INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
Int. J. Network Mgmt 2009; **19**: 237–252
Published online 22 July 2008 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/nem.700

OpenRSM: a lightweight integrated open source remote management solution

Y. Karalis, M. N. Kalochristianakis, P. Kokkinos and E. A. Varvarigos*†

Department of Computer Engineering and Informatics, University of Patras, and Research Academic Computer Technology Institute, Patras, Greece

SUMMARY

The management of the corporate information technology (IT) environment is rapidly increasing in complexity as server logic architecture becomes more distributed and the number of entities deployed increases, forcing enterprises to resort to thick, complex and expensive high-end integrated systems and network management solutions. Investing in such systems can be inefficient for small and medium corporations, since the vast majority of management tasks performed are routine tasks, while personnel specialization requirements and costs are high. At the same time, the open source community has not yet produced a reliable and complete system and network management solution. Even though there are open source initiatives specializing in specific fields of remote management, such as network management, there has been no integrated open source solution yet. This paper introduces the Open Source Remote Systems Management (OpenRSM) platform. OpenRSM is an integrated remote management system created by integrating individual specialized open source management initiatives and significantly augmenting them to support additional functionality, so that a complete lightweight system and network management solution is produced. The system implemented facilitates daily management by providing an efficient, simple and adaptable environment for the majority of management operations. Copyright © 2008 John Wiley & Sons, Ltd.

1. INTRODUCTION

The total cost of ownership (TCO) for assets such as workstations, installed software, configuration, and custom code is often far higher than the initial purchase or development cost. The difference between the two costs can be very significant [1] for corporations, and thus every IT organization aims at minimizing the TCO for such assets [2]. To this end, organizations employ enterprise management systems (EMS) [3] that are generally scalable, well organized, and cost far less than the TCO of the corporate IT assets. Systems and network management has always been a critical field for IT organizations, since it facilitates management tasks and provides important economies of scale [4].

The EMS market is dominated by few but well-known products, most of which are released by major enterprise systems constructors [5]. Such organizations carry profound system-level knowledge and experience gained over years of producing management platforms. EMS systems offer a variety of highly specialized features and capabilities that extend the network and systems management functionality [6]. They cover areas such as systems and assets management, storage management, software development, security and business integration, and offer an ecosystem of features, services and tools [7]. Competition and marketing strategy has led EMS vendors to introduce and advertise specialized, sophisticated

*Correspondence to: E. Varvarigos, Research Academic Computer Technology Institute, N. Kazantzaki Str., University Campus, 26500 Rion, Greece.
†E-mail: manos@ceid.upatras.gr

Received 7 March 2007
Revised 6 February 2008
Accepted 13 May 2008

features such as end-to-end root cause analysis, return on investment optimizations [8] and time savings estimations [9]. Since EMS are oriented towards the enterprise scale, where marginal profits are optimized and competition is minimum, the high-end EMS market is characterized by strong introversion in contrast to other IT fields where many viable alternatives exist, some of which are based on open source software. This is one of the factors that led the EMS market to be expensive in terms of licensing and support, at least for small-scale organizations.

Today the picture is changing [10]. The exponential rise in computing power reduces the price of server hardware and software and the maturing [11] of the technologies involved is exposing the enterprise world to whoever may be interested in IT. One does not need to own a high-end dedicated server machine with large physical memory in order to run a heavy application server. The enterprise software market is also opening to the open source community and so too is the market of enterprise-supporting tools, such as resource management tools. The EMS market has not been left unaffected by this trend; high-end EMS providers respond to the changes by releasing express versions of their software, aiming to penetrate the new market of mid-sized corporate management systems before open source systems do. However, they are criticized for being overly complex, inflexible [12], and difficult to deploy, learn and manage. Even if many EMS features and tools are technologically advanced, they are not in the first line of needs for small organizations. Root cause analysis or fault localization may employ advanced technologies such as artificial intelligence, model traversing or fault propagation [13]. These solutions, however, may not be that useful in managed environments that are small and structurally dynamic. EMS subsystems are tightly structured and thus hard infrastructure changes result to inefficiency and flexibility loss. Moreover, a management infrastructure based on a high-end EMS will most probably require specialized technical knowledge and devoted personnel in order to return its investment, and most small organizations cannot afford these costs.

The open source community is, however, still rather immature as far as remote management systems are concerned. There are only few network monitoring system (NMS) projects under the SourceForge Open Source project hosting site and only a small number of them are on the road to maturity. As far as integrated systems and network management tools are concerned, there is none at the moment. The state of the European open source management tools market is depicted in the 'catalogue of available Open Source tools for the PA' [14], where only a couple of network management systems appear in the administration and management tools section.

The Open Source Remote Systems Management (OpenRSM) platform is a pioneering initiative aiming to open the lightweight EMS market to open source tools. The initiative has been based on the observation that most of the components comprising an open source EMS tool can be developed by extending existing rated components offered by the open source market. Even though mature open source EMS tools are lacking at the moment, the basic market needs for integrated and free enterprise-level EMS tools, and the existence of reliable freely redistributable tools that cover the basic requirements of EMS subsystems, will ensure the creation of a thriving open source market in this area in the near future [11]. OpenRSM aims to earn a place in the area of small management systems, not by competing with high-end EMS in terms of feature richness and technology solutions, but by offering an efficient infrastructure management solution that is suitable for the highly dynamic and diverse small corporate environments, where high-end EMS are heavy and difficult to manage, and their technology is poorly utilized.

OpenRSM utilizes the power of third-party tools that have been significantly enhanced and appropriately altered in order to deliver inventory and asset management, software delivery, remote desktop control and network monitoring services, all integrated into one system. A great deal of effort has been invested in system integration, extending from database back-end migration, to server and web content porting, and agent logic concatenation. The combined subsystems have been modified so that information can be shared among them. At the same time, the overall system has been designed so that it is subject to the minimum possible set of limitations. Its capabilities extend to managing any stations reachable through standard IP connectivity in a secure manner. The architecture at the server tier has been kept open and thus adaptable to any specific needs and business models [15]. Extra care has been taken so that OpenRSM can manage stations hidden behind the NAT protocol, using a proxy server developed

DOI: 10.1002/nem

for that reason. The system also supports multi-platform systems management and provides a multilingual user interface.

The remainder of the paper is organized as follows. In Section 2 the architecture and individual components are presented in terms of design and functionality. Section 3 discusses security considerations. Section 4 describes the tests performed in order to evaluate OpenRSM in realistic usage scenarios. Section 5 presents our conclusions and directions for future work.

2. THE OpenRSM SYSTEM FOR REMOTE MANAGEMENT

2.1 Design and system components

The design of the OpenRSM system considered several implementation strategies [16–18] and development solutions [19], ranging from peer-to-peer technologies and layered server side middleware [20], to more traditional client–server approaches. In principle, OpenRSM needed to be simple and lightweight so that it can be used by end users who are not specialized in the use of management or asset-reporting tools. OpenRSM has also been designed for fast and automatic deployment in order to cover the needs of administrators who manage very dynamic environments. The OpenRSM system adopted the open source development model so as to exploit the dynamics of open source projects on management technologies and to gain value from integration. As mentioned above, even if there is no complete, integrated open source EMS, the related technologies have matured to the point that the open source community can provide all the necessary components [21]. Several open source projects have been examined in order to find the most appropriate open source management tools available for the purposes of OpenRSM. During the development of the project and the compilation of the present article, the authors have not been aware of the existence of any other integrated open source EMS system.

The architecture of the OpenRSM has been chosen to be modular in order to follow the logical categorization of the entities involved [22] and to favour integration with other open source management tools. Thus, OpenRSM has been based on the agent–server model. The OpenRSM agents model abstract manageable entities that convey administrative actions from the OpenRSM server. The agents support many different operating systems. Administrative actions originate from the OpenRSM management console, the tool exposed to the end-users and the administrators of the service; the users perform administrative tasks but cannot manage the OpenRSM services. Management commands are conveyed to the OpenRSM server and then to the agents. The architecture of the OpenRSM system is presented in Figure 4.

The OpenRSM server (Figure 1) schedules and synchronizes the execution of jobs defined by users. Each job corresponds to a distinct administrative task manageable within the OpenRSM system. Jobs are entities abstracted and designed with the use of the object-oriented model. The OpenRSM design was based on the principle that jobs play a central role in terms of usability, design efficiency and system scalability. Thus jobs are designed to behave as standard abstract system tasks, for example, inventory, remote control, remote command, or as reusable user-created objects. Jobs can themselves be managed by administrators, decoupling their creation and execution stages. Jobs are created by the end-users by using the OpenRSM management console, also called administration console. They are then submitted to the OpenRSM integration server and are finally executed by the OpenRSM agents at the managed stations. The integration server is the front-end of the OpenRSM server logic. Its purpose of operation is to schedule job execution according to user commands, prepare (wake) the agents for job execution, concentrate access to back-end services, and interact with OpenRSM proxy modules and the database. The back-end of the OpenRSM server consists of web server(s), database server(s) and proxy server(s).

After a job has been submitted, the integration server checks its syntax and dispatches it to the appropriate station(s) for execution. The integration server wakes the agent(s) using a specific handshaking protocol. Details about this communication are presented in following sections. If there are other complementary tasks that must be executed as a result of job execution (e.g., due to job dependencies), the integration server makes sure they are also dispatched. Even if there are no dependencies among user

DOI: 10.1002/nem

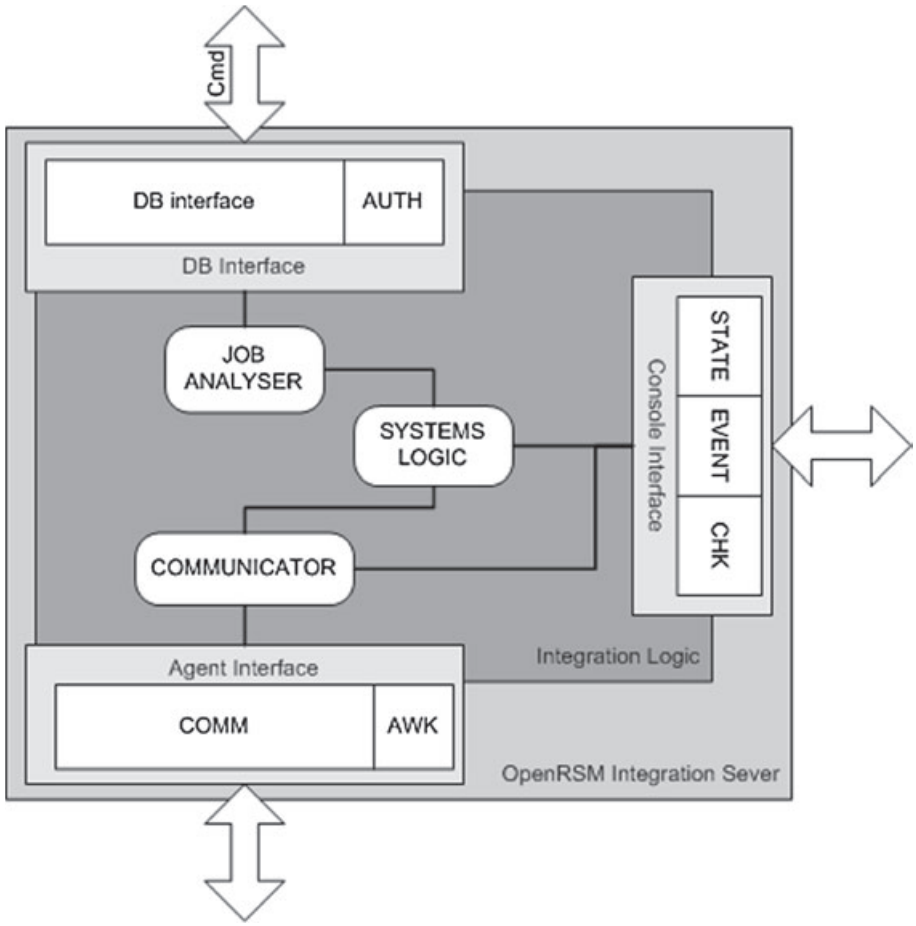

Figure 1. OpenRSM integration server architecture

commands (see Section 2.4), inconsistencies resulting from arbitrary job execution have to be resolved. The OpenRSM integration server is also responsible for monitoring the job execution progress. It sends event and logging information back to the management console so that all management scenarios can be handled. It is also responsible for communication with the proxy server modules. When jobs reach the execution stage they are most likely served by one of the subsystems incorporated in the OpenRSM system. For instance, if the job is an inventory query the agent registers inventory information about the station it resides on and sends the information to the inventory web application, hosted by the web server of the OpenRSM system. Each server subsystem is presented in detail in the corresponding paragraph in this section. The OpenRSM approach is to use double-layer server logic. Jobs are served in two levels: the actual level of execution and the abstraction layer of job checking, scheduling and routing.

The OpenRSM project is hosted in SourceForge [14]. The development team numbers three developers. The code repository of the project is available for anyone to download and contribute to.

2.2 The OpenRSM server subsystems

This subsection presents the four OpenRSM core server-tier subsystems: inventory, deployment and software distribution, network monitoring, and remote control [23]. Subsystem integration software has been selected on the grounds of functionality, maturity, compatibility and interoperability. Additional

DOI: 10.1002/nem

features such as the discovery of managed stations and the creation of dynamic groups are also presented.

Inventory

Inventory and asset management constitute the core of remote systems management. Asset management provides large organizations with the ability to gather information on the hardware and software of the managed workstations. This functionality is significant, because it provides the necessary data for effective troubleshooting, facilitates the planning of upgrades, increases company control and security and, last but not least, helps keep the TCO under control by means of accurate determination of capital and administrative costs. Asset management is usually realized through a silent software agent loaded on the managed system. The agent retrieves information about the system and presents it in a user-friendly way to administrators through the appropriate EMS console. The technologies used, namely CIM/WBEM [24], are mature enough to provide vendor interoperability.

The OpenRSM platform is based on the Open Source OpenAudit [25] inventory software. OpenAudit is a web application written in PHP, and is designed to read assets/inventory information and store it into a MySQL database. The idea is that users need to manually configure and run the audit software locally on their workstation; the audit software reads information about their system and posts it using the http protocol to the web application. Inventory information is then stored in the database and presented through a web interface. The OpenRSM inventory subsystem builds upon OpenAudit by enhancing both automation of use and functionality. The audit software is integrated with the OpenRSM agent module and can be run from a remote administrative location. Thus, physical presence is not necessary and stations can be audited remotely. Moreover, the database schema is integrated with the rest of the information stored, making it usable by other components of the OpenRSM system. The dynamic groups feature takes advantage of this fact; the administrator is presented with the capability of creating groups of stations that share one or more common characteristics so that they can be treated in a uniform manner. Integrating database schemas makes asset management information available to all parts of the OpenRSM.

Software delivery and deployment

The software distribution and deployment functionalities facilitate the management of already installed software, or of software that is to be installed on workstations within an administrative domain. Software management is time and resource consuming, in terms of experienced and specialized man hours. The lack of software management tools has led system administrators to practices that hinder flexibility. Excessive security is often imposed at the expense of user needs and convenience. However, administrators are required to know and manage the software of their managed stations, along with every other IT infrastructure asset.

The functionality provided by the software delivery subsystem is transparent to the users. The administrator simply chooses the software to be delivered and designates the path to the executable, or the link from where it can be downloaded. Then, the software is uploaded and registered within the repository, and is subsequently delivered by the server. When software is delivered, the designated installation file runs, and users may be required to complete the installation procedure. If the silent mode of operation is chosen, however, which is a decision made by the administrator, then the installation is not interactive and users are not distracted in any way.

OpenRSM uses an extended version of the Windows-get [26] open source tool, specifically enhanced in order to meet the requirements of the integrated graphical delivery subsystem of OpenRSM. The delivery subsystem is complementary with the inventory subsystem; administrators may use the inventory system in order to supervise software distribution and they may use the delivery subsystem in order to install/uninstall desired modules. The OpenRSM delivery subsystem is fully automated and complete in supporting the installation and uninstallation of software on one or many workstations, silently or interactively. Silent is the type of installation/uninstallation where no user interaction is needed. This is a basic mode of software deployment for OpenRSM, since silent installations/uninstallations are very

DOI: 10.1002/nem

useful for routine administration. The OpenRSM delivery exploits input information about the packages used, such as the executable location, name, software description, download URL, application type, uninstallation executable, etc. It is capable of running the installation or uninstallation executable using various switches, with respect to the type of installation/uninstallation the program itself supports.

Windows-get functionality has been extended to meet the OpenRSM requirements. To start with, it has been extended in order to support software uninstallation, implemented by calling the system's uninstaller program and by providing it with appropriate information about the type of uninstallation to be performed. For Windows systems this piece of information is typically stored in the Windows registry. The rest of the information needed has been integrated in the OpenRSM database and is checked and updated as needed after every installation or uninstallation. The way Windows-get handles archive files has also been extended by the OpenRSM delivery system. Windows-get inherently handles archive files in no other manner except opening them with the appropriate program. The OpenRSM delivery system is capable of opening the majority of archive file types and then installing the software they contain. The OpenRSM delivery system also supports optimized execution methods for most installation package types supported in Windows platforms. Another enhancement made is that the OpenRSM delivery system does not rely on Wget, the well-known *nix-world network download utility, which has been replaced by custom code [27] designed to provide full control over the delivery procedure. This permits, in addition to other advantages, resuming software delivery. If a software delivery command fails due to an error, it will resume when the communication is re-established. This may be very useful when large software modules are deployed.

Network management

Network management systems (NMS) are essential tools for remote systems management and are capable of managing active network elements using the SNMP protocol. The OpenRSM remote management system integrates with the NINO open source NMS tool. NINO has been integrated in terms of database, web server components and business logic with the rest of the OpenRSM components. Changes in the OpenRSM environment are instantly reflected in the monitoring output. NINO is a full-featured network management system; it utilizes SNMP and WMI [28] technologies for the provisioning of rich real-time monitoring information for stations and network active elements. The list of features includes network discovery using various methods, events (that is, traps), monitoring presets and groups, various presentation methods (web interface device browser, reports, applet graphs), various utilities, such as MIB browser, snmpwalk, service response meter (HTTP, FTP, POP), and other useful features.

OpenRSM is designed to extend network management to network segments hidden behind NAT by installing one lightweight network monitoring server per hidden network. When the management task concerns stations located outside NAT segments, the OpenRSM server is capable of performing it, whereas when it concerns stations hidden behind NAT the management task is redirected to the respective network management server.

Remote control

OpenRSM uses the UltraVNC remote control package to deliver the remote control service. OpenRSM is capable of starting the UltraVNC server at a managed station or at a group of managed stations. The UltraVNC viewer is then started in order to connect to the managed station(s). If the UltraVNC server remains idle, it is closed after a configurable timeout period. OpenRSM takes full advantage of the features of the underlying open source tools. The remote control server is awakened by the agent after the remote control job has been delivered; next, the agent calls back the administration station. Thus no synchronization failures may occur, since the server is guaranteed to have started when the remote control client (viewer) initiates the connection. Synchronization also enhances security, since the server wakes when the agent has been informed of a new connection request. The server sleeps again after a specific and configurable amount of time. Besides the above, the remote control software has been configured to ask the users of the managed stations for permission whenever a connection is to open, in order to avoid unwanted remote access.

DOI: 10.1002/nem

OpenRSM proxies

OpenRSM has been designed to deliver remote management services in all environments. In order to meet the requirements posed in the case of networks that are partially exposed to the Internet, such as networks running under the NAT protocol, a proxy server has been developed. The role of this server is to receive connections that cannot be established directly, and forward them to the desired agents. In the case of NAT, forwarding the downlink traffic is sufficient, since the agents can send traffic directly to the OpenRSM server. This is widely known as half-proxy operation. In cases where the agents are completely isolated, the OpenRSM proxy is designed to operate as an access concentration server, otherwise called full proxy, which handles all uplink and downlink traffic. Proxies are also designed to be able to communicate among themselves (cascaded proxies), and thus trees of proxies that concentrate access can be deployed. Full proxy functionality and cascaded proxies have been included in the design.

2.3 The OpenRSM agents

The OpenRSM agents (Figure 2) are the OpenRSM client modules residing in the end-user workstations. Their functionality is limited to the execution of commands sent by the server, and they do not interact with any module of the OpenRSM server system unless it is absolutely necessary. The needs for uniform logic design and security converge to this implementation; the execution of each job triggers communication with the integration server and, from there, with the appropriate server subsystem. The server subsystem controls the communication and performs all the complementary actions and database transactions. Agents are implemented by integrating subsystems corresponding to different OpenRSM functionalities: integration logic, communication with the server, system-dependent execution logic and agent type implementation.

The fundamental agent module implements the communicational logic. As will be described in Section 3, this part of the agent ensures consistent and secure communication through a handshaking protocol and wake/sleep mechanisms. The remaining modules composing the OpenRSM Agent are the subsystems that implement the job execution logic, namely the asset management, network monitoring, remote

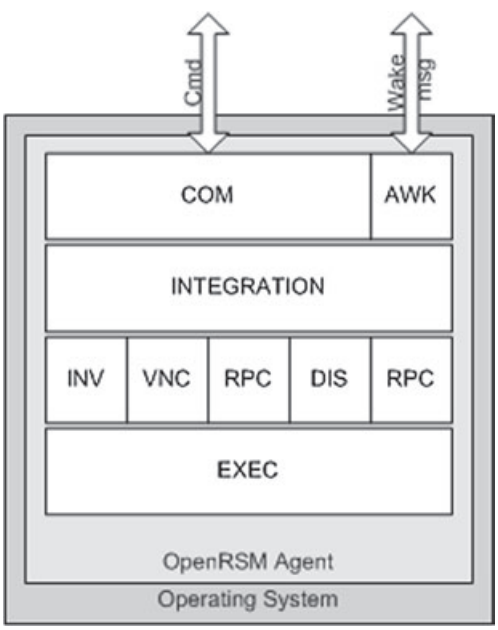

Figure 2. The OpenRSM agent module

DOI: 10.1002/nem

control, discovery and remote command. Whenever possible, subsystems take advantage of existing software or other open source agent modules (e.g., the inventory subsystem uses the OpenAudit agent for asset retrieval, as described earlier). The job execution subsystems integrate with the communication logic so that all job execution stages can be monitored by the integration server.

The OpenRSM system is capable of managing both Windows and *nix systems, through corresponding agent distributions that take into account the characteristics of each platform. Each distribution includes different agent flavours that correspond to different types of usage; the agent can be executed as a background process for silent operation, as a graphical application user for verbose interaction, as a service, or as a console application.

2.4 The OpenRSM management console

OpenRSM provides a controlling interface (Figure 3) that can be used by the administrators to control all the subsystems and their interactions. The design has focused on synthesizing the independent functionalities provided by the subsystems in a user-comprehensive and effective manner, and on the provisioning of additional supervisory functionality.

The OpenRSM management console exposes a multilingual control environment. The console can send any system command that is supported by the operating system of the managed stations. Commands accept parameters related to CPU priority, type and user visibility. However, the starting point for

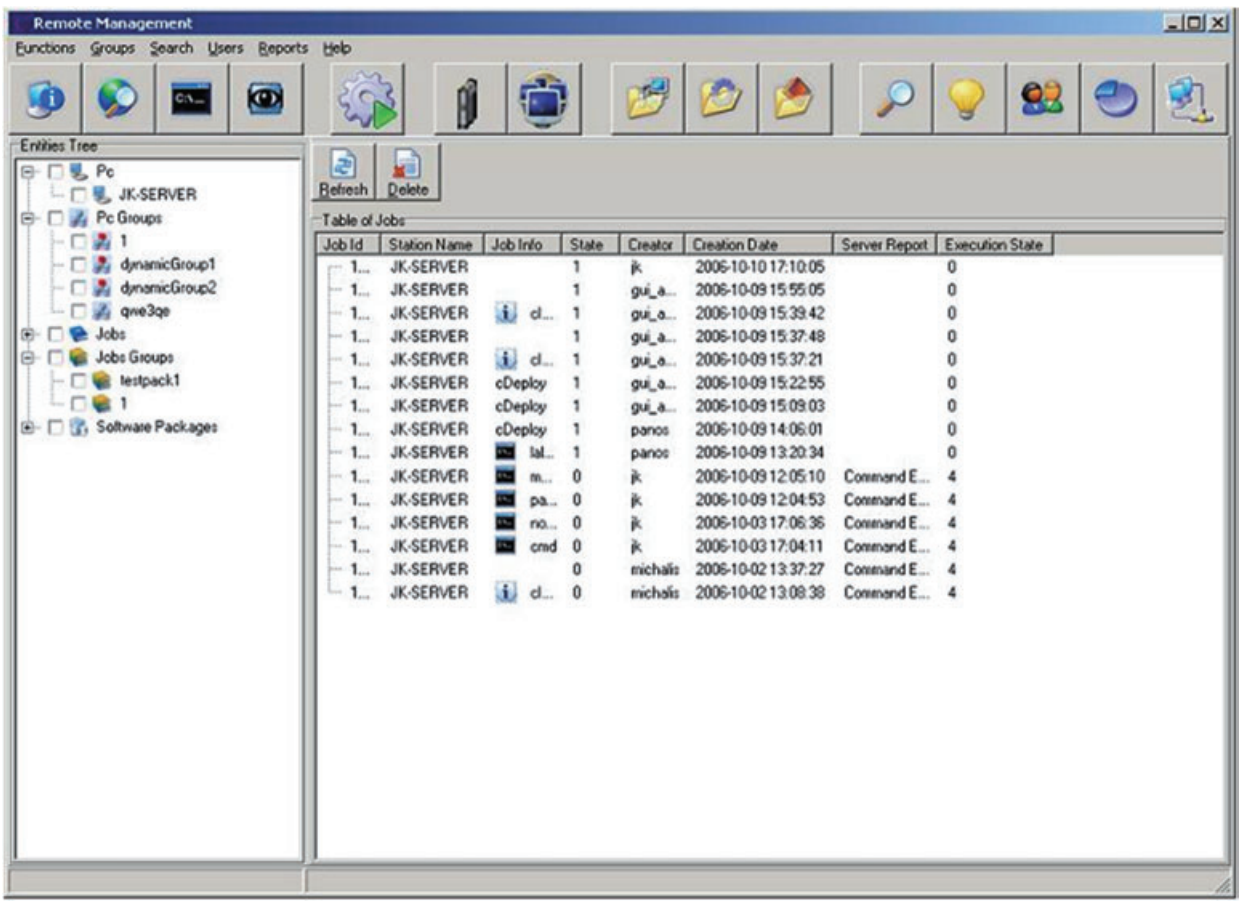

Figure 3. The OpenRSM management console

DOI: 10.1002/nem

management is the functionality that discovers the available stations that run OpenRSM agents and are therefore manageable. Discovery of OpenRSM agents can be directed towards any part of the Internet address space. Only agents configured to communicate with the specific OpenRSM server that originated the discovery packets will respond to the agent discovery. The result of the discovery process is to create active interface elements, representing corresponding managed stations. Along with core jobs (of the inventory, remote control, delivery or remote execution type), they constitute the basic interface elements. Machines and jobs are entities that can be combined, resulting in jobs assigned to specific agent-equipped machines. They are both presented on the management tree for easy supervision. Both jobs and machines can be grouped. Groups of machines and jobs, or groups of jobs and machines, can also be combined in order to create submittable machine–job mappings. Groups are generic: a group of jobs can contain any kind of jobs and no dependencies are implemented. It is the administrator's responsibility to create a rational sequence of a group of jobs for execution.

OpenRSM allows the user to create custom jobs. Following a clean installation, only core jobs exist. The core jobs include predefined inventory, software delivery, remote control and remote command jobs. As stated in the previous paragraph, a job must be visually combined with one or more agents. Thus, jobs can be considered as templates for submitted administration tasks. Each type of job is created using interface components used for that purpose only. For example, a delivery job must 'know' the software that it has to install/uninstall etc. OpenRSM provides the interface for custom job production. The jobs created are stored and made available through the interface so that they can be reused. A user can also use the machine and job groups forms to define machine and job groups, respectively. The management console also provides means of job execution supervision. Users can submit and monitor the execution of jobs in real time through the active job list. The job state is displayed along with information on the agent that executes it, related timestamps, etc. Filters can be applied to the job list, creating job–machine assignments that meet specific characteristics (e.g., owner, date, job type).

One key usability feature of OpenRSM is related to its reporting functionality. This functionality can be further combined with the creation of dynamic groups of machines. The management console reporting can search across the database produced by inventory jobs for machines that match specific user-defined characteristics. The objective is to enable the easy identification of workstations that share common characteristics and group them together in new machine groups, or present their selected attributes on a visual form. An example would be the retrieval of all workstations that have, for example, more physical memory than a specific value, selected by the user. The selection of attributes and the results are performed visually. The resulting workstation information can also be presented as a group of machines, called 'dynamic' because of the way it is created. Dynamic groups behave as normal groups, but they also enjoy the special feature that they are associated with the database statement that created them. The query that created them may be executed at any time, in which case the group is recreated based on updated workstation information.

The reporting functionality is complemented by the data explorer form, created to provide complete database supervision. The user is capable of browsing database entities and combining their contents whenever internal linking is possible. Combinations are presented in the form of reports that can be exported in various formats, such as doc, xls, html, txt and csv.

The management console is also capable of producing cumulative and detailed statistics about system utilization. The generated statistics can record general system usage, job distribution, workstation utilization and user actions. General information is presented in visual charts providing summary information on the job submission rates, job error rates and job distribution with respect to job type and submitting user. Detailed information is presented in individual reports.

2.5 Topologies

The OpenRSM server-side components (Figure 4) can reside on one or more servers. Several of these components can be treated autonomously, so that many topologies and configurations are possible for the OpenRSM server tier. The standard installation uses one machine to host the OpenRSM server;

DOI: 10.1002/nem

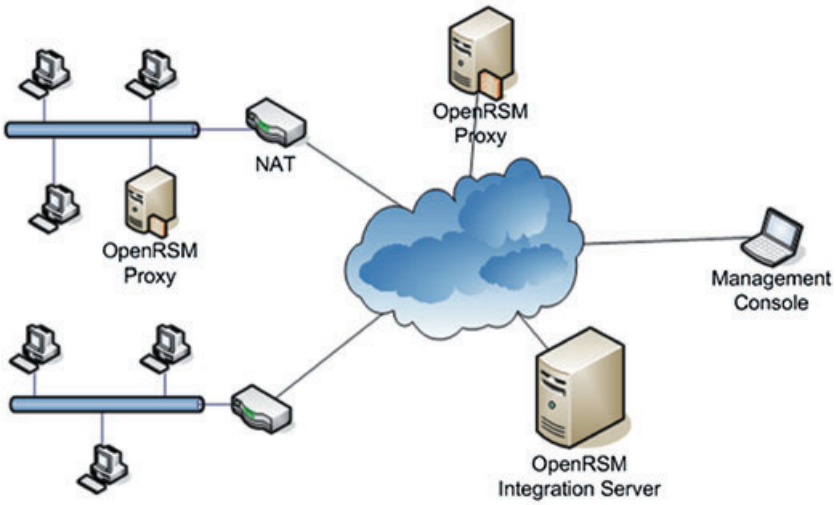

Figure 4. OpenRSM architecture

however, the database, web server and OpenRSM integration server modules could also be installed in different machines according to needs or specifications. The benefits that can be derived from a distributed server topology are mainly related to customization, performance, availability and efficiency. Since overall performance depends on the system load, subsystems that are more frequently used or subsystems that bring greater load to the system can be installed on separate server stations. In that case service availability also increases, since if a single server station fails then only a portion of the system service fails.

NMS usually poses a heavy load on the overall system, and it might be preferable to set it up on an autonomous server. If the server cannot cope with the load, the database server can also be installed on a separate machine. The software repositories of the delivery service can also be separated from the web server. It can be configured to provide service to a subset of the managed terminals so as to balance the overall load of the delivery subsystem. Future work includes the decoupling of the two web applications, asset management and network monitoring, so that the former can be installable on a different server.

A variety of different topologies are also possible. The system can be set up with many OpenRSM integration servers so as to avoid single points of failure. These server modules, each of which orchestrates the integration of subsystems, form the heart of the OpenRSM system and can be more than one per installation, so as to provide enhanced service availability.

Other valuable topologies that may be useful for network traffic planning purposes make use of the OpenRSM proxy server module presented previously. The proxy can be used to aggregate access to the integration server if there are network links that can sustain heavier load. The proxy server is also capable of establishing communication with agents hidden behind NAT networks. If installed on a machine with a visible IP address, it can expose OpenRSM Agents within the NAT network to OpenRSM integration servers that are located outside the NAT network. Implementation depends on the management needs of each application domain.

3. SECURITY CONSIDERATIONS

User authentication and communication channel safety are two important security issues that are raised in most distributed applications, and have also been faced in OpenRSM. Regarding user authentication, the mechanisms employed by the OpenRSM system at the management console end are implemented

DOI: 10.1002/nem

using the secure connection mechanisms of the underlying MySQL database service engine. Furthermore, all local access security issues raised by the execution of the OpenRSM management console and the agent are covered by the local administrative account security specifications. Since no authentication information is transmitted in clear text, prospective attackers may listen to commands and management metadata during communication with the server, but they cannot gain access to the system, even if they manage to take over the station.

The link between the OpenRSM server and the agent uses a protection mechanism based on the a priori knowledge of server location at installation time, the complex server–agent communication protocol, and the agent no-reply connectionless wakening mechanism. The agent state is initially idle and no connections are held open. The server–agent protocol uses combined UDP/TCP communication for the initiation/negotiation mechanism. If a command is to be addressed to an agent, the server informs the agent by sending a UDP packet on a known, customizable port. The agent then wakes up and commences standard TCP communication with the predefined server without exposing any other network traffic. Therefore, network connections are practically server initiated, while server address is statically predefined in the agent. The communication protocol followed consists of a several-step procedure, where each step corresponds to a state in the core FSM of the OpenRSM Agent. After the communication ends or times out, the agent shuts down all connections. When the agent enters this state it is practically immune to remote attacks and invisible to port scans or any network attacks. Even if attackers gain access on the system, they can solely shut down the agent. They cannot send commands to the server, since the communication is server initiated and the communication protocol does not allow the agent to send commands. Even if attackers beat the above mechanism, they will have to use advanced networking techniques to gain control of the agent. This task is far harder than attacking the station itself, and attackers can gain nothing more than the control of the station at best, where their control will again be confined. They can still not damage the OpenRSM system or take control of the server so as to control the stations within the managed domain. Advantages of this design model include the absence of communication problems related to firewalls and routing after the agent has woken up.

4. SERVER TESTING

4.1 Test description and rationale

The OpenRSM system characteristics have been tested through an extended set of experiments emulating realistic usage scenarios. Our goal was to determine the performance levels that can be attained, formalize the corresponding end-user experience, and identify the factors that influence performance. In our experiments and tests, the system was brought close to its limits, in order to verify conceptual and implementation correctness. We focused our experiments on testing the integration server module, and not on the constituent server modules, which are well-known open source web or database servers or applications, utilities or libraries. The following paragraphs present the testing procedure used and the performance results obtained for the integration server module of the OpenRSM system.

The OpenRSM server was installed on a Fujitsu-Siemens RX300S3 machine using four double-core Xeon 5050 3 GHz, 2 × 2 MB processors running Windows 2003 server. Ten custom workstations of varying power were used as the managed agents. The weakest managed system used was a Pentium II, 400 MHz, 256 MB memory workstation. All stations were running Windows 2000 or XP operating system, and all but one resided in the same local network segment. For the purposes of testing, and specifically for the task of sending large numbers of jobs to the server, a particular visual component has been developed. This component, called the testing form, enabled the tester to send multiple copies of one or more selected jobs to the managed workstations. New jobs were treated as new job entities ready for execution, so that the system was not burdened with the overhead of new job creation but only dealt with their successful execution. This choice enabled the emulation of active job submissions without the overhead of the job creation process.

DOI: 10.1002/nem

Since the objective has been to test the behaviour of the server alone, a lightweight job has been chosen for execution. Remember that the integration server is responsible for the management of jobs, and not for their execution. Thus, sending a heavy job instead of a lightweight one adds nothing to the information we can obtain on the performance of the integration server. The testing form was used to send 4, 16, 64, 256, 512, 1024, 2048 and 4096 remote command execution type jobs to the 10 agents, for a total load of the server of up to 40 960 job service requests. During the execution of the test cases, measurements were taken using the Windows native logging system, and a variety of system counters representing the machines and processes states were logged.

4.2 Test results and conclusions

During the testing procedure the server responded in the desired manner to all the system loads we experimented with. It responded to requests and to visual component rendering and always processed data correctly, without halts or hangs. This is also illustrated in Figure 5, where the server throughput is presented. The throughput is a good measure of the overall performance perceived by the end-user. Beyond the quantitative results, Figure 5 also shows that performance degrades smoothly as the load increases up to the rather heavy total load of about 40 000 jobs. The results in Figure 5 also indicate that the software design is sound and appears to be free of major flaws. The diagram starts linearly and then bends smoothly; linear behaviour is maintained up to about 10 000 jobs, at which point the diagram starts to bend smoothly. Figure 5 also shows that the OpenRSM server is capable of dispatching management requests in reasonable amounts of time. In 1 min the server can dispatch about 1250 jobs. This service rate is maintained for a total load up to 10 000 jobs, yielding a throughput of about 21 jobs s^{-1}. When the load reaches 20 000 jobs the throughput falls to 19 jobs s^{-1}, corresponding to a total delay of 17 min. If

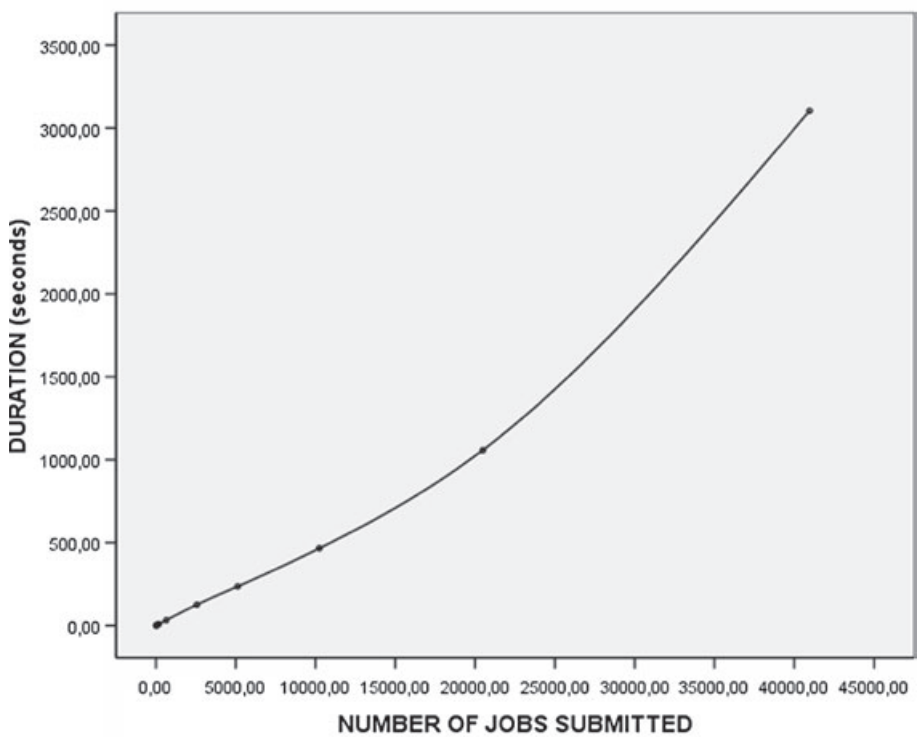

Figure 5. Total test case execution time duration (in seconds) versus the number of served jobs for each test case

DOI: 10.1002/nem

the load is doubled to 40 000 jobs the throughput falls to 13 jobs s^{-1}. The above numbers reflect that the user-perceived performance is maintained practically stable for up to 20 000 submitted jobs, while beyond this point the performance degrades gracefully.

Figures 6 and 7 illustrate the available memory and the CPU utilization at the server machine, as measured by corresponding operating system counters. It is evident that the OpenRSM server has a small memory footprint, since it causes the reservation of only 5 MB of memory under the heaviest test case. Note that the server machine RAM size was 4 GB. It is also evident that memory allocation takes place uniformly. Figure 6 shows that CPU utilization is low and rises linearly with the number of submitted jobs, indicating that the system scales in a predictable and desired way. It must be noted that CPU utilization never reached high levels. This fact can be explained by the cumulative delays (execution, communication protocol and network) introduced by the slowest of the agents. Threads serving fast stations completed job execution rapidly, whereas threads serving slow machines waited for agent response. The slower the agent was, the longer were the waiting times for the respective server threads, and the more were the resources available for fast threads. The reason the server did not reach its limits was that the jobs submitted were far too many for standard workstations to cope with. This result might have been expected since the server was tested against job quantity alone. Another possible series of tests, left for future work, would be to test the way server performance varies with respect to the number of agents or with respect to the speed of the agents.

