Computing and Informatics, Vol. 30, 2011, 749–760

VIRTUAL GRID – NEW PARADIGM OF SYSTEM RESOURCES DYNAMIC ORGANIZATION

Joanna Kosińska, Jacek Kosiński, Krzysztof Zieliński

Department of Computer Science
AGH University of Science and Technology
al. Mickiewicza 30
30-059, Kraków, Poland
e-mail: {kosinska, jgk, kz}@agh.edu.pl

Communicated by Jacek Kitowski

Abstract. This paper describes the model and software tools for virtualization of the Grid execution environment. The basic assumption of this study is that Virtual Grid (VG) is constructed dynamically according to application requirements with virtualized computational and communication resources. A VG instance is dynamically created for each application and its initial configuration can be further modified during runtime to satisfy the requested level of resource usage and to maintain the quality of service indexes at agreed-upon values. A VG model that takes into account this approach has been proposed and analyzed in the context of Grid-related technology.

Keywords: Virtualization, resource management, Virtual Grid

1 INTRODUCTION

The concept of distributed computing based on the Grid model has been widely adopted by research environments and commercial organizations. A crucial task is the creation of systems for Grid resource management. These systems can increase the effectiveness of usage and operation of Grid infrastructures for wide groups of applications.

The classic approach to resource management, realized by middleware, carries some limitations both with regard to its operation and environment capabilities [4]. The application is not able to search, choose and obtain resources which would ensure optimal operation. The list of possible configurations would be too long for effective selection. In addition, the Grid resources are distributed, shared among different organizations, and possess heterogeneous features and configurations. Moreover, their availability is not guaranteed at any given time. All these factors result in difficulties, or even in the inability to obtain a valid set of resource configurations that would guarantee optimal performance of a distributed application.

By analyzing the architecture of management systems currently used in Grid environments [16, 3, 22] and proposed as solutions for further implementation of such environments [17, 18, 5], it is possible to define the main aims of a resource management system. Primarily, these include efficient management of generic computing resources as CPU power, memory allocation or disk access. In distributed systems, a crucial role is also attributed to communication between distributed components and network resource allocation.

This paper describes the model and software tools for virtualization of the Grid execution environment. The basic assumption of this study is that Virtual Grid (VG) is constructed dynamically according to application requirements with virtualized computational and communication resources. This highly demanding VG environment requires selection of a suitable virtualization technology for CPU nodes and communication networks. An integrated approach to virtualization of CPU power and network usage is the main innovation of this study. By complementing this solution with virtual organizations [14] there rise opportunity for more flexible resource allocation. The primary form of usage of proposed environment is the optimization of the way of using shared resources by computing applications (HPC).

Section 3 describes the requirements that Resource Management System (RMS) should meet in distributed systems. In the subsequent section, a system that meets the stated requirements is described. This section also presents the concepts of using virtualization. Section 5 presents a multilayer implementation model of Virtual Grid Resources Management System (VGRMS) which is designed to support VG lifecycle management. By using VGRMS, the allocation of computational and communication resources can be autonomicaly influenced without the need for application reconfiguration and behavioral modification. The paper wraps up with conclusions and ideas for further improvement.

2 RELATED WORK

A great deal of effort has gone into developing virtualization techniques for computing environments. Most of them focus on manipulating virtual resources in existing computing infrastructures. Paper [2] describes the opportunities offered by virtual computing in the area of Grid computations. It shows benchmark results and includes an overview of how virtual systems could be integrated with Grid computing. Another work ([3]) describes the requirements and solution for modelling dynamic virtual environments as entities in a distributed environment (DVE), with Grid service interfaces defined to negotiate creation, monitor properties, and manage the lifetime of DVE. The article also visualizes some use cases where the usage of virtualization can provide an advantage and explains why there is a drive towards achieving virtualization in computing environments. Moreover, it also provides a short introduction to Xen and presents benchmark results which compare the Xen system with an SMP computer.

In [5], the authors present the concept of a virtual workspace that allows a Grid client to define an environment according to its requirements (such as resource requirements or software configurations), manage it, and then deploy the environment in the Grid. This paper describes how virtual workspaces fit into the Grid architecture, present a prototype of such an architecture based on the Globus Toolkit and experiment with virtualization based on VMWare. The rest of the paper presents the experience integrating VMs with the Grid infrastructure. Paper [7] presents virtualization middleware addressing complete networks instead of single virtual machines. Having proposed a solution, the author presents benchmark results for selected Grid applications running in this virtual network.

[13] presents a Grid service – VMPlant – that provides the possibility to create VMs that, once configured to meet application requirements, can subsequently be cloned and instantiated in order to provide homogeneous execution environments across distributed Grid resources.

As can be noticed, most of these efforts address the specific issue of virtualizing computing resources and concentrate on problems resulting from the integration with existing Grid environments at the middleware level. There is an observable lack of complex solutions addressing both network and resource virtualization. In the works presented above, the problem of autonomic regulation of resource access and the manner of ensuring service quality is largely neglected.

3 THE PURPOSE OF VIRTUAