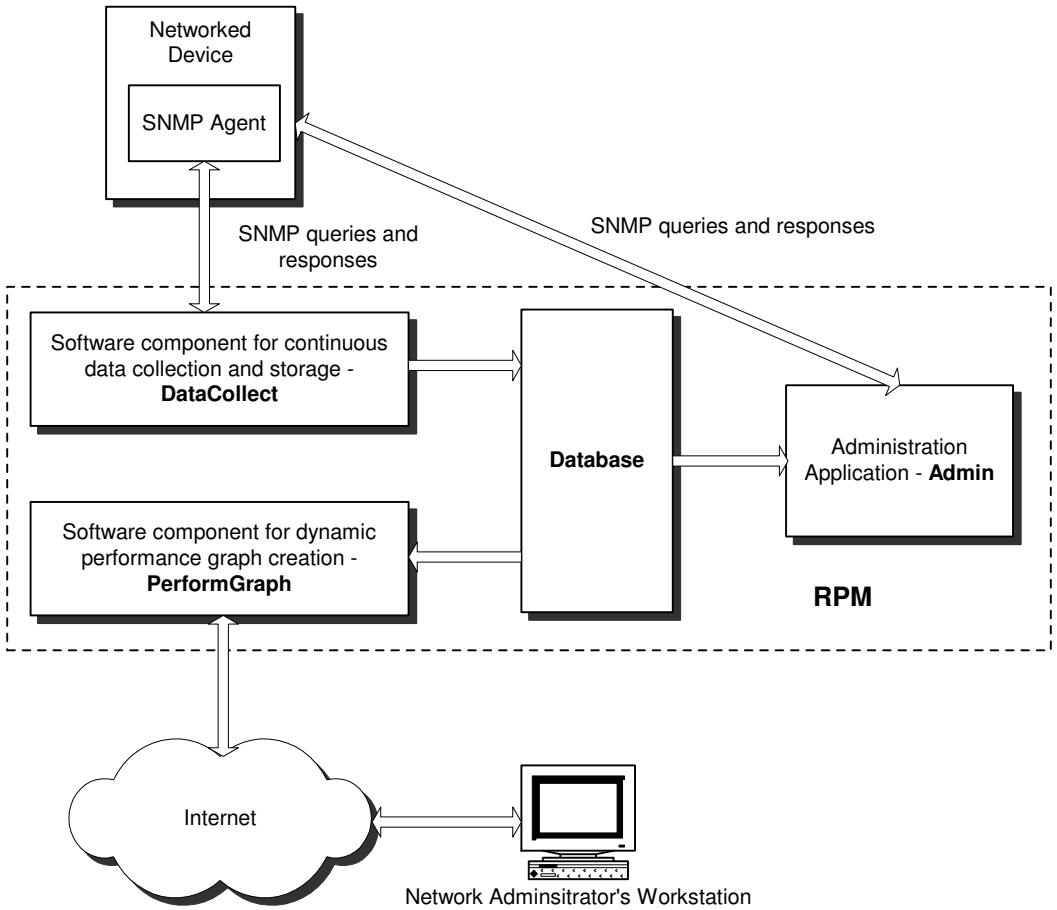

Figure 2. Structure of the network management application

2.1. DataCollect - component for continuous data collection and storage

This part of the network management application is responsible for permanent monitoring of network devices. It sends SNMP queries to agents in regular time intervals. Hence, the DataCollect must be capable of constructing an SNMP PDU carrying Get command (GetRequest, GetNextRequest) with appropriate *community name*. The SNMP is then encapsulated in a UDP PDU and sent to an agent. Upon the receipt of the agent's response DataCollect extracts data from SNMP PDU (protocol Data Unit) and stores data into a database. DataCollect is built as a daemon, i.e. system service. The daemon is written in C++ using Borland C++ Builder version 5 and IPWorks V4 C++ Builder Edition for SNMP based communication.

The daemon can collect any data from network device's MIB. In our practical realisation we have decided to collect input and output rate on managed device interfaces. Hence, the collected data are variables *ifInOctets* and *ifOutOctets* from the group *interfaces*, and *sysUpTime* from the group *system*. Values of *ifInOctets* and *ifOutOctets* present a number of octets transferred through a particular interface from the moment of the last network device startup. Maximum value of both variables is 2^{32}-1. If the variable reaches it's maximum (overflow), the values is reset to zero and the process of counting continues. The value of *sysUpTime* presents a number of centiseconds past from the last network device startup or

from the variable's overflow. Using the above mentioned data it is possible to estimate average transmission rate within a specified time interval. Of course, the shorter time interval chosen the better precision can be achieved. Our application tolerates even time intervals that are only 0.1 seconds short. Such a short time interval cannot be achieved from the remote network management station that manages a LAN via Internet.

