Remote Service Deployment on Programmable Switches with the IETF SNMP Script MIB

Jürgen Quittek and Cornelia Kappler

C&C Research Laboratories, NEC Europe Ltd.
Hardenbergplatz 2, 10623 Berlin, Germany
{quittek,cornelia}@ccrle.nec.de

Abstract. Some approaches to network programmability require control processes for network devices. Such control processes are executed at the device or at a control node locally connected to the device. This paper discusses the management of these control processes, which is part of configuration management.
Typically, similar or identical control processes have to be installed on several devices of the same network in order to realize a desired service. In such cases an automated remote installation and maintenance of control processes is desirable, especially when devices are located far away from each other.
The idea presented and evaluated in this paper is managing the control processes remotely with the IETF SNMP Script MIB. It is shown that the Script MIB provides sufficient means for installing, configuring, starting, updating, replacing, and terminating control processes for programmable network devices.
We applied our idea to GSMP control processes realizing an IP switching over ATM service. This case study demonstrates the feasibility, but also discusses practical problems and restrictions.

1 Introduction

Programmability of networks is becoming an important factor in business scenarios for network providers. Major reasons for this trend are the increased speed of the development of technology and standards, the specialization of providers, and the need for rapid service creation and deployment.

Currently, several approaches to network programmability are being discussed. They range from solutions with restricted programmability fitting into the existing networking frameworks, to very general and flexible approaches requiring new technologies and devices.

An example for the first kind of approaches is policy-based networking using the COPS protocol [1]. Here, access to networking services is controlled by policies which can be changed during operation of the network. Policies are stored at one or more policy servers which are asked for access decisions. They can e.g. be consulted by an IP router to decide the acceptance of an RSVP reservation.

An example for a radical new approach are active networks [2], where each packet sent via the network may carry code to be executed on its way. The

code may just influence the way a packet is routed, but it may also implement a complex service, e.g. a multicast of a packet stream with a data compression adapting itself to the available bandwidth.

Approaches relevant to this paper are in-between the two extrema, using APIs or communication protocols for programming single network devices or even entire networks. They offer a high degree of programmability without requiring large investments in new technology. These approaches require *control processes*, using the protocol or the API for managing and controlling the network. The main topic of this paper is how such control processes in turn can be managed within the Internet management framework.

An example for the API approach is the proposed IEEE PIN standard for application programming interfaces for networks. It is being defined by the IEEE P1520 working group[1] which consists of three sub groups covering ATM switches, IP routers and switches, and SS7 systems. The API supports control of network services involving several devices, as well as settings of specific properties at single devices.

An approach using a communication protocol for controlling single switches is the General Switch Management Protocol (GSMP). The first versions have been proposed by Ipsilon Networks, Inc. as informational RFCs (RFC 1987 [3] and RFC 2297 [4]). In February 1999, an IETF working group has started producing a standard[2]. In May 1999, the Multiservice Switching Forum[3] (MSF) selected GSMP as its base line protocol.

GSMP is used for managing and controlling ATM switches including switch and port configuration, connection management, and monitoring. GSMP separates the control software from the switch fabric. They communicate with each other via a reserved ATM connection. This gives high flexibility to control software design.

Typically, for GSMP as well as for other protocol and API approaches, similar or identical control processes have to be installed on several devices of the same network in order to realize a desired network functionality. In such cases an automated remote installation and maintenance of control processes is desirable, especially when devices are located far away from each other. Usually, this is considered to be a part of configuration management.

We suggest to use the IETF SNMP Script MIB [5] for installing, starting, updating, replacing, and terminating control processes remotely. The Script MIB has been designed for delegating management functions as scripts to SNMP agents. As a case study we managed IPSOFACTO control processes. IPSOFACTO [6] is an IP switching technology realizing IP services over ATM with GSMP control processes. It is currently being discussed as an multicast extension for MPLS [7]. Based on this experience, we discuss the benefits and problems of managing network control processes with the Script MIB.

[1] see http://www.ieee-pin.org/
[2] see http://www.ietf.org/html.charters/gsmp-charter.html
[3] see http://www.msforum.org/

The remainder of this paper is structured as follows. The Script MIB is introduced in more detail in section 2. Section 3 describes GSMP, particularly covering requirements for the control processes. A brief introduction to IPSOFACTO is given in section 4. Section 5 describes our experiences with managing IPSOFACTO control processes with the Script MIB.

2 The IETF Script MIB

The Script MIB has been developed by the IEFT distributed management working group[4] and was published as proposed standard RFC 2592 in May 1999 [5]. The Script MIB supports starting, controlling, and terminating remote processes (scripts) within the SNMP management framework. It furthermore gives access to output of the scripts. The scripts may be implemented in arbitrary programming languages and code formats. The MIB provides the following capabilities:

- Transferring management scripts to SNMP agents at distributed locations.
- Transferring script arguments.
- Initiating, suspending, resuming and terminating management scripts at these locations.
- Transferring the results produced by running management scripts.

2.1 Overview of the Script MIB

The Script MIB consists of six tables, the main four tables are introduced below. In order to keep our overview clear and short, we omit some features, such as *language extensions* and *pushing* of scripts.

The *language table* provides information about the languages supported by a Script MIB implementation, e.g. the Java virtual machine or the Tcl interpreter.

The *script table* lists all scripts known by a Script MIB implementation. Scripts can be permanently installed, or downloaded (pulled) by the agent from a URL.

The *launch table* describes scripts that are ready to be launched. An entry defines the argument passed to a script and the owner indicating the permissions and credentials of the script during execution. A managed object in the entry called *launch button* allows a manager to invoke a script with a single SNMP set operation.

Finally, the *run table* lists all scripts that are currently running or that have terminated recently. It contains objects that allow a manager to

- retrieve status information from running scripts,
- control running scripts (suspend, resume, abort),
- retrieve the arguments passed to scripts at invocation,
- retrieve intermediate results of running scripts,
- retrieve results from recently terminated scripts,

[4] see http://www.ietf.org/html.charters/disman-charter.html

– retrieve starting and termination time of scripts,
– control the remaining maximum lifetime of a running script,
– control how long script results are stored in the MIB.

The Script MIB supports SNMPv3 including the User-based Security Model (USM) and View-based Access Control (VACM). A detailed discussion of Script MIB security can be found in [8].

2.2 Script MIB Usage and Communication Model

We explain the usage of the Script MIB by describing the steps required for running and controlling a script on a network node. Figure 1 illustrates these steps:

1. In order to install a script at the network node, the manager has to check which languages are supported by the Script MIB implementation. He retrieves this information at the language table.
2. Based on the information about supported languages, the manager selects an appropriate script from a script repository. The script has to be installed at a new row of the script table which the manager creates. Now, the manager writes the script URL to the script table and then requests the SNMP agent to download the script from the script repository.
3. When the desired script is installed, the manager creates an entry in the launch table indexing the script and describing an execution environment. For the same script, several entries can be created in order to start the script

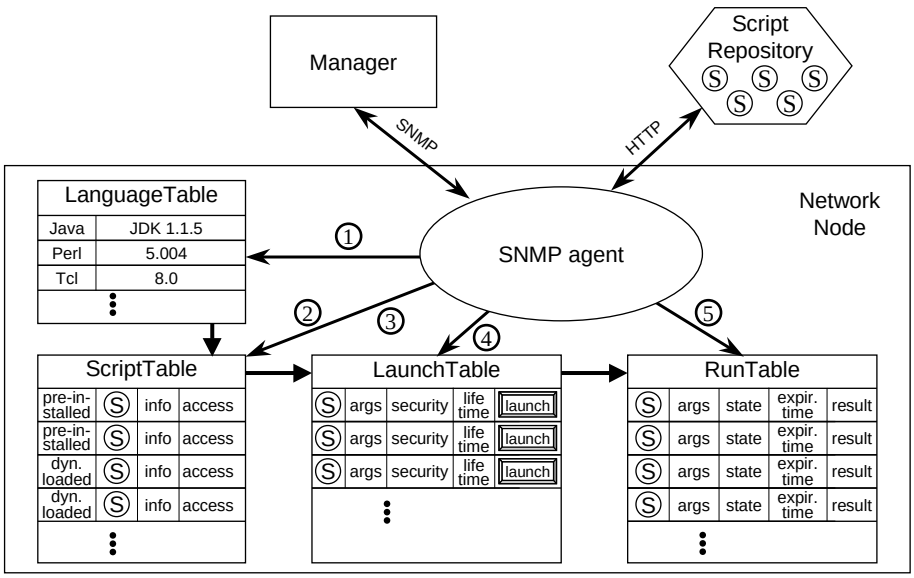

Fig. 1. Usage of the Script MIB. Circled digits refer to step numbers in section 2.2.

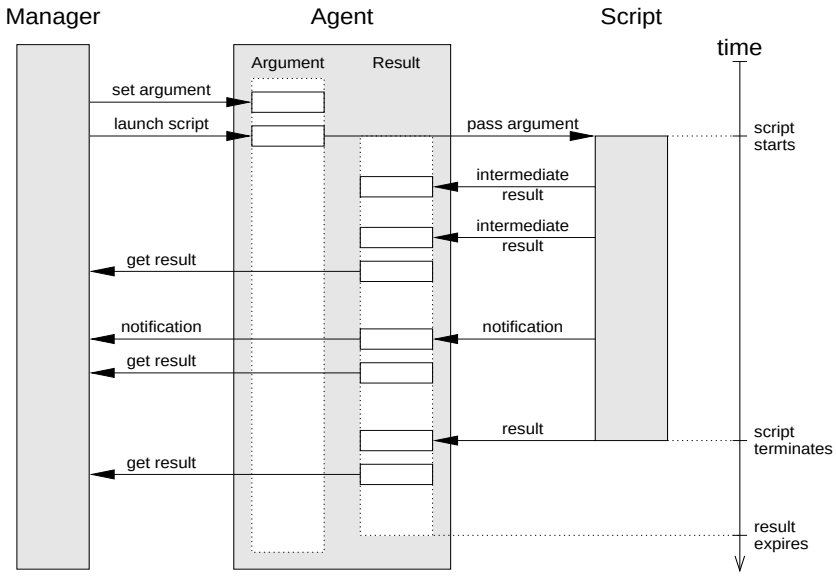

Fig. 2. The manager–script communication model

with different predefined arguments, or to run the script under different runtime security profiles.
With the entry in the launch table properly set, the manager can start a script by setting the launch button. By setting the launch button more than once, several concurrently running scripts can be started from the same row of the launch table. For each running instance of a script, an entry in the run table is created automatically.

4. The manager controls running scripts and retrieves intermediate or final results of running scripts by accessing the run table. The communication model between the manager and a running script is shown in Figure 2. The final result produced by the script is kept by the agent for a expiration time given by a managed object in the according run table entry. During this time the manager can read the result. The manager can clear the result anytime by setting the expiration time to zero.

2.3 Implementations

There are currently three implementations known to the authors. One has been developed at the Open Group Research Institute at Grenoble, France, and another one at SNMP Research International in Knoxville, TN, USA.

The implementation used for the experiments described in this paper has been developed jointly by the NEC C&C Research Laboratories in Berlin, Germany, and the Technical University of Braunschweig, Germany.[5]

[5] see http://www.ibr.cs.tu-bs.de/projects/jasmin/

This implementation supports SNMPv3 security and uses the Java virtual machine [9] as script language. It separates the SNMP agent functionality from the runtime engine for executing Java byte code. The agent controls one or more runtime engines via the Script MIB eXtensibility protocol (SMX) which is defined by RFC 2593 [10].

SMX facilitates adding new script languages to the agent implementation. A runtime engine for a new language can be built independent of the agent. The only change required for the agent is adding a new entry to the language table. It is planned to extend the current implementation by a real scripting language, such as Tcl, Perl, or Python.

3 The General Switch Management Protocol

The General Switch Management Protocol (GSMP) has been suggested by Ipsilon Networks Inc. in 1996. The first version 1.1 published as experimental RFC 1987 [3] defined a communication protocol between an ATM switch and a switch controller, with the controller acting as master and the switch as slave. It is available available on several commercial ATM switches.

GSMP 1.1 allows the controller to

- configure the switch,
- configure ports,
- establish and remove virtual point-to-point connections,
- establish and remove virtual point-to-multipoint connections,
- request monitoring information,
- request event notifications.

The switch acknowledges all received control messages. Monitoring requests are acknowledged by sending port statistics or connection statistics. If event notification has been requested by the controller, the switch notifies the controller on events such as *port* up and *port down* by GSMP messages which have to be acknowledged by the controller.

Messages between switch and controller are sent over a reserved ATM connection which must be established in advance. This allows the control process to be executed on the switch, as well as on a device connected to a port of the switch.

The abstract switch model of GSMP 1.1 just considered plain connections without further properties for Quality of Service (QoS). This drawback was one of the major motivations for developing GSMP 2.0 published as experimental RFC 2297 [4] in 1998. However, the suggested solution for QoS support was not accepted as widely as the original GSMP 1.1.

In February 1999 the new IETF GSMP working group has started to refine the QoS support of the abstract switch model and adapt GSMP also to non-ATM switching technologies.

4 IP switching with IPSOFACTO

IPSOFACTO maps IP flows to connections in ATM networks [6]. The ATM switch controllers run an IP routing protocol and perform IP forwarding. ATM signaling is not used. Figure 3 shows components of an IPSOFACTO network. An IPSOFACTO network consists of ATM switches (IPSOFACTO routers), ATM hosts running the IP protocol stack (IPSOFACTO clients), and IPSOFACTO gateways to other data link layer technologies, e.g. Ethernet. IP packets are transferred over ATM connections. In the initial state, a switch forwards all packets coming in from other routers or from hosts to the IPSOFACTO controller. In this state we call all connection identifiers *unused*.

If an IP packet arrives with an *unused* incoming connection identifier, it is passed to the controller. The controller performs IP routing in order to decide to which outgoing port the packet must be forwarded. Then the controller selects an *unused* outgoing connection identifier of this port and sends the packet.

In a further step, the controller creates a connection by establishing a shortcut from the incoming connection identifier of this packet to the outgoing one. Thereby, the incoming connection becomes *used*. Now, all subsequent packets arriving via this connection are switched directly to the same outgoing connec-

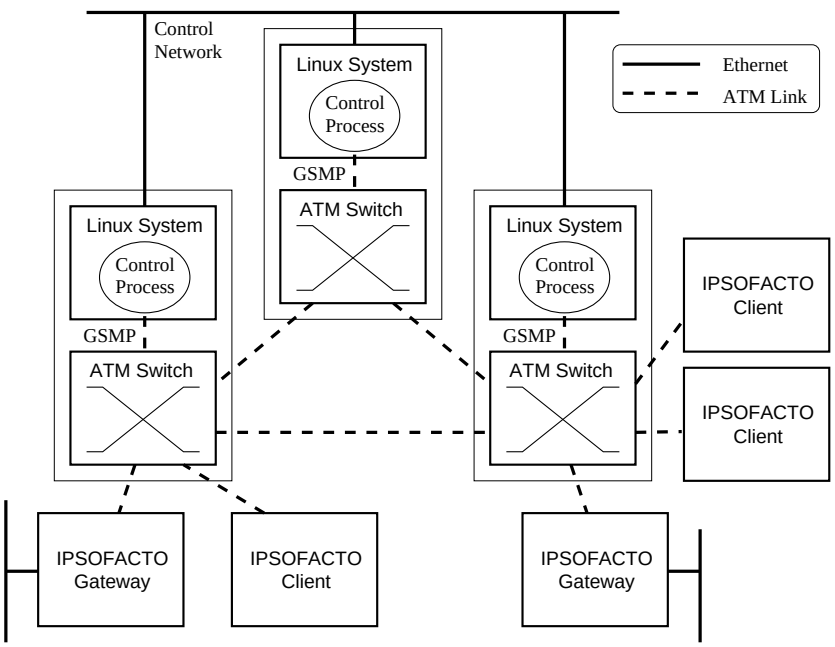

Fig. 3. An IPSOFACTO network containing three routers, two gateways to Ethernet, and three IPSOFACTO clients. An IPSOFACTO router consists of an ATM switch, and a Linux system running a GSMP control process.

tion without executing the IP protocol stack. For multicast IPSOFACTO may replicate packets and forward them to several several outgoing connections.