5. CONCLUSIONS AND FUTURE WORK

The Open Source Remote Systems Management (OpenRSM) system, described in this report, covers all core systems and network management needs in a reliable and scalable manner. OpenRSM is an

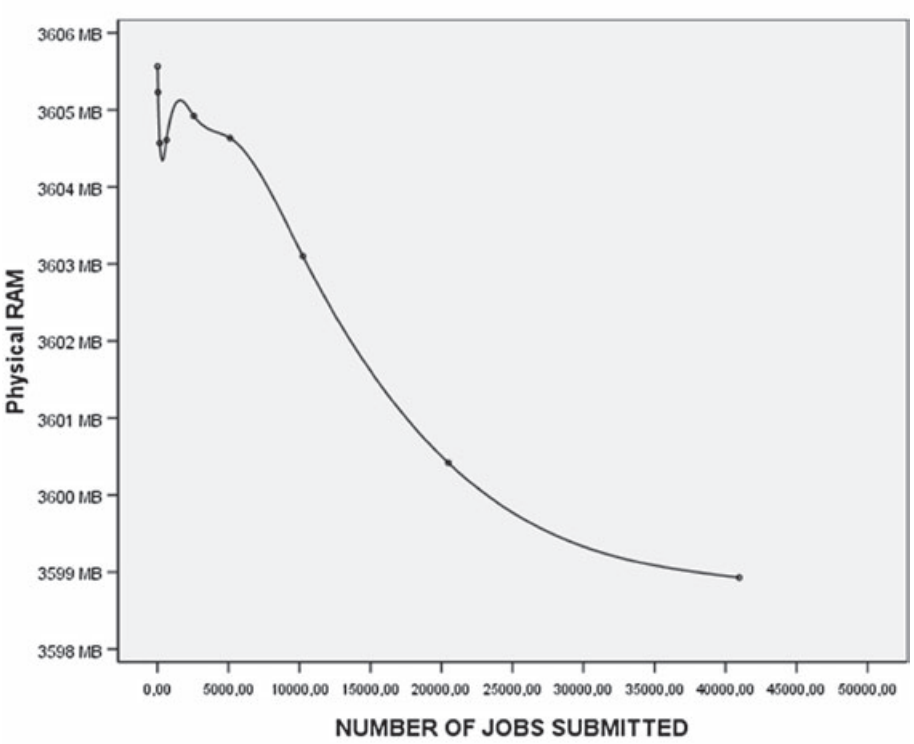

Figure 6. Available memory versus the number of submitted jobs for each test case

DOI: 10.1002/nem

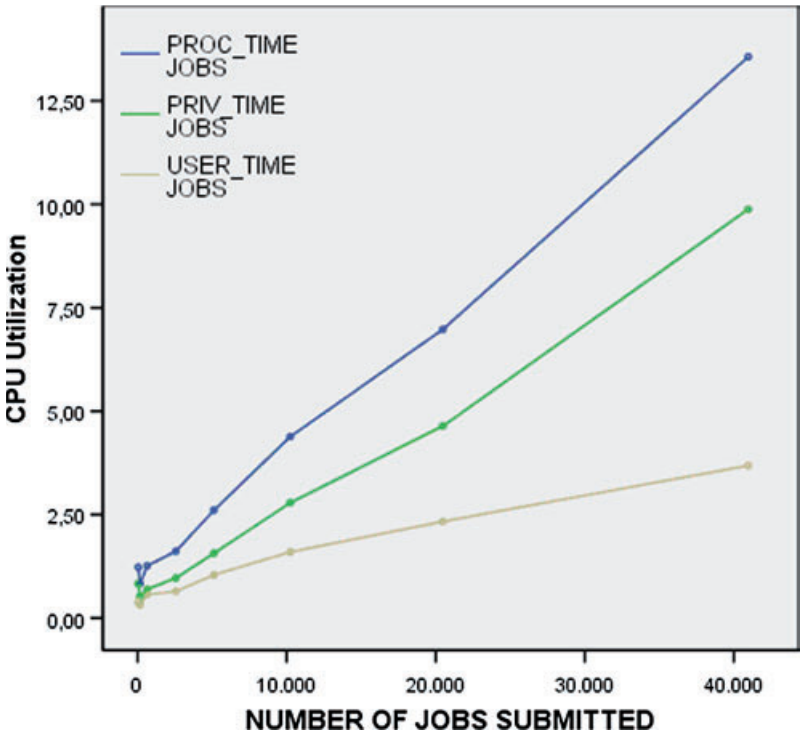

Figure 7. Processor consumption versus the number of jobs submitted. Processor time is categorized into total, privileged and user time

integrated remote management system, implemented by integrating individual specialized open source management tools, and significantly augmenting them to support additional functionality. Use cases in OpenRSM are simple and functional, hiding unnecessary complexity through the utilization of a realistic user-centred design. The OpenRSM system has been deployed in three small-scale pilot installations at the Citizens Convenience Offices [29] under the Ministry of Interior—Public Administration and Decentralization [30], the independent Body of Public Administrator Inspection [31] and the Local Prefecture Offices [32]. The OpenRSM system has been successful in the above pilot installations. OpenRSM has also been gathering attention from the open source community, where medium-size installations numbering several hundreds of managed stations have been reported in the forums of the project. During 2006 the project was downloaded 708 times, corresponding to a volume of 20.7 GB. The corresponding numbers for 2007 are 17.086 downloads and 376.5 GB traffic.

Future work will focus on the extension of the platform to cover a wider range of administrative needs and business goals [18,33]. Although the half proxy module, described in Section 2.2, enables the management of workstations hidden behind NAT, the full proxy module is expected to complete the proxy architecture by providing deterministic management traffic routes to traffic originated by the agent. Along with cascading proxies, it is expected to enable the definition of management traffic paths according to administrative needs. The same architecture can be implemented for the integration service. Future work will also include the development of mechanisms for dynamic failover among integration servers. The separation and installation autonomy of different modules is another future task expected to enable easy deployment, server logic decoupling and application configuration. Another important enhancement expected to impact usability would be the introduction of virtual domain management, a feature not included in the first releases of OpenRSM. Introducing additional tasks that may enhance core administrative use cases, such as simple file transfer, OS unattended installation and managed domain policing [34], also constitute work scheduled for the future. OpenRSM will consider best practices, standards and models such as ITIL and ITSM [19,35–37].

DOI: 10.1002/nem

ACKNOWLEDGEMENTS

We would like to thank the contributing community of the following projects: Winventory/OpenAudit, MySQL direct, Windows-get, Winget, Nino, Xampp, VTree, Jvcl, Jcl libs, Libcurl, 7-Zip and OCS for Linux. We would also like to thank the Greek Information Society program for providing financial support for this work. This work was supported by the Information Society program with 75% funding by the EC through ETPA, and 25% by the Greek State.