DataCollect can be configured via the administration application using GUI. The configurable parameters are IP address of a managed network device, interface identifier and the length of a time interval during which the data will be collected. The parameters are stored in a configuration file SERVIS.CFG. DataCollect reads that file before each query sent to an agent. The configuration can be changed in any moment and it does not influence the consistency of the data in the database.

2.2. Database

The database is created by means of Microsoft Access. It is optimised for storing data collected from agents. The data are stored to the database on the basis of network device IP address, time of collection, interface on a network device, and other parameters. Such organisation gives an opportunity to search the database for a particular network device and its interfaces.

The database consists of two tables. The tables are defined in a way to minimise necessary memory space. It is not a round robin database. It rather holds all the data from the moment when the application is installed and used for the first time. Of course, such an approach imposes a necessity for a large disk space. The example shown in figure 3 reveals a database structure.

ScanID	IPaddress	DateTime	SysUptime
391	213.149.32.1	23. 04. 02 00:36:34	217746184
392	213.149.32.1	23. 04. 02 00:36:49	217747684
393	213.149.32.1	23. 04. 02 00:37:04	217749184
394	213.149.32.1	23. 04. 02 00:37:19	217750684

Figure 3a. Table Scan

ScanID	PortID	InOctets	OutOctets
391	1	381798068	-2102937980
391	2	992965693	706101305
392	1	389300213	-2101005227
392	2	993611515	710444242
393	1	396786926	-2098848774
393	2	994251018	714897329
394	1	405232621	-2096421459
394	2	994905763	719763799

Figure 3b. Table PortData

Upon a receipt of each SNMP response from an agent the application RPM writes a record in the database containing IP address of the network device whose agent sent the response, a value of *sysUpTime* and date and time of the response receipt. The application adds to each record a unique key called ScanID (it is a number of the received response counting from the moment of the first application startup). After that the application writes records in the table PortData. Each record has four fields: ScanID of the relevant response (taken from the table

Scan), Port ID, the identifier of an interface on a managed network device, and values of *ifInOctets* and *ifOutOctets*.

2.3. Admin - administration application

This software component performs several functions. It reads data from the database and creates graphical presentation of desired network performances within specified period of time. In other words, Admin checks for a specified network device and time interval if there are any relevant data in the database. If there are any, it reads them, processes and presents in a graphical form. It is especially useful for monitoring traffic loads on interfaces and pending links. By continuous tracking of the traffic loads it is possible to predict eventual bottlenecks and to avoid them. Admin is written in C++ using Borland C++ Builder version 5.

Admin consists of four components. The first component is the tool for graphical presentation of network device performances, called PerformMeter. It takes data from the database and estimates average transmission rates per interface. The transmission rates are estimated using formulae

$$InputRate = \frac{(ifInOctets_2 - ifInOctets_1) \cdot 8}{(sysUpTime_2 - sysUpTime_1)/100}[bit/s]$$

$$OutputRate = \frac{(ifOutOctets_2 - ifOutOctets_1) \cdot 8}{(sysUpTime_2 - sysUpTime_1)/100}[bit/s]$$

Values of variables $ifInOctets_1$ and $ifInOctets_2$ are collected by two consecutive queries sent to the same network device and one of its' interfaces. The same rule applies to variables $ifOutOctets_1$ and $ifOutOctets_2$, and to $sysUpTime_1$ and $sysUpTime_2$. As stated earlier, consecutive queries can be sent each 0.1 seconds. But from the practical reasons (small querying period necessitates enormous disk space) we have used scanning period of 10 seconds. An example of a graphical presentation of estimated transmission rates is given in figure 4. The blue lines present *ifOutOctets*, and green lines present *ifInOctets*.

