An Integrated and Automated Network Management and Operation System[*]

Diego Kreutz and Benhur Stein

Tecnologia da Informação (TI)
Programa de Pós-Graduação em Engenharia de Produção (PPGEP)
Universidade Federal de Santa Maria (UFSM)
kreutz@inf.ufsm.br, benhur@inf.ufsm.br

Abstract. Computer network management is an area that demands lots of work and lacks tools and professionals qualified to the management of the most diverse administration domains. In this context, this work describes the project and development of an integrated, dynamic and automated system that is able to simplify and reduce the time needed for many daily tasks of network administrators. The initial results on the developed prototype show that the system can be used to solve or automate many management tasks.

1 Introduction

The expansion and installation of networks of computers does not seem to stop. Every day more and more equipments and new networks are included into corporative and academic environments and, commonly, connected to the Internet. As a result, there is an increasing complexity in security and management needs.

Many tools exist to help network administrators in their tasks. Many times they are designed for particular types of environments or to fulfill only some sets of administrative needs. Every tool makes the management of some aspects or resources of a network easier. However, they also seem to lack resources and characteristics in some important aspects like using an unified information source, allowing the use of the existing administration systems, being easily extensible, having scalability capabilities or providing mechanisms for automated detection and processing of configuration updates.

One way of reducing the relative isolation and lack of interoperability associated with the current management solutions is to provide some kind of mechanism (or system) that will support the control and maintenance of a network. This solution should also be able to easily incorporate new functionalities and at the same time integrate the different solutions in use. The advantages of such an integration goes from reduction of management faults, through the use of automated processes to verify and maintain the configuration information, simplifying and speeding up common management tasks. And this is one point where this work intends to provide resources for.

[*] Supported by CNPq

In this context, we show here the project and development of an integrated network operation system that aims to simplify and automate many management tasks, as the configuration and availability maintenance of network services. After the system, which is called `ida` (a shorthand for `idaNMOS` [1]), is installed on the network, services and computers configuration updating and maintenance will be thrown out in a transparent, effective and simple way. The network manager will only feed a database with information needed by `ida` to control the systems. As it is based on independent and flexible components, create new ones or extend existing ones is a simple and relatively easy task, providing a way of easily adapt it to different management needs.

The next section presents some related works. Section 3 explains `ida`'s architecture. Sections 4 and 5 show the current state and validation of the system. Finally, we have the conclusion and future work.

2 Related Work

There are many management tools. To facilitate the characterization and exemplification of these systems, we have classified them in four categories:

Protocols: communication and control standards used to monitor and manage mostly network devices;

Configuration languages: configuration and management platforms that have a language to describe, configure and control systems;

Distributed Management Tools: tools that simultaneously manage many devices or systems of a network;

Single-System Management Tools: conceived to control the systems of an unique machine or device.

Protocols. The management protocols are standardized communication models that intend to control and operate a vast number of devices of different manufacturers. One of the goals of the management protocols is to define operating standards that can be used and implemented for the management of network equipments in general.

SNMP [4] is one of the most known and used protocols, both by private and academic organizations and NetConf [6] is a new management protocol being developed, that should cover some of the deficiencies of SNMP [10], like scalability, flexibility and security.

Configuration Languages. Configuration languages are management systems that have a specific data description and configuration language. Those systems are normally based on representations with higher levels of abstraction than interpreted languages like Shell Scripts, Perl and Python, or are guided by standards as XML.

Among the tools used to describe the configuration of systems we can cite: PAN [5], Cfengine [3] and LCFG [1]. Each of them has its own characteristics, target environments and particularities.

Some of the difficulties of those systems arise with the need of learning to use their own specific configuration languages, that sometimes exceed two thousand configuration options [5]. Moreover, those tools do not use a unique information

[1] **I**ntegrated, **D**ynamic and **A**utomated **N**etwork **M**anagement and **O**peration **S**ystem

source to fully configure all the managed systems. Other lacks include flexibility and administrative granularity, not allowing, for example, the network manager easily extend the system to particular situations or needs of the local domain.