Recently, the IPSOFACTO approach has been accepted by the IETF as the multicast extension of Multi Protocol Label Switching (MPLS) [7]. So far MPLS only supports unicast. As multicast extension, the IPSOFACTO approach would be used for multicast only, unicast would be handled by MPLS itself. This version of IPSOFACTO will not anymore be restricted to ATM. Like MPLS it can be applied to several switching technologies.

The current implementation of IPSOFACTO has been developed by the NEC C&C Research Laboratories in Princeton. This implementation and also the upcoming implementation as multicast extension for MPLS use external control processes connected to the switch via GSMP 1.1. Control processes run on Linux machines connected directly to the controlled switch by an ATM link (see Figure 3). The devices hosting the control processes are accessible via an IP control network.

5 Managing control processes

The main topic of this paper is automated and remote management of control processes with the Script MIB, which is part of network and service configuration management. The management tasks considered are

- software installation for control processes,
- updating the software installation,
- removing the software installation,
- customizing the local configuration,
- starting control processes,
- monitoring control processes,
- terminating control processes.

Below we discuss these tasks individually. As a case study, we investigated the applicability to an IPSOFACTO network.

The general idea – performing the above listed management tasks automatedly and remotely – is based on the assumption that tasks are the same on each device, or at least on subsets of all devices. In this case, scripts can be written performing all necessary steps required for a task locally on devices. If we assume that a management station has a list of devices to perform a task at, then it can – via the Script MIB – transfer the corresponding script to each device of the list and execute it. The Script MIB also allows monitoring the execution of the script and reacting on error messages.

An obvious requirement for this procedure is the availability of an IP stack on the device executing the control process. This is given for pure IP devices, but not necessarily for e.g. ATM switches. On many switches, installing an IP stack is just a matter of a local pre-installation, on some it might be impossible to get one. For the IPSOFACTO control processes this problem did not occur, because the control processes were running on Linux machines.

5.1 Installing Network Control Software

In case of a software installation task, the installation script has to transfer the network control software from a repository to the device, and install it properly. For the transfer, ftp or http are recommendable protocols, because they can easily be used within scripts.

After the transferred files have been put in place, it might be necessary to edit the configuration of the device which is stored in some configuration storage, such as files or non-volatile RAM. This may cause problems, because the current state of the file or other storage to be edited might depend on the history of the device, i.e. the required steps of editing might depend on what has been installed before. This is a typical problem of automated software installation and it cannot be avoided in some cases.

When installing IPSOFACTO, we ran into another problem. The software to be installed included a new operating system kernel. So after installation the device had to be rebooted. Similarly, editing the configuration as described above may only becomes effective after a reboot.

A reboot interrupts the management of the device for some time and since we do not consider persistent scripts, actions to be performed before and after the reboot have to be part of different scripts. Furthermore, it is not recommendable to call a reboot at the end of the installation script, because an error message which might have been generated during installation might get lost. Therefore, after finishing the installation successfully, the manager should start a separate script which reboots the system.

Unfortunately, the Script MIB does not support the installation of scripts which are executed automatically after a reboot, i.e. when the SNMP agent is restarted. Hence, the manager has to poll the device until it becomes operational again after reboot. Then further steps, e.g. starting the installed control process, can be performed.

5.2 Removing and Updating Software

Requirements for software removal scripts are similar to those for the installation scripts. The major difference is that usually no transfer of the network control software is required. Editing of configuration storage has to be reverted, and instead of installing files, files are being removed.

In case of the removal of IPSOFACTO software, the old kernel has to be restored, which requires again a transfer of software, if the old one is stored on the device itself. As for the installation, a reboot is required after removal.

Software can be updated by removing the old version and installing the new version. In many cases this procedure can be abbreviated, because only few files need to be replaced, and editing of the configuration storage might not be necessary. In this case, it is more efficient to use update scripts performing only this reduced number of steps.

5.3 Customizing the Local Configuration

While the steps performed for software installation often are the same for all devices, network control processes might require local configuration which is specific to each device. The process can obtain the local configuration information from configuration storage or from the arguments passed to it when it is started. Providing the local configuration information might require manual steps and therefore inhibits the automation of this task.

E.g. the IPSOFACTO control process needs to be informed about the number of ports of the controlled switches, the IP addresses assigned to these ports, and the IP addresses of the adjacent switches or hosts. To solve the problem we manually created a configuration file for each switch. The files were put into the software repository and named according to the device they were to be used for.[6]

After this preparation, the script for installing the local configuration was no longer device specific.

5.4 Starting and Terminating Control Processes

Scripts for starting and terminating control processes are rather simple. The starting script just has to start one or more processes using the software installed previously. The terminating script must be able to identify the processes to be terminated. This can be done by scanning the process table, or with support of the starting script which then needs to store the process identifiers of processes it started.

If the starting script passes local configuration information to the control processes, a manual preparation of each device might be necessary.

For IPSOFACTO, the starting script containing just three simple steps worked well. However, the terminating script, scanning the process table and terminating the IPSOFACTO processes, could not bring the device to a consistent state from where IPSOFACTO or some other control process could be started subsequently. The reason is that the current IPSOFACTO implementation does not contain code for a clean termination – automated management was not considered during design of the software. Consequently, after terminating IPSOFACTO, the device had to be rebooted in order to bring it into a state allowing further operation (cf. section 5.1).

We also tested termination and restarting of control processes without rebooting, with a simple GSMP control process which just established and removed a single ATM connection.

[6] It is also possible to automate this task by generating the individual files from a single description of the entire network.

5.5 Monitoring Control Processes

How a control process is monitored may vary significantly. It depends on the runtime environment of the control process, and on the output produced by the process. We identified two typical monitoring actions: scanning the process table to find out whether the process is still running and what resources it consumes, and parsing log files. Parsing log files includes checking output sent to the standard output, since the standard output can be redirected to a file.

For both monitoring actions, two kinds of scripts can be useful: scripts running permanently and repeating their actions with a given frequency, and scripts performing a monitoring action only once. The permanently running scripts are preferable if a fault condition should be detected quickly. These scripts can notify the manager when they detect such a condition. The other kind of scripts is useful if the manager wants to obtain monitoring information on a non-regular basis.

5.6 Complexity of the Management Procedure

In this section we analyze the complexity of the management procedure described above. The following section discusses a possibility of reducing it.

The procedure requires the development of a script for each management task. In order to execute the whole procedure, for each script and for each device on which the script is run, several steps have to be performed: the script must be installed at the script table, a launch entry with proper script arguments must be created, the script must be started and controlled via the run table, and after successful execution, the created entries in the launch table and in the script table should be removed.

Each of these steps requires several SNMP requests to be sent to the device. However, the steps are identical or very similar for each script and a management application can use a common subroutine for each of them. Consequently, the management application is rather simple, once these subroutines are available as a library for accessing the Script MIB.

The major effort and complexity reside in the creation of the scripts themselves. However, this complexity is unavoidable, because most of the steps are definitely required for managing these control processes. Hence, the additional complexity of the presented procedure is small, while the advantage of managing a set of devices automated-ly and remotely might be quiet high.

5.7 Possible Simplification of the Management Procedure

There is a possible simplification of the procedure. Instead of transferring scripts to the managed device which in turn install software and run it, it is possible to implement a GSMP control process as a script directly managed via the Script MIB. In this case, management steps, such as installing the software, starting it, and terminating it, are performed by only a few SNMP requests. The effort

required for the whole management procedure is then comparable to the effort required for a single task of the procedure described above.

Unfortunately, this simplification has some restrictions. The control process has to be written in a language supported by the used Script MIB implementation. This might require porting existing control software to another language which might be not preferable to use for other reasons. Preferred languages for control processes on network devices are C and C++, because these languages can easily interface with operating systems and communication stacks usually being written in C. However, C or C++ are currently not supported by any Script MIB implementation, because their portability is restricted.