REFERENCES

1. Design principles for IT monitoring systems. GroundWork Open Source: San Francisco, CA, 2006.
2. Aziz M.H., Ong ConNie, Jesse Chan Mei Yam, Lee Chang Wei. TCO reduction, In *the proceedings of the 9th Asia-Pacific Conference on Communications*, 2003, vol. 3, 1147–1151.
3. Kakadia D, Thomas T, Vembu S, Ramasamy J. Enterprise management systems part I: architectures and standards, Sun BluePrints TM, April 2002. http://www.sun.com/blueprints [14 June 2008].
4. Hochstein A, Zarnekow R, Brenner W. ITIL as common practice reference model for IT service management: formal assessment and implications for practice. In *Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service*, 2005; 704–710.
5. From OpenView to Open Source. GroundWork Open Source: San Francisco, CA, 2006.
6. List of IT service management vendors. http://en.wikipedia.org/wiki/List_of_IT_Service_Management_ providers [January 2008].
7. Official Computer Associates Infrastructure and Operations Management web site: http://ca.com/us/ products/category.aspx?ID=315 [January 2008].
8. Grieser T, Perry R. Achieving business value and gaining ROI with CA's EITM software for optimizing IT infrastructures. http://ca.com/files/IndustryAnalystReports/eitm_roi_idc_white_paper.pdf [April 2007].
9. Bladelogic operations manager datasheet. http://www.bladelogic.com/products/pdfs/Operations_ Manager_ Datasheet.pdf [January 2008].
10. Westerinen A, Bumpus W. The continuing evolution of distributed systems management. *IEICE Transactions on Information and Systems* 2003; **86**: 2256–2261.
11. Sale M. IT service management and IT governance: review, comparative analysis and their impact on utility computing. HP Laboratories: Palo Alto, CA, 2004.
12. Ding J, Kramer B, Xu S, Chen H, Bai Y. Predictive fault management in the dynamic environment of IP networks. In *Proceedings of the IEEE Workshop on IP Operations and Management*, 2004; 233–239.
13. The OpenRSM project site. http://sourceforge.net/projects/openrsm/ [14 June 2008].
14. Consortium for studying, evaluating and supporting the introduction of Open Source software and Open Data Standards in the Public Administration. Deliverable 2.1 for WP2 'Catalogue of available Open Source tools for the PA'. COSPA: Bozen-Bolzano, Italy, 2005.
15. Dekhil M, Machiraju V, Wurster K, Griss M. Remote management services over the web. HP laboratories, Palo Alto, CA, May 2000.
16. Wren M, Gutierrez J. Agent and web-based technologies in network management. In *Proceedings of the Global Telecommunications Conference (GLOBECOM)*, Vol. 3, 1999; 1877–1881.
17. Lee S, Choi M, Yoo S, Hong J, Cho H, Ahn C, Jung S. Design of a wbem-based management system for ubiquitous computing servers. http://www.dmtf.org/education/academicalliance/ [January 2008].
18. Carey K, Reilly F. Integrating CIM/WBEM with the Java enterprise model. http://www.dmtf.org/education/ academicalliance/ [January 2008].
19. Hochstein A, Zarnekow R, Brenner W. Evaluation of service-oriented IT management in practice. In *Proceedings of the International Conference on Services Systems and Services Management*, Vol. 1, 2005; 80–84.
20. Qin J, Meng D, Gu Z. Research of remote management technology in cluster management system. In *Proceedings of the International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2003; 492–496.
21. The list of available projects in the SourceForge hosting portal. http://sourceforge.net/softwaremap/index.php [December 2007].
22. Bailey I. A simple guide to enterprise architecture. Model Futures TM white paper, 2006. http://www.modelfutures.com/file_download/4/SimpleGuideToEA.pdf [January 2008].

DOI: 10.1002/nem

23. Troni F, Coza R. Client management: a comparison of the leading PC vendors. Technology Overview. Gartner: Stamford, CT, 2004.
24. WBEM. http://www.dmtf.org/standards/wbem/ [14 June 2008].
25. OpenAudit. http://sourceforge.net/projects/openaudit/ [14 June 2008].
26. Windows-get. http://windows-get.sourceforge.net/ [14 June 2008].
27. Curl. http://curl.haxx.se/ [14 June 2008].
28. WMI and SNMP. http://www.microsoft.com/technet/prodtechnol/windows2000serv/maintain/ [14 June 2008].
29. Citizens Convenience Offices. http://www.kep.gov.gr [January 2008].
30. Ministry of Interior—Public Administration and Decentralisation. http://www.ypes.gr [December 2007].
31. Independent Body of Public Administrator Inspection. http://www.seedd.gr/ [December 2007].
32. Achaia Prefecture Offices. http://www.achaia.gr/ [December 2007].
33. Braet O, Ballon P. Business model scenarios for remote management. *Journal of Theoretical and Applied Electronic Commerce Research* 2007; **2**: 62–79.
34. Granville L, Tarouco L. QAME: QoS Aware management environment. In *Proceedings of the International Computer Software and Applications Conference on Invigorating Software Development*, 2001.
35. Bellur U. Automating applications management in the enterprise using DMTF information models. http://www.dmtf.org/education/academicalliance/ [January 2008].
36. Schaaf T. Frameworks for business-driven service level management: a criteria-based comparison of ITIL and NGOSS. In *Proceedings of the IEEE/IFIP International Workshop on Business-Driven IT Management* 2007; 65–74.
37. Shwartz L, Ayachitula N, Buco M, Surendra M, Ward C, Weinberger S. Service provider considerations for IT service management. In *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*, 2007; 757–760.

AUTHORS' BIOGRAPHIES

Yiannis Karalis received a Diploma in Computer Engineering and Informatics in 2005 from the University of Patras, Greece. He is working as an engineer for the Research Academic Computer Technology Institute of Patras and is pursuing the MSc program at the Department of Computer Engineering and Informatics of the University of Patras.

Michalis N. Kalochristianakis is currently occupied as an engineer at the Research Academic Computer Technology Institute of Patras and is also a PhD candidate at the Department of Computer Engineering and Informatics, University of Patras. He has worked in the software industry as a programmer/analyst. He has received the MSc degree in Computer Science from the Computer Science Department, University of Crete, and a Diploma in Electrical Engineering and Computer Technology from the University of Patras.

Panagiotis Kokkinos received a Diploma in Computer Engineering and Informatics in 2003 and an MSc degree in Integrated Software and Hardware Systems in 2006, both from the University of Patras, Greece. He is currently a PhD student at the Department of Computer Engineering and Informatics of the University of Patras.

Emmanouel (Manos) Varvarigos was born in Athens, Greece, in 1965. He received a Diploma in Electrical and Computer Engineering from the National Technical University of Athens in 1988, and MS and PhD degrees in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology in 1990 and 1992, respectively. He has held faculty positions at the University of California, Santa Barbara (1992–1998, as an assistant and later an associate professor) and Delft University of Technology, the Netherlands (1998–2000, as an associate professor). In 2000 he became a professor at the Department of Computer Engineering and Informatics at the University of Patras, Greece, where he heads the Communication Networks Lab. He is also the Director of the Network Technologies Sector (NTS) at the Research Academic Computer Technology Institute (RA-CTI), which through its involvement in pioneering research and development projects has a major role in the development of network technologies and telematic services in Greece. Professor Varvarigos has served on the organizing and programme committees of several international conferences, primarily in the networking area, and in national committees. He has also worked as a researcher at Bell Communications Research, and has consulted with several companies in the USA and in Europe. His research activities are in the areas of protocols for high-speed networks, network architectures, ad hoc networks, network services, parallel and distributed computation and grid computing.

DOI: 10.1002/nem