Distributed Management Tools. Distributed management tools allow a network manager, from a unique administrative point, to control and maintain systems and computers of a network. These tools presuppose that there is an information base that will enable the updating and manipulation of the configurations of distributed machines and services.

NetInfo [2], NetView [9] and OpenView [8] are examples of distributed management systems that allow the control of devices and services of an entire network. Almost all of these tools are designed for specific environments or need specific versions of server software, not being applicable to the general management of computer networks with diversified types of systems and devices or where managers do not want to change current systems interfaces.

Single-System Management Tools. Single-system management tools provide mechanisms for speeding up and simplifying the administration of a unique computer. Some of those tools have Web interfaces that allow a remote management of the computer.

SMIT[14], Webmin[13], Linuxconf[7] and CLIs[12] are examples of such systems. Each one has its target environment and its particularities. In essence, they are conceived to the same end. However, they do not fit for the simultaneous management of various system and do not use an unique information source.

Important Characteristics of Network Management Tools. Network management systems are tools designed to control and maintain machines and services in an administrative domain. Most network environments will have heterogeneous equipments and services. In this context, the use of the management systems goes from monitoring of devices to general configuration of a diversified set of programs and computers.

When we talk about an integrated, adaptable and automated management environment (or even in less integrated and automated situations), some characteristics should be considered in the project and development of tools that will have to manage diversified resources, probably in distributed locations or different network domains. Among the essential characteristics of an integrated management solution we can include: *a single information source*, that provides all data needed to manage network devices and services, avoiding data redundancies; *keep the current architecture of the systems*, not forcing the development of new versions or the modification of existing systems; *component-based*, allowing the easy and fast development of new components and be adaptable to the uncommon needs of different administrative domains; *possibility of distributed execution*, increasing the management range for wide spread systems or even to distribute the computing and data traffic through the available computational resources; *ease of use*, where the administrators have only to change data on the information source and they will automatically be considered by the affected systems; and *fault tolerance*, enabling the automatic detection and correction of network services availability problems, for example.

Other characteristics such as administrative granularity, security, performance and independence of programming language and operating system can increase the application and capabilities of an integrated network administration tool.An integrated and automated management system for the control of services and components of heterogeneous networks should provide as many of those characteristics and resources as possible.

Considering the tools previously referred, there are tools that cover some of those characteristics, however, none of the researched tools covers most of them. As will be shown in the following sections, `ida` covers all the essential characteristics and most of the other resources, cited on the previous paragraph.

3 Architecture of the System

3.1 Overview

`ida` intends to provide a simple, extensible and automated architecture for the integrated management of network services and computers. The tool should also give freedom and choices to the system managers, allowing them to keep using many of their management tools and tactics to control services, computers and devices.

Figure 1 presents the basic idea of the system. A single information source keeps the data needed to configure the networked devices and systems. This information is processed by `ida` to generate configuration files as expected by the controlled systems. The controlled systems do not need to be aware of the existence of `ida`. Thus, we have a simple and transparent architecture, viewed from the managed systems side.

3.2 Basic Architecture

`ida` is a modular system. Its components are classified as modules or agents. Modules are the components that directly maintain the configuration of services and devices, while agents control the execution of the modules and do maintenance tasks on the system itself.

There are different categories of modules and agents. Each of them has some specific characteristics and functionalities.

Modules. The basic task of these components is to keep the systems of the network up to date relative to the data in `ida`'s database. The modules are simple components (without any intelligence). When executed they only process data or execute commands associated to their tasks.

There are three types of modules:

Generation: generate adequately formatted configuration data from information stored in the database. The formatted information is put back into the database. There is typically one specialized generation module for each system (or group of similar systems) configured by `ida`.

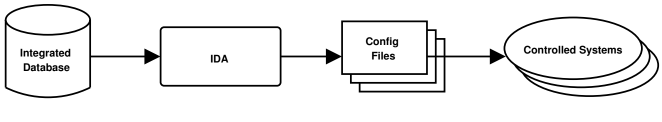

Fig. 1. The Idea

Update: copy the formatted configuration data from the database into the place where the affected systems expect them to be. Make sure the affected systems are aware of the changes, restarting them if necessary. They are very simple and easy to develop, because they basically do only two things: (1) copy configuration files from the database to a local filesystem, and (2) execute existing scripts (responsible for starting, restarting and stopping a service) to put into effect the new configurations.

Execution: provide remote or distributed execution of commands; an application example is the recuperation of a system that is not accepting user login anymore.

Service Agents. Control the activation and execution of modules. This is basically done by monitoring data modification notifications on the system's database. The service agents look only after an update state key value on each module's configuration entry. If the key was updated, then the module will be executed.

A service agent can control several modules. Different service agents, on different machines, can manage different (or even equal) modules that maintain a specific service.

Each managed machine of the network will have a service agent, except devices that can only be managed remotely. In the farther case the service agent can be located on any machine of the network.

Figure 2 shows the base architecture of the system. It is composed by configuration generation and update modules (that maintain services and devices configurations) and service agents, which control the execution of modules.

Operation example. One simple example of the operation of modules and agents is the DNS service maintenance. A generation module for DNS generates the name resolution tables (and related configuration files), adequately formatted, using the information provided in the database. Those formatted configurations are copied to the filesystem where the DNS service is running by an update module. This module would have to restart DNS daemons to put the new configurations into effect. Both modules would be run automatically by service agents every time modification on related data is detected on the database.

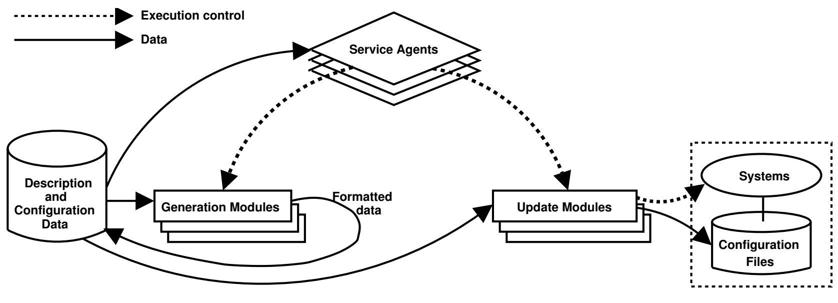

Fig. 2. ida's basic architecture

3.3 Other Agents

The basic architecture shown in the previous section do not deal well with issues like automated data modification detection, fault tolerance, location of the system's components, migration of services and immediate updates notification. This section presents the system's architecture in more details. New components will be presented, to deal with the mentioned problems.

Figure 3 presents a general view of `ida`'s architecture. New components are the local, global and monitoring agents, explained below.

Each component of the system works independently of the others. They all search their data and execution options directly from the system's database. With this architecture it is possible to distribute the work load of the system and the data traffic among the components and the database (see a simple example on section 3.5).

As the components can be distributed over the network, locality and migration problems arise. The **local agents** exist basically to solve these two problems. To do so, there is one local agent for each managed computer. Local agents are responsible for starting `ida`'s local components, like service agents. Each local agent configuration entry, on the database, tells which are the components that should be executed.

Local agents have an unique identification and locality information, consequently allowing the location of service agents and modules of the system. This information will be used, for example, by the global agents to support automatic activation of updating processes.

We can use the local agents to migrate a service from one system to another. Removing the description of a service agent from the entry of a local agent in the

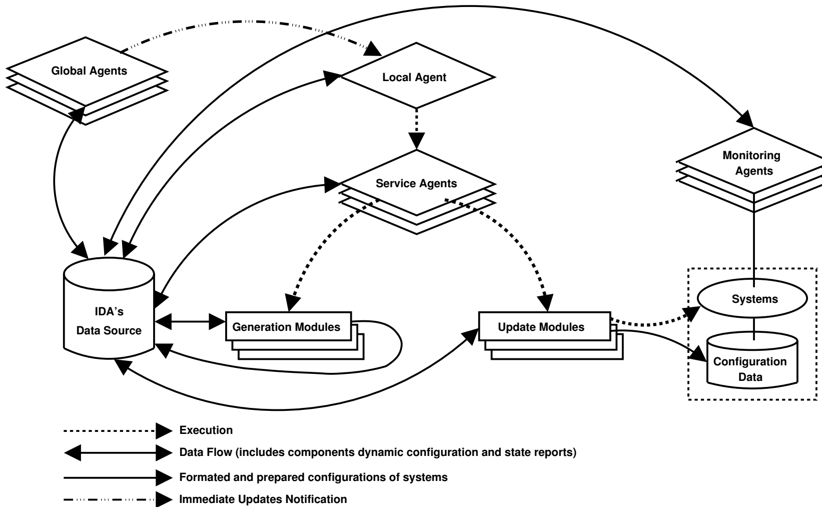
Global Agents
Local Agent
Service Agents
Monitoring Agents
IDA's Data Source
Generation Modules
Update Modules
Systems
Configuration Data
Execution
Data Flow (includes components dynamic configuration and state reports)
Formated and prepared configurations of systems
Immediate Updates Notification

Fig. 3. System's overview

database, that agent and the corresponding server will be stopped. Including it in the local agent description of another machine, the service will be started there by its local agent.

Another use of local agents is to simplify the service agents by controlling their polling and immediate activation mechanisms (by default the agents work by polling intervals). However, immediate activation is an auxiliar updating mode, working by the automatic invocation of an updating notification. Each local agent has a listening thread waiting for immediate activation signals.

As `ida` has a single source of information, there can be an increase on the probability of misconfigurations and incorrect data references among different systems description entries, that could affect many systems. To reduce the database error probabilities we created the **global agents**. They do periodic checkings verifying data completeness and validating relationship references. All references are evaluated and notifications are sent to the system managers when inconsistent configurations are found. This helps to identify and rapidly correct many possible management mistakes that can be done by an administrator, like typing errors and wrong data references caused by lack of attention during data modifications.

Global agents are also responsible for signaling data updates to local agents. This is done by verifying the data tree, looking for updates. The global agent then generates a new update key in the affected modules entries in the database. The update key of each module is used by service agents to decide when to run a module. The global agent knows which modules have to receive a new update key by looking at the service's entry. It tells who are the modules that use its data to maintain the corresponding service.

The **monitoring agents** emerge as a way of provide information about the state of network services. The resulting monitoring data can be used by global agents for automatic service migration decisions, service state reports and to generate alerts for system managers.

3.4 Other Features

Configuration Overlaps and Mappings. To have general configuration entries for each set of machines or services, where specialized configuration can easily we extended, we provide overlap of information. On each services entry we can define a different and refined configuration, using and/or overlapping general configuration options. In most cases, this reduces also the amount of data to be managed on the service's entry. Each redefined option on the service entry will overlap the general one. New configuration options will be added to the general ones.

Relationships between machines and its services should be maintained in an abstract and transparent way. To provide this characteristic we use mapping entries. They provide a way of establishing or changing relationships of machines and their services. The modification of a mapping entry, for example, will cause automatic updating of all systems configurations that maintain any data relationship with it. Each module, responsible for the configuration of a specific system, that uses the information of the mapping entries, reads the mapping informations when it starts formatting the data needed to configure its managed system.

Automation. Automatic detection of data modifications and activation of updating components is something that provides mechanisms that ease the control

of the management system. Thus, ida's normal management needs only modifications on the database. When someone updates the database, the modifications are automatically detected by global agents and the updating processes of services or systems affected by the new changes are started and transparently done.
Code Maintenance. A complementary task of local and service agents is the maintenance of the execution code of their subordinate components. This is another tip on system self-maintenance and dynamic migration or allocation of components. If we update the source code of a module, for example, the new code will be loaded to a main repository and afterwards it will be downloaded and run by the service agent responsible for it. To migrate one component we need to: (1) remove its reference from the current father agent; (2) put its reference on the configuration entry of a second agent. The former agent will stop the service on the original machine, while the further will download and execute the code of the new component, consequently, starting the service on the new provider machine. The source code of the components can be maintained on the unique source of information or in a reference location (accessible through HTTP or other commination protocol).

3.5 Application Example

This section presents an example of application of ida to the management of network services and computers. Figure 4 illustrates an application of some ida's agents and modules.

The environment shown on figure 4 is composed of two network servers, one client and one administrative machine. In each computer there is a local and a service agent, despite the generation and update modules.

Server 1 is running DHCP and DNS services. Thus, the service and local agents of that machine control update modules of these services. The generation modules, which will create the configuration files for both services, are running on

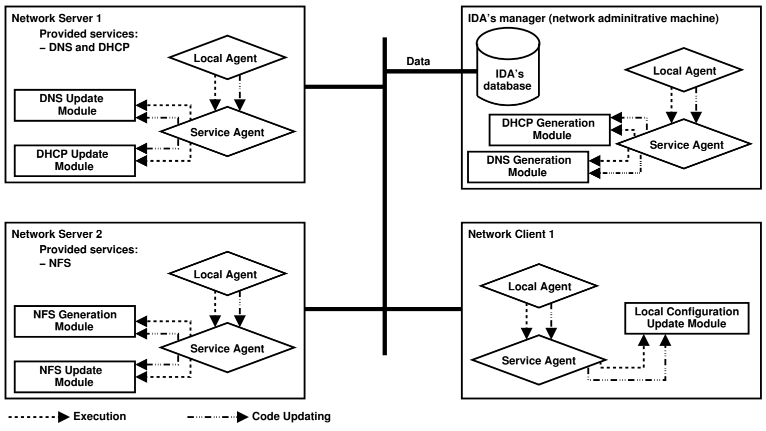

Fig. 4. Simple example of ida's usage

the administrative computer. In the case of the second server (2), both generation and update components for NFS service are running on the local system. Client machine 1 runs just its machine configuration maintenance module. This example gives a basic idea of the system's functioning and distribution possibilities of the components across the networked machines.

Suppose that the network manager is going to include two new machines on the network. Both of them will be registered on the DNS server (1), allowed to use dynamic configuration from the DHCP server (1), and allowed to mount remote file systems from the NFS server (2). After including the new machines into the database, as soon as the service agents controlling the generation modules run they will detect the new data updates. The service agents of administrative machine and server 2 will immediately run their generation modules, affected by the data changes on the database. By this time the update modules are not going to be ran, because the new formatted data is not generated yet. The corresponding service agents will only run the update modules after the new formatted data for the respective services has being included in the database. When the update modules are executed they will copy the new configuration files from system's database to the location where the respective services expect them to be, restarting the services to put into effect the new configurations.

4 Current State and Availability

Implemented Components. The current prototype contain the developed components of the system: global (divided in control and maintenance agents), monitoring, local and service agents; generation and update modules, for various services and for machines configuration; and database management tools. More details about each of them can be found in [11].

Other Implemented resources. Despite the components previously mentioned, there are other implemented resources. This include: configuration overlap, system's components activities report and services monitoring reports. These resources are available within the software packages. More details about them can be found in [11].

Technologies Used. In the current version of the system, access policies (administrative granularity, security and components access) are controlled with the Lightweight Directory Access Protocol (LDAP), using OpenLDAP. Thus, `ida`'s modules can be written on any language that have support for the LDAP protocol (you can see some of the languages that support this protocol in [11]). The database distribution and replication are also controlled with the resources provided by OpenLDAP.