Furthermore, the simplification is only efficient if configuration files and log files can be avoided, because otherwise further scripts are required for installing or reading them, respectively. Hence, the configuration information should be passed within the list of arguments, and the output is restricted to the means of the Script MIB, i.e. producing intermediate results and notifications. Here, the final script result is not of much interest, since the control process is intended to run until it is terminated explicitely.

Hence, the simplification can only be applied to control processes for which these restrictions are acceptable. In our opinion this hold only for a small set of network control processes.

6 Conclusions and Future Work

Our case study with GSMP control processes showed that it is possible and efficient to manage control processes for programmable networks with the SNMP Script MIB as part of configuration management. This includes installing, configuring, starting, updating, terminating, and de-installing control processes.

The main effort for these management tasks is caused by writing scripts. But since most of the steps contained in the scripts are required anyway for managing the control processes, the generated overhead is very small. On the other hand, the reduction of work introduced by an automated remote management of network control processes on a set of devices can be quiet significant.

We showed that the suggested management procedure is even applicable for control processes which are hard to manage, particularly which require a customized kernel and which do not leave the device in a proper state after termination.

For simple control processes, the procedure can be simplified by managing the control process directly via the Script MIB without using additional scripts.

Since we did not have to consider GSMP specific issues, we assume that we can apply this way of managing control processes also to other kinds of programmable networks. Particularly, we are interested in studying the applicability to the IEEE PIN standard, as soon as an implementations is available.

Acknowledgments

The authors would like to thank Jürgen Schönwälder at the Technical University of Braunschweig who is one of the authors of the Script MIB and who designed and developed – together with Matthias Bolz and Sven Mertens – the Script MIB agent used for this paper.

Further contributions came from the NEC C&C Research Laboratories in Heidelberg, Germany, where Jürgen Röthig, Sibylle Schaller and Frédéric Griffoul installed, supported, and enhanced the used IPSOFACTO system.

References

1. J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan and A. Sastry. *The COPS (Common Open Policy Service) Protocol.* internet draft, draft-ietf-rap-cops-06.txt, February 1999.
2. D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall and G. Minden. *A Survey of Active Network Research.* IEEE Communications Magazine 35(1):80-86, January 1997.
3. P. Newman, W. Edwards, R. Hinden, E. Hoffman, F. Ching Liaw, T. Lyon and G. Minshall. Ipsilon's General Switch Management Protocol Specification Version 1.1. RFC 1987, Ipsilon Networks, Inc., Sprint, August 1996.
4. P. Newman, W. Edwards, R. Hinden, E. Hoffman, F. Ching Liaw, T. Lyon and G. Minshall. Ipsilon's General Switch Management Protocol Specification Version 2.0. RFC 2297, Nokia, Sprint, Fiberlane Communications, March 1998.
5. D. B. Levi and J. Schönwälder. Definitions of Managed Objects for the Delegation of Management Scripts. RFC 2592, SNMP Research, TU Braunschweig, May 1999.
6. A. Acharya, R. Dighe and F. Ansari. A Framework for IP switching over fast ATM cell transport (IPSOFACTO). In *Proc. SPIE Voice Video and Data Communications, Broadband Networking Technologies,* pp. 20-28, November 1997.
7. D. Ooms, W. Livens, B. Sales, M. Ramalho, A. Acharya, F. Griffoul and F. Ansari. *Framework for IP Multicast in MPLS.* internet draft, draft-ietf-mpls-multicast-00.txt, Alcatel, NEC, Bell Labs, June 1999.
8. J. Schönwälder and J. Quittek. Secure Management By Delegation within the Internet Management Framework. In *Proc. 6th International Symposium on Integrated Network Management*, Boston, May 1999. (to appear).
9. T. Lindholm and F. Yellin. *The Java Virtual Machine Specification.* Addison Wesley, 1997.
10. J. Schönwälder, and J. Quittek. *SMX - Script MIB Extensibility Protocol Version 1.0.* RFC 2593, TU Braunschweig, NEC Europe Ltd., May 1999.