Controlling wireless access points with open infrastructure management tools

Michael Kalochristianakis
Department of Science
Technological Educational Institution of Crete
Heraklion Crete, Greece
kalohr@cs.teicrete.gr

Konstantinos Koumoutsos, Emmanouel Varvarigos
Department of Computer Engineering and Informatics
University of Patras
Rion, Greece

Abstract* — Managing infrastructure is usually demanding and expensive but at the same time essential for organizations. Open source enterprise management solutions are generally few and immature however, there are tools that aim to deliver lightweight, remote and flexible infrastructure management, easily customizable to cover the needs of small or medium sized organizations. OpenRSM implements a management framework that models the generalized use cases of infrastructure management so that uses can adapt them to meet their needs. As network environments grow to digital ecosystems, the targets of infrastructure management increase in number and diversity. Network elements, smart phones, embedded devices, sensor elements are becoming increasingly common and need to be brought under standard management practices in the same manner as routers or workstations. Despite the maturity of management tools, it is unclear how convenient it would be for users to extend their functionality so as to manage any type of device, besides desktop or laptop stations. This paper describes how OpenRSM can be extended in order to run on embedded wireless access points and how it can be customized in order to effectively manage them.

Index Terms — systems management, network management, open source, assets management, remote desktop control, software management, wireless networks

I. Introduction

While the wide adoption of enterprise computing has triggered the growth of IT infrastructure management (IM) systems in various ways [1], the field is immature as far as open source is concerned. The field is moving towards new directions [2][3] however, during the last decades the evolution of network management (NM) and system management (SM) platforms has led to high-end integrated enterprise management systems (EMS) [4] that cover general purpose IM. Designed to appeal to IT staff mostly [5], such systems currently offer elaborate characteristics such as flexible architectures, wide ranges of configurable management commands, data mining, incident analysis and correlation, among others [6]. Meanwhile, the field of open source IM tools offers valuable NM platforms, specialized tools and services, such as the ones mentioned in section II, but no integrated IM systems. Yet, even if homogeneous, scaled enterprise environments can rely on existing integrated solutions [7], smaller, modern IT ecosystems that is, environments that tend to be dynamic, consisting of a wide variety of networked devices, operating under stringent economics, in constant search for flexible and customizable solutions, could be the target of open source IM. Moreover, there is a need for efficient IM for wireless devices especially in suburban locations. In order to penetrate such environments and to apply open, uniform IM practices it can be suggested that emphasis may need to focus on SM besides NM and on extensible, customizable management architectures, suitable for any type of platform or device, such as the case of wireless embedded devices [8].

The open remote systems management tool (OpenRSM), is a novel approach for remote network and systems management as it is one of the few tools that offer remote SM and NM functionality by integrating and extending valuable open source tools. OpenRSM relies on a management framework [9] that takes advantage of object oriented principles and model driven design and integrates the functionality of its building blocks. It aims to cover the needs of general infrastructure management practices, that is, supervision, detailed assets reporting, remote desktop control, network management and customized intervention. The framework is designed to be generic, so that the system is not dependent on any operating system or underlying platform and so that it produces added value in terms of the overall functionality offered to the end-user. Although lightweight, fast and simple in use, OpenRSM appeals to experienced users or system administrators who can exploit its philosophy and create customized management procedures. OpenRSM is hosted in the Sourceforge [10] open source project repository site and is distributed under the general public license.

The latest challenge for OpenRSM has been to extend the IM use cases it supports for active network devices, especially for wireless access points (AP) that run LINUX on embedded processors. With the increasing adoption of wireless technologies at the access tier of its architecture, the Greek School Network (GSN) [11] is motivated [12] to use OpenRSM, which has already been installed in pilot deployments within its domain. Extending work on OpenRSM for APs IM was based on migrating the agent tier of the system in order to run for embedded processors. It was also necessary to extend the management framework of OpenRSM in order to

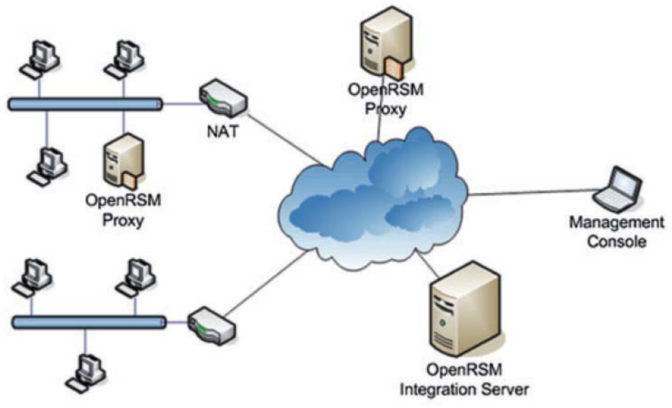


Fig. 1: The component architecture of OpenRSM

take advantage of its general purpose philosophy that is, to produce customizable management tasks that exploit native features and system facilities of APs. Management tasks can then be organized to implement scalable IM use cases that foster business purposes for the management of the wireless access network for GSN, such as massive firmware updates, reports about the state and the assets of wireless APs, single-click configuration of operation modes, network characteristics and more. The extension was tested in conditions that simulate GSN school labs connected via wireless technologies.

The rest of the paper is organized as follows: section II presents the related work in the field of open source remote management systems with emphasis on systems that can be extended to manage embedded AP systems. Section III briefly overviews the main features of the OpenRSM system. Section IV presents how OpenRSM was extended in order to support wireless devices that run on embedded processors, the challenges and the solutions, and how OpenRSM can be customized in order to support the management of wireless APs. Finally, section V discusses the conclusions and future work.

II. Related work

There is limited documentation about work in process on open source IM systems that integrate [13] functionality for wireless APs. Besides supporting the SNMP protocol, wireless AP vendors implement a multitude of SM software solutions that vary with respect to the specific AP type or model. There are vendors that release NMSs for the uniform monitoring of their platforms but there has been no significant effort for integration with any well known IM. On the other hand, IM platforms and EMSs may not manage wireless devices that cannot support their agents or do not conform with the required SM management standards. The former diversity of IM practices may be an additional reason why open source may be suitable for AP management. The core functionality offered by integrated IM platforms can be categorized in network management (NM), inventory and assets management (AM), software distribution (SD), remote desktop control (RDC), software delivery (SD). Although there are very few if any open source integrated IM platforms there are many open source tools that offer pieces of EMS functionality. Scrutiny of field yields the main open source systems that relate to IM that is, OpenPegasus, Nagios, OpenNMS, GroundWork, ZenoSS Core, Hyperic HQ for NM, QUATOR, Arusha, CfEngine, PIKT, BCFG2, TALOS, APT-GET, OPSI, WinGET, CURL, PatchMonkey and MINST for SD, OpenAudit, OCS and H-Inventory for AM and TightVNC, TeamViewer for RDC. There are platforms available for enterprise architecture management [14] [15] and it is also notable that some NM systems have lately developed basic SM features; BigSister and ZenOSS platforms offer remote task execution, the separating characteristic between NM and IM. Even so, the vast majority of the systems described above were built to implement specific use cases for IM and not to rapid application and flexibility so that they can support extreme cases such as the management of AP systems running on embedded processors.

III. OpenRSM overview

OpenRSM integrates several systems, such as, OpenAudit, TightVNC, Win-GET, CURL, OpenNMS and the NINO NMS. The successful integration of the former systems in OpenRSM resulted in an open source tool that is stable and mature enough for integrated remote systems management. The advantage of OpenRSM, against other platforms and open source tools lies in the potential for simplicity, customization and configurability that can potentially protect administrators and IT managers from difficult situations such as the ones described in [7]. OpenRSM is designed so that it can easily be extended to include different kinds of devices under its management and also customized in order to adapt to the differentiated requirements of management, such as the ones for wireless APs that run on embedded processors. OpenRSM has passed stress and scalability tests and has been deployed in tree pilot installations [9]. All the components of the system that is, the server, the agent, and the manager support automatic installations that are created using known open

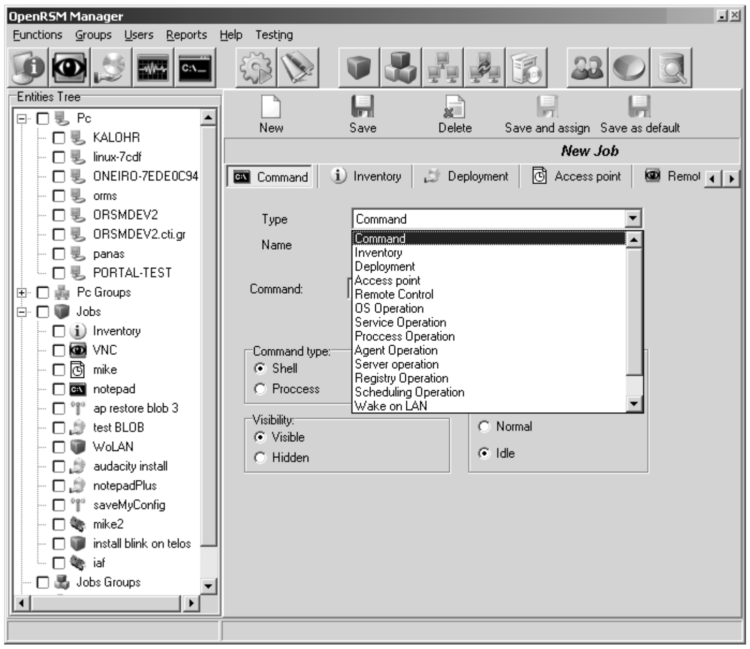


Fig. 2: The management console supports templates at the task creation form

installer packagers. The component architecture of the system is illustrated in fig. 1. The agent size is small, about 1.5 megabytes, and can be distributed easily in cases of installations of scale or ones in remote areas connected via slow lines, such as the case for remote GSN school labs [16]. After the installation, which exposes a graphical user interface, no configuration is necessary and the administrators can start using the tool. In order to facilitate scalable installation, besides operation, the OpenRSM agent is installable in unattended mode. The remote management is conducted mainly via the management console, which is a graphical interface designed to be simple, lightweight and easy to learn by administrators or technical staff. OpenRSM delivers the fundamental IM services, NM, AM, SD, RDC, complemented by useful characteristics such static or dynamic agent or task groups, reports and machine state discovery. Figure 2 presents the management console. The left panel presents the main management entities that is, managed hosts, task instances, machine/task groups and software packages on the tree structure. The main panel in figure 1, on the right, presents the task creation form that can be used to create any type of commands which are supported by the system.

Customizing OpenRSM is straightforward; administrators can organize and elaborate management in terms of tasks or machines which can be organized in static groups. Machines can also be organized in dynamic groups whose members can be defined on the basis of one or more common AM characteristics they may share. Such correlations are created by the dynamic construction and the execution of queries on the AM database. Groups of tasks are used in order to execute commands in batch, sequential mode, under the main agent thread or in one thread per task. From the perspective of design, tasks are classes that belong to the OpenRSM hierarchy of extensible objects. The user is provided with template tasks, one for each subsystem that can be customized and then instantiated to suit their needs. There are templates for AM, RDC, remote procedure call, SD, wake on LAN, server task, AP command.

The task inheritance and instantiation mechanism that leads to tasks creation is the basis for the separation of system and user logic. Top-level, abstract task classes correspond to fundamental management use cases integrated in OpenRSM. The object oriented approach of the management framework provides for flexibility and thus tasks can be subject to all the customizations supported by the integrated subsystems or tools. For instance, the RDC command can be configured to use proxying through the OpenRSM server instead of direct connections from the management console to the agent. This feature exploits the corresponding feature of TightVNC remote control tool and can be proven enabling in network environments where network segmentation cuts off direct connections. The user can use the task template to create tasks, one connecting to stations within their network segment, and one for connection to remote ones. Remote control tasks support the underlying VNC features such as asking the user for permission to access their desktop, defining the access to be view only and defining time-outs. SD tasks can also be customized to the limits of the underlying tools and components since they can be used to install/uninstall software using the methods supported by the installer used by the delivered software. Driven by the lack of dominant standardization, there is a plethora of installers available, each of them using custom techniques to handle the installation of software. The SD tasks do not depend on any technology and allow users to supply the desired method of installation such as silent, unattended, graphical, etc and all the relevant information that is, serial keys or installer switches, when necessary. The most customizable task is the remote command task since it is generic and platform independent by design and it also supports all the execution parameters such as visibility, priorities and process type. Users can utilize this task type as a wrapper for any command they wish to send to remote work stations. Other characteristics of OpenRSM in terms of customization include the use of proxying in order to build connection or traffic paths that can be suitable for purposes that may relate to traffic optimization or security. Finally, the architecture of the OpenRSM server is designed to flexible so that it can support various configurations and installations in order to adapt to the needs of users, ranging from centralized to distributed configurations.

The features discussed above can be enablers for adaptive remote management capable to cover the routine of administrators when managing a wide range of devices and systems. However, OpenRSM needs to run on platforms such as school network wireless AP that run LINUX on embedded MIPS processors. In order to meet such needs the OpenRSM agent module needed to be appropriately customized in order to run on such systems without limiting the functionality the system already offers.

IV. Extending and customizing the agent for embedded devices (and open source systems)

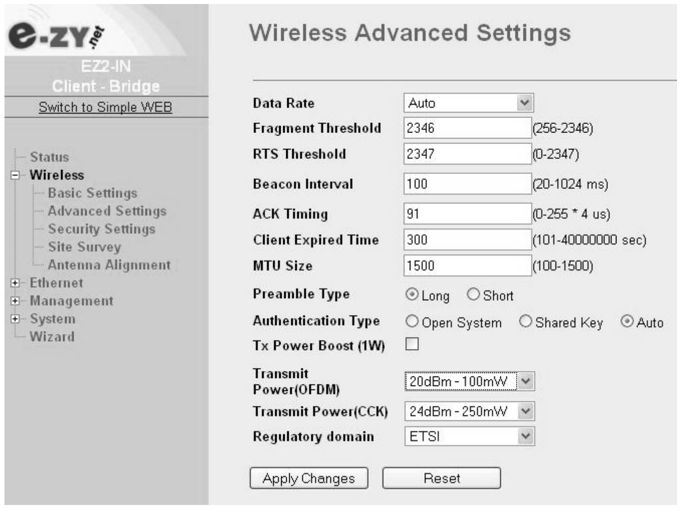
The OpenRSM service was designed and built to be cross-platform and the agent was initially developed to run on WINDOWS operating systems. Its distribution for open source operating systems is based on the cross-compilation of the code written for the WINDOWS platform using extensions for the Delphi integrated development environment (IDE) such as the one described in [7]. This agent release, though thoroughly tested against all major LINUX platforms, cannot be considered reliable for embedded devices due to the diversity of characteristics of the latter in terms of memory and CPU resources. That is, there are resources that need to be managed within the source code instead of relying on general compiler configurations. The alternative, to use the open source release of the IDE, Kylix [17], that could run on open source operating systems and produce native object code, would be a risk, since it is an inactive project. Other similar environments were examined such as the Lazarous open source PASCAL project but in order to directly address the problem of effectively producing agents for any potential platform it was considered preferable to write a command line version of the OpenRSM agent in C that could be compiled and run in the widest range of platforms. AnsiC was preferable since it is supported by almost every compiler for almost any CPU and thus the new agent can potentially be used as basis for extending OpenRSM functionality for many embedded devices.

TABLE I: THE PACKET HEADER FIELDS

Name	Description
VER	Protocol Version
TYP	Packet Type
DUMMY	Dummy details
UID	Sender identification
JID	Working tasks machine id
CMD	Command Type
PRM	Command Parameter 1
PR2	Command Parameter 2
PR3	Command Parameter 3
DKY	Data Key
DAD	Discovery Address
NAD	Next Address
LEN	Length of data bytes coming next
CRC	Head or Data CRC

A. System migration

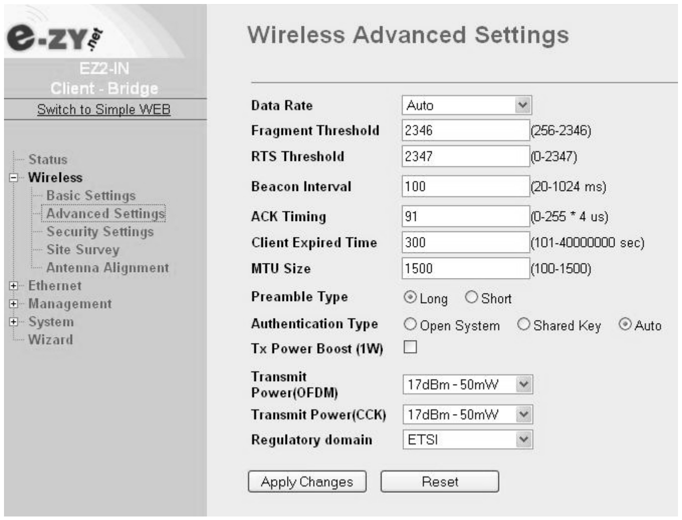
The OpenRSM agent is released in four forms that can be used to cover different uses. The graphical version presents an interface which is nestable in the desktop toolbar and which can be used to display the state of the system and also messages about administrative actions. Alternatively, the agent can be distributed in the form of a windows service, as a background process or a command line program. Migrating the command line OpenRSM agent code for C involved the resolution of a number of problems mostly relating to lower levels of compilation and logic, such as byte alignment and byte ordering. Miss-alignment can be caused by different padding optimization policies applied by compilers. Policies can be defined at compile time, but predefined configurations differ with respect to compilers version, type and underlying platform. The alignment of data may consequently vary, resulting in inconsistent representations of data structures when communicating ends are not compiled with the correct alignment. In the case of OpenRSM, the problem was exposed when the structure that represents the packet header exchanged by the server and the agent needed to be read by the AnsiC agent. The packet structure and explanations of the fields of the packet header is presented in table I. Delphi supports alignment of structures at multiples of the processor alignment and the default record field alignment value is 8 bytes while the corresponding value for the MIPS processor, that is typically used in GSN wireless AP, is 4 bytes. Furthermore, there can be type sizes incompatibilities between AnsiC and Delphi, which result to differences in the size of structures. Byte alignment was resolved by employing techniques that explicitly control the alignment of the data structure with padding and appropriate compiler configuration and more specifically the #pragma pack directive for explicit control of the alignment. When the default value that is, 4 bytes is used, bytes 2, 3, 4 and 6, 7, 8 are padded in order to preserve the alignment. However, the server uses the alignment shown in table II and thus the agent cannot read it correctly. For example, when it tries to read TYP it gets the first byte of UID and so forth. Except for the first byte of the structure, all the others are misaligned. The directive is used to explicitly instruct the agent to preserve 8 byte ordering. The directive is used only for the data that come from the server while analogous syntax, based on the pack directive is used to recover the native alignment. Configuring the byte ordering of executable with respect to the type of processor was also taken into account. The APs of interest used MIPS processors with a



Wireless Advanced Settings

Data Rate	Auto	
Fragment Threshold	2346	(256-2346)
RTS Threshold	2347	(0-2347)
Beacon Interval	100	(20-1024 ms)
ACK Timing	91	(0-255 * 4 us)
Client Expired Time	300	(101-40000000 sec)
MTU Size	1500	(100-1500)
Preamble Type	[✓ Long] Short	
Authentication Type	Open System Shared Key [✓ Auto]	
Tx Power Boost (1W)	☐	
Transmit Power(OFDM)	20dBm - 100mW	
Transmit Power(CCK)	24dBm - 250mW	
Regulatory domain	ETSI	

(a) The assets management interface before the remote management task is sent to the AP



Wireless Advanced Settings

Data Rate	Auto	
Fragment Threshold	2346	(256-2346)
RTS Threshold	2347	(0-2347)
Beacon Interval	100	(20-1024 ms)
ACK Timing	91	(0-255 * 4 us)
Client Expired Time	300	(101-40000000 sec)
MTU Size	1500	(100-1500)
Preamble Type	[✓ Long] Short	
Authentication Type	Open System Shared Key [✓ Auto]	
Tx Power Boost (1W)	☐	
Transmit Power(OFDM)	17dBm - 50mW	
Transmit Power(CCK)	17dBm - 50mW	
Regulatory domain	ETSI	

(b) The assets management interface afterthe remote management task is sent to the AP

Fig. 2: Adjusting the transmit power setting from 20dBm-100mW to 17dBm-50mW using a remote FLASH SET command

big endian configuration, while x86 CPUs are typically little endian. The endianess of the binary code was configured through appropriate compiler configuration.

The codebase of OpenRSM was branched to support individual versions of the OpenRSM agent for both LINUX and embedded systems. The main distribution of the agent supports the full functionality offered by OpenRSM while the one for embedded devices does not support RDC tasks, since the firmware of devices do not run window manager services. Also, the AM system needed to be configured and tuned to adapt to the type and volume of information supported by APs.