The base of the system has being developed with Perl. More details about the why's of the chosen language can be found in [11]. There you can also find more information about the database architecture and organization, among other information.

Availability. One version of the system can be found in `http://www.inf.ufsm.br/~kreutz/files/ida.tar.gz`.

5 Validation Tests

In this section we present three basic operation tests. The first one was looking over the correct functioning of system's components and testing automatic detection of data updates (in the database) and consequently activation of updating processes for the affected services. The second and third tests were to see the working of manual and automatic migration/allocation of services.

Updating Detection. Updating detection consists in automatically detect information changes in the database and start the processes that will update all the configuration of the systems affected by the new data. Thus, the test consists in modifying some information in the database and see if it will be automatically detected and spreaded to the related services.

The test was done by using generation and update components for DNS and DHCP services. It took the following basic steps: **(1)** assign DNS and DHCP generation and update modules to service agents; **(2)** assign the service agents to local agents (on the machines where the services should run and be maintained); **(3)** change an IP address of the database (looking for the accuracy of the system over one of the smallest data change granularity); **(4)** check on the service providers if the update was detected and spreaded to the local systems (services).

After change of the IP address on the database DNS and DHCP services were automatically updated. The detection of new data and the updating process took a few seconds. This time was mainly due to the 10 seconds polling interval of the global agent and the time needed for the service agents detect the new update keys and run the corresponding modules.

Migration of Services. The migration, or allocation, of services can be done in two ways. The former one is done directly by the system administrator. In this case, the modules or service agents are manually removed from one local (or service) agent's entry and included on the entry of another agent. This will cause the stop of the service on the original machine and start it on the new machine, where the second local agent runs. To do so, the first agent will notice the remove of one of its components and will remove it from its local list. The second agent is going to detect the new components under its domain and take the necessary measures to activate and execute them. On the tests executed, all operations of this kind succeeded well. General components, modules and service agents were allocated from one machine to another without problems.

The second way of migration is done by using the global agents. In this case, they will identify migration needs and take the necessary measures to proceed them. This will mainly be used in failure cases, were a global agent analyses services state reports and with them decide if the service is working or not and if a service migration should proceed or not, based on migration policy rules. To see if this second option of migration really works, we did some simple tests. One of them took under consideration the DHCP service. We simulated a service response failure, were the monitoring agent should detect and report it to the database. After the failure detection, a specialized global agent should automatically analyse the monitoring state reports and, based on migration policies (for more details see [11]), migrate the service to an auxiliary machine. Next are presented some steps took during the test: **(1)** while the services were working, a connectivity/response fault was caused on the environment; **(2)** the global agent, looking over the monitoring reports, verifies a fault on a service response and analyses the migration polices; **(3)** as soon as the global agent detects the existence of a migration policy, a migration process is started; the service's update module is removed from the actual service agent entry and included in a new entry, on a service agent on the auxiliar machine - the new server. This simple experience has shown the use and functioning of the automatic migration of services. A similar example, yet using the DNS service, is also shown in [11].

As a general overview, the migration of services, if carefully used, can provide a way of easily maintain the availability of network services. This resource can also diminish some of the worries and headaches of network managers.

6 Conclusion

The management of computer networks is an area still under expansion. More and more networks are built and interconnected every day. New equipments are also added to local networks with a certain frequency. With this scenario, increasing efforts are needed to manage and supervise those networks. Yet, we need also specialized and high level tools that are able to provide flexibility, simplicity, scalability and any resource or characteristic that can help do improve the management of devices and systems in general.

One way to provide more effective and largely applicable tools is through research and identification of the main necessities of heterogeneous administrative domains and systems. Integrated solutions are an option for these environments. With this in mind we came up with our management architecture.