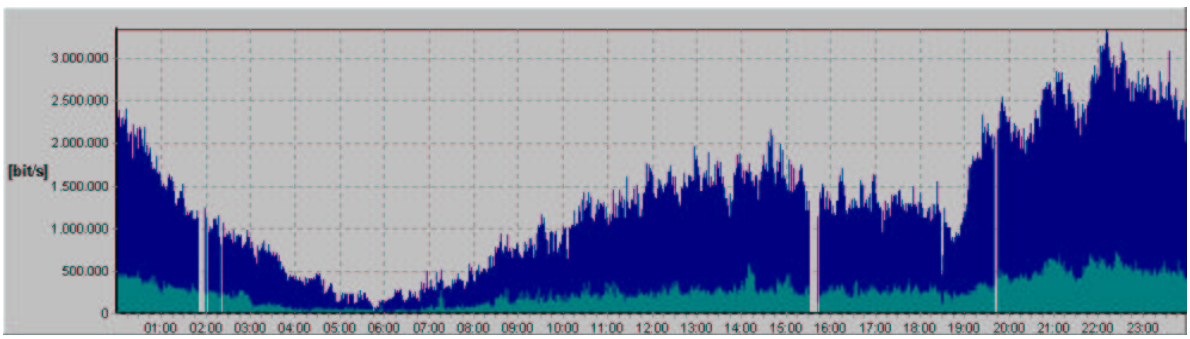

Figure 4. Graphical presentation of input and output transmission rates

Second tool that belongs to the administration application is the tool for graphical presentation of network device real-time performances, called RealTimePerformMeter. It works independently from the database that stores input and output transmission rates. It directly communicates with an agent by means of SNMP queries and collects values of variables *ifInOctets*, *ifOutOctets* and *sysUpTime*. It is written in C++ using Borland C++ Builder version 5 and IPWorks V4 C++ Builder Edition. RealTimePerformMeter user specifies IP address of the managed network device, network device's interface identifier,

community name and time interval between consecutive SNMP queries. The graphical presentation of transmission rates looks very similar to the one given in figure 4.

Third part of the Admin is MIB Browser and the implementation of commands *snmpget*, *snmpgetnext*, *snmpset* and *snmpwalk*. This component, called Manager, is built into the Admin because almost all network management applications support these commands and offers a user GUI that enables him to easily use the commands. MIB Browser gives a graphical presentation of the standard MIB-II structure (figure 5).

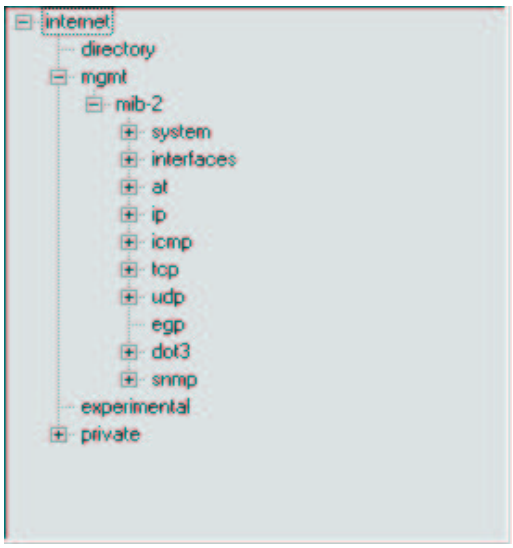

Figure 5. MIB Browser's output

Fourth component is the tool for configuration of the software component for continuous monitoring of network devices, called Config.

2.4. PerformGraph - software component for a dynamic performance graph creation

This component is in fact something that makes our application different from many other network management applications. It is especially useful for a remote network monitoring. Admin should be installed in a local area network and is used primarily for a local network management. DataCollect is installed in a local network, too. Finally, the PerformGraph is in fact a connection between a local network and remote monitoring stations. Hence, our NMS architecture imposes some rules on the system setup. All the application components should be installed on a PC in a local network and the management functions are performed locally. Management stations that connect to a local network via Internet are used for monitoring only. They send command and get graphs prepared locally.

Such an approach is good from two important reasons. First, the amount of SNMP queries travelling through the Internet is minimised and the efficiency of the management data collection is maximised. Second, the security of the system is increased. SNMP protocol has three versions, SNMP, SNMPv2 and SNMPv3. Only the third version implements some security mechanisms. Unfortunately, most of the agents and network management packages do not support SNMPv3 yet. Using SNMP and SNMPv2 is a serious threat to a system security. SNMP PDUs are unencrypted and malicious users can intercept them and read confidential data about local networks. They can also change networking parameters in local routers and switches and cause network failure. Our approach, in a combination with a firewall eliminates security threats. Remote management stations do not send SNMP queries to a local network, but they only receive performance graphs.

PerformGraph is created using Borland C++ Builder version 5. It is in fact a Web application. A computer in a local network, running our management application, must be configured as a Web server. Remote user connects to the Web server using standard Internet browser. After the successful connection the user gets a web page for the managed network devices' performances browsing. The Monitoring Web page is downloaded from server only once during the remote user's connection to the server. PerformGraph, responsible for the remote performance browsing, is created as a DLL. Choosing adequate parameters on the Web page the user sends to the web server an http request for desired resources.