OpenRSM was also extended to support specialized tasks for the configuration and the management of firmware, operations that were supported by the APs used. Configuration as well as firmware management includes retrieving or loading AP configuration files or binary firmware to or form the OpenRSM server via appropriate tasks that can save or restore the configuration text or the firmware binary image for the wireless device. The management graphical interface was extended in order to cover the above functionality in terms of the components that represent the tasks, the logic and the representation on the entity tree. This included representation of configuration files and firmware images on the entity tree and the construction of task creation forms as well as the underlying mechanisms for uploading the files. The server was extended to implement all the relevant encoding for the new tasks, the underlying transfer and store functionality and the database tier was extended to provide information storing. The agent was extended to support the new features of the management framework.

B. Using the system

In order to customize OpenRSM for the needs of wireless AP management, custom tasks needed to be created and sent to remote APs. The creation of administrative tasks included the encapsulation of native commands into OpenRSM RPC tasks. After the creation of an RPC task, a graphical component represents the instantiation of the task object, which is the standard operation of the OpenRSM system. When associated with a machine entity in the console interface, or with a group of machines, the task is committed to the server where it is decoded and it is then sent to the OpenRSM agent(s). Native commands for APs, capable to control all the aspects of the system were FLASH SET commands. Their purpose was to set parameters such as the ones listed in table III for the AP. FLASH SET commands were encapsulated in custom OpenRSM remote command tasks and were conveyed and executed on the agent(s) as simple console commands. Examples of the effect of the management tasks are presented in figure 2. Figure 2 presents that the transmit power of the wireless AP is decreased from 20dBm OFDM / 24dBm CCK to 17 dBm OFDM/CCK. The change is the effect of a remote execution task creation or modification so that it encapsulates the FLASH SET TX_POWER command and send it to the wireless AP using the graphical management interface. Checking the new setting can be done via the refreshing of the AM information for the remote AP by running the inventory management task on it as in figure 2. An analogous FLASH SET REPEATER command can be sent in order to enable the node to operate as a repeater. The target AP can be configured as if the administrator was connected through a local console. They can thus configure any aspect of the system such as operation mode, band, SSID, channel, auto channel select, data rate and max retries, ACK timing, IAPP enable/disable, authentication type, transmit power, IP configuration, hostname, services.

The extended OpenRSM system was tested in a wireless local area infrastructure that was used to bridge local area wire networks and to provide wireless access to roaming stations,

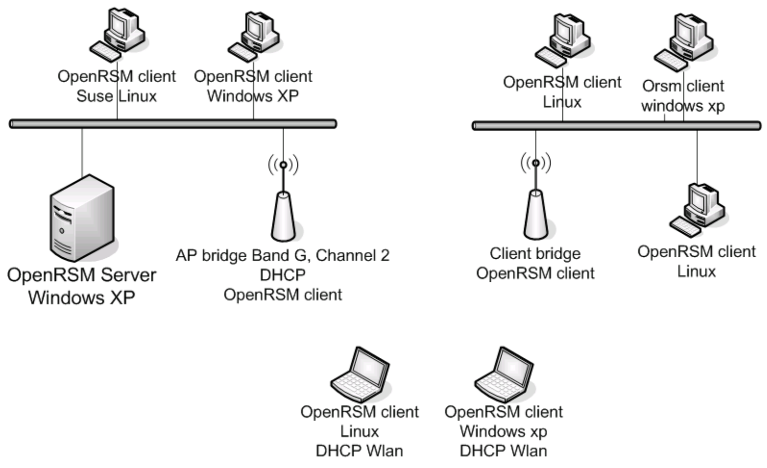


Fig. 3: The topology of the wireless AP management network

just as in GSN school labs. Figure 3 presents the typical network configuration where a wireless bridge connects the OpenRSM server with other network segments. The network connected all types of workstations with the OpenRSM server and consequently all types of OpenRSM agents were utilized. The agent was installed on workstations based on WINDOWS or LINUX operating systems and on wireless devices. The administration tasks that were performed included state checking tasks, AP configuration or maintenance monitoring, firmware upgrade and saving and reloading configuration. The most common tasks involved configuring the wireless operation mode, altering communication and connection properties, enforcing security, changing the internet protocol settings, managing services, backing up/restoring the configuration. Passive management, that is, monitoring, included the standard practices for the retrieval of AM information and SNMP management. Active management was performed mainly via custom tasks that relied on the systems' extended functionality.