This work showed some important characteristics in integrated management systems. It also presented the project and development of a system that is capable of reducing some deficiencies of other management tools. The resulting is `ida`, an integrated and automated architecture for managing computers and services of a network. This system has many of the characteristics considered important in a tool that aims to cover a large number of administrative domains and every day situations of network administrators. Yet, providing useful and simple mechanisms for managing, in an integrated and automated way, the most diverse kinds of services and devices of a network.

The validation results showed that the system is applicable to real environments and constitutes a good tool for managing different administrative contexts. Its use goes from simple local networks, that have only a dozen of computers, to more complex networks, that need to control hundreds of computers and/or many different environments. This can be archived because the developing of a new update module, for example, for a new operating system (or service) is relatively easy and fast (see section 3.2).

Future Work. We have just developed a first prototype. More tests, improvements and new features should come with future work, like, for example, auto-diagnosis system, graphical interface, SNMP management modules, integration with tools like Cfengine and LCFG, integration with monitoring systems, integration with user control systems, storage of system code and database modification history, description of network devices in general, general software installation management, integration with fault prediction systems, management of general network devices and XML representation. More details about each one of them can be found in [11].

References

1. Paul Anderson. Towards a High-Level Machine Configuration System. In *8th USENIX Conference on Large Installation System Administration (LISA)*, pages "19–26", Berkeley, CA, 1994. USENIX Association.
2. Apple Computer, Inc. Mac OS X Server 1.x: What Is NetInfo?, 1999. `http://docs.info.apple.com/article.html?artnum=60038`. Last access: February 2004.
3. Mark Burgess. A Site Configuration Engine. In *USENIX Computing systems*, Norway, 1995.
4. J. Case, R. Mundy, D. Partain, and B. Stewart. Introduction and Applicability Statements for Internet-Standard Management Framework. Technical report, Dec 2002. `http://www.faqs.org/rfcs/rfc3410.html`. Last access: October 2004.
5. Lionel Cons and Piotr Poznanski. Pan: A high-level configuration language. In *Proceedings of the 16th Systems Administration Conference (LISA-02)*, pages 83–98, Berkeley, CA, November 3–8 2002. USENIX Association.
6. R. Enns. NETCONF Configuration Protocol. Technical report, October 2004. Internet Draft. `http://www.ietf.org/internet-drafts/draft-ietf-netconf-prot-04.txt`. Last access: December 2004.
7. Jacques GÉLINA. Linuxconf - Linux Administration Made Easy!, 2003. `http://www.solucorp.qc.ca/linuxconf/`. Last access: March 2003.
8. Hewlett-Packard Development Company, L.P. HP OpenView Management Software, 2004. `http://openview.hp.com/downloads/index.html`. Last access: December 2004.
9. Stephen Hochstetler, Peter Glasmacher, Morten Moeller, Pritesh Jewan, Mangesh Pathre, Jonathan Polacheck, Varatharajan Sampath, and Helmut Ziller. *Tivoli NetView 6.01 and Friends*. IBM RedBooks. IBM International Technical Support Organization, 1 edition, 2000.
10. Guofei Jiang. Multiple vulnerabilities in SNMP. *IEEE Computer — Security and Privacy*, 35(4):2–4, April 2002.
11. Diego Kreutz. *Um Sistema Integrado para o Gerenciamento Automatizado de Redes de Computadores Locais*. Master dissertation, Universidade Federal de Santa Maria — UFSM, Santa Maria, RS, Brasil, 2005.
12. BJ. Lee and TS. Choi. Cli-based mediation mechanism using xml for configuration management (draft-lee-xmlconf-xcli-00.txt), 2002. `http://www.ietf.org/internet-drafts/draft-lee-xmlconf-xcli-00.txt`. Last access: November 2004.
13. Susanne Schmidt. Web Based System Administration with Webmin, 1999. `http://www.heise.de/ct/english/98/14/068/`. Last access: December 2004.
14. Susan Segura. System Management Interface Tool (SMIT). Technical report, IBM, November 2000.