The parameters are sent by mouse clicking on the button Show (figure 6). The DLL component receives these parameters and creates graphical reports. Graphical report is sent to a remote user as a GIF file. Performance graph, contained in the GIF file, appears on the remote user's web page. An example of a graphical output produced by this application component is given in figure 6. While a mouse pointer is placed somewhere above a graph, a blinking frame appears, pointing to an interval one hour long (see figure 6). By choosing the interval a pop-up window is opened (figure 7).

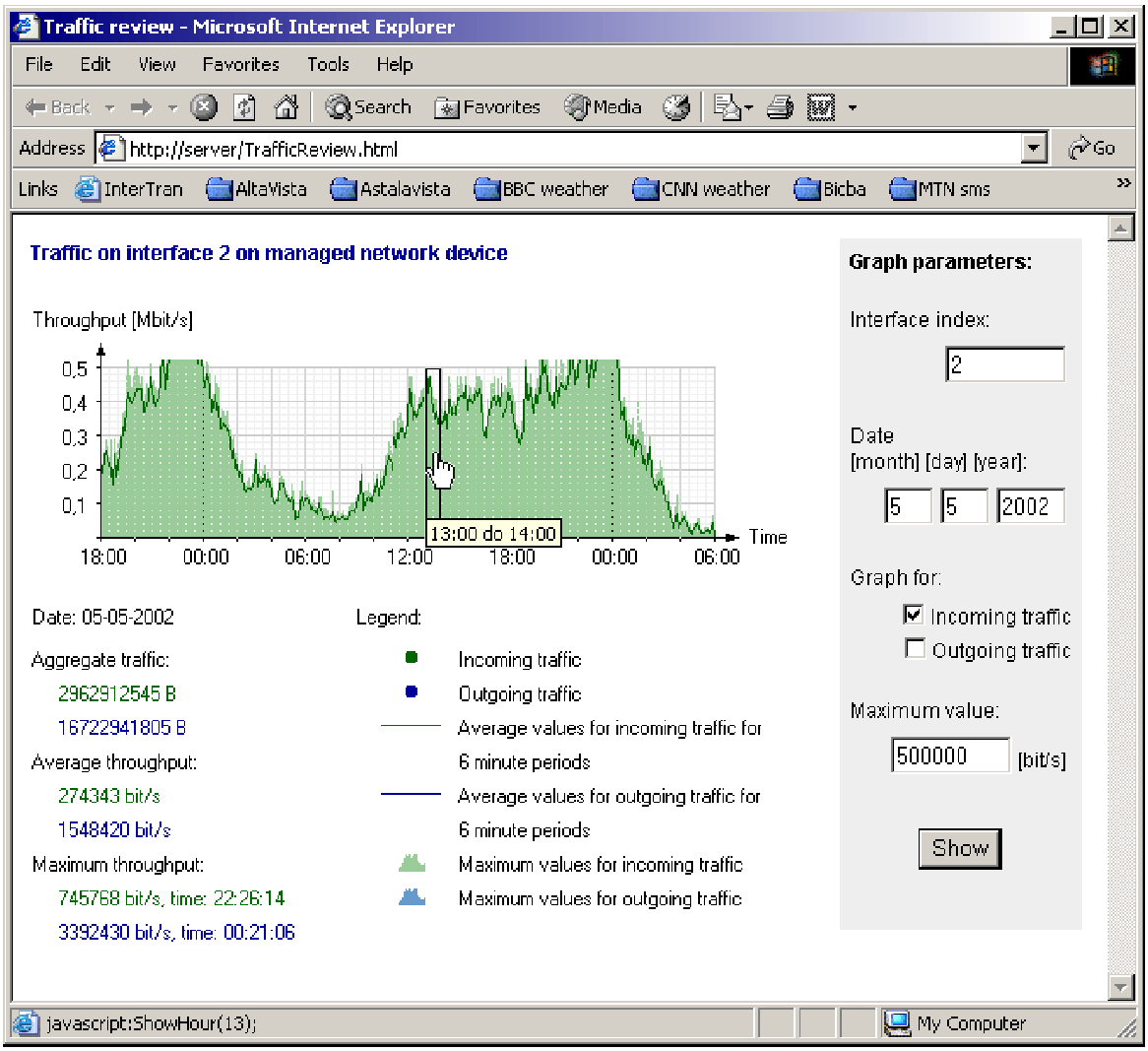

Figure 6. Web application for performance browsing

The body of the web page is written in HTML. DHTML [9] is used for automatic current date presentation, for the synthesis of DLL calls (parameter synthesis), for the web page entry checks and for the pop-up window call. JavaScript is used for creating rollover images on the performance graph.