V. CONCLUSIONS AND FUTURE WORK

This current paper described how the agent of the OpenRSM system was migrated to C and then adapted to run on wireless AP that rely on lightweight versions of LINUX for embedded devices, more specifically for the MIPS architecture of processors. With the latest release of the agent, the functionality of the system can be extended to include administrative tasks for any operating system and any device, for which it can be compiled. The paper outlined the design-time and compile time considerations for running the agent in embedded platforms and presented how the task description framework of OpenRSM was exploited in order to construct custom tasks for use cases that can fall within the routine of GSN administrators. The extension of the OpenRSM system was tested on a network that used bridging to connect school lab networks and roaming stations in an open medium environment. The experience with OpenRSM for wireless AP management brought us to the conclusion that IM was simplified and tasks dispatch rate was accelerated, since tasks could be sent to stations simultaneously and could also be reused.

Future work on AP IM with OpenRSM will focus on producing releases for families of AP infrastructure that support similar functionality for example common configuration and firmware transfer interfaces. OpenRSM is also being extended in the direction of wireless sensors management and power management and modern management technologies.

REFERENCES

[1] M. van Sinderen, "Challenges and solutions in enterprise computing", in Enterprise Information Systems, special issue, challenges and solutions in enterprise computing, 1751-7583, vol. 2, issue 4, 2008, pp. 341 – 346

[2] M. Carlson, "Systems and virtualization management: standards and the cloud (a report on SVM 2011)", Journal of Network and Systems Management, 2012, vol. 20, issue 3, pp 453-461

[3] L. Granville, D. Medhi, Y. Kiriha, J. Hong, N. Fujii, "Towards management of future networks and services: a report on IEEE/IFIP NOMS 2010", Journal of Network and Systems Management, 2010, vol. 18, pp. 348-353.

[4] D. Oberst , "Enterprise systems management" in Educase, 2001, pp 58-59

[5] A. Gupta, "Network Management: current trends and future perspectives", Journal of network and systems management, vol. 14, number 4, pp. 483-491, 2006

[6] P. Stephenson, "The application of formal methods to root cause analysis of digital incidents", International Journal of digital evidence, vol. 3, no. 1, 2004

[7] M. Kalochristianakis, K. Vassilakis, M. Paraskeyas, E. Varvarigos, "Considerations for successful enterprise information systems deployment: the case of the Greek School Network", International Conference on Service Systems and Service Management, 2012

[8] Bernstein, "Network management isn't dying, it's just fading away", in Journal Of Network and Systems Management, vol 15, number 4, pp. 419-424, 2007

[9] I. Karalis, M. Kalochristianakis, P. Kokkinos, E. Varvarigos, "OpenRSM: a lightweight integrated open source remote management solution", Intl Journal of Network Management, Vol 19, issue 3, pages 237-252, May 2009

[10] The OpenRSM project, http://openrsm.sourceforge.net/, accessed on July 2012

[11] The Greek School Network portal, http://www.sch.gr/, accessed on July 2012

[12] M. Kalochristianakis, M. Paraskeyas, E. Varvarigos, N. Xypolitos, “The Greek School Network: A Paradigm of Successful Educational Services Based on the Dynamics of Open-Source Technology”, IEEE Transactions on Education, vol. 5, issue 4, pp. 321-330, Nov 2007

[13] M. Feridan, M. Moser, A. Tanner, Building an abstraction layer for management systems integration, in: Proceedings of the First IEEE/IFIP International Workshop on End-to-End Virtualization and Grid Management (EVGM’2007), 2007, pp. 57–60.

[14] H. Jonkers, M. Lankhorst, L. Doest, F Arbab, H. Bosma and R. Wieringa, “Enterprise architecture: Management tool and blueprint for the organization”, Journal of Information Systems Frontiers, vol. 8, number 2, 2006, pp 63-66

[15] S. Buckl, F. Matthes, C. Schweda, “A Viable System Perspective on Enterprise Architecture Management”, in 2009 IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, USA, 2009.

[16] The Greek School Network remote control service support web pages, http://www.sch.gr/rc, accessed on July 2012

[17] Kylix developer support, http://info.borland.com/devsupport/kylix/, accessed on July 2012