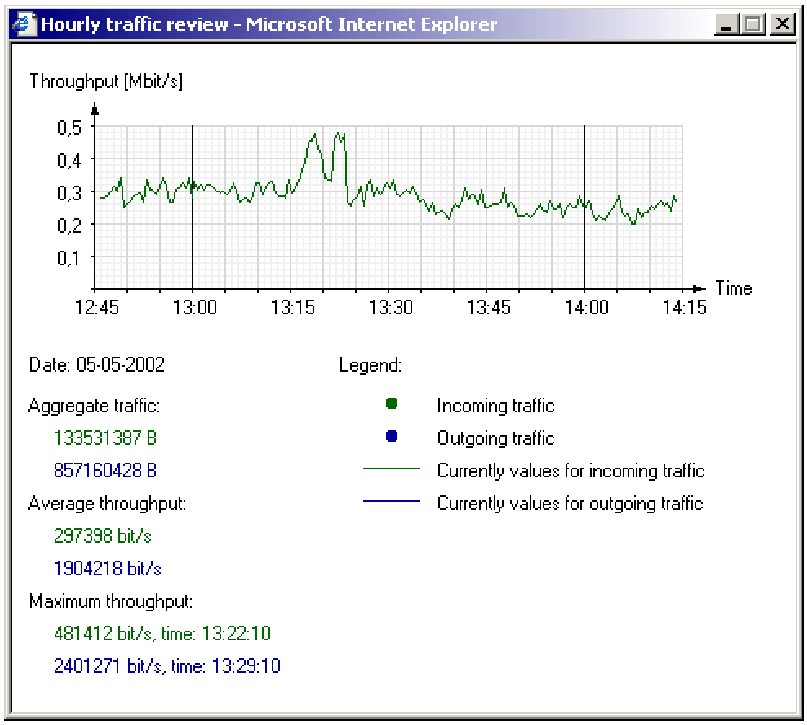

Figure 7. Pop-up window showing performances within one-hour period

Conclusion

The application RPM, presented in the paper, is our proposal of an approach to the network management application building. It is not a complete software package, but it rather suggests how some critical aspects, especially remote performance management, could be solved. The application is open for further development.

References

books:

[1] S. Maxwell, *Red Hat Linux Network Management Tools*, McGraw-Hill, 2001.

[2] William Stallings: "SNMP, SNMPv2, SNMPv3, and RMON 1 and 2", Addison-Wesley, 3rd ed., 1999.

[3] Douglas R. Mauro, Kevin J. Schmidt: "Essential SNMP", O"Reilly, 1st ed., 2001.

web pages:

[4] *SNMP*, NASA Advanced Supercomputing Division, http://www.nas.nasa.gov/Groups/LAN/ClassNotes/snmp/

[5] *RFC 1303: A Convention for Describing SNMP-based Agents*, The Internet FAQ Consortium, http://www.faqs.org/rfcs/rfc1303.html

[6] *Lessons about SNMP*, Institute of Electronics and Telecommunications, Poznan University of Technology, http://www.et.put.poznan.pl/snmp/

[7] *SNMP for the Public Community*, Williams Technology Consulting Services, http://www.wtcs.org/snmp4tpc/default.htm

[8] *MSDN Library*, Microsoft Corporation, http://msdn.microsoft.com/library/

[9] David Gardner, *What is DHTML?*, Irt.org, http://tech.irt.org/articles/dhtml.htm

Short biographies

TIHOMIR BABIĆ is absolvent at the Department of Telecommunications on the Faculty of Electrical Engineering and Computing, University of Zagreb. Since February 2001, he has been working in Globalnet Grupa d.d., Croatian ISP company.

VEDRAN ŠALINOVIĆ received B.Sc. degree at Department of Telecommunications, Faculty of Electrical Engineering and Computing, University of Zagreb, in June, 2002. Since August 2002, he has been in the Army Service.

ALEN BAŽANT is assistant professor on the Faculty of Electrical Engineering and Computing, University of Zagreb. He is with the Department of Telecommunications on the faculty. He is involved in many projects, like seminars “Network Architectures” organized for participants from Ericsson Nikola Tesla d.d, applications WLAN SNMPMan Suite and IPNet Suite developed in cooperation with Siemens d.d., General plan of the Croatian Electricity (HEP d.d.) new telecommunication network and Telemedicine in Croatia. He published more than thirty scientific papers on domestic and international conferences and is co-author of the book “Introduction to ATM”. His main area of interest is data link layer and multiple access protocols applications in local and metropolitan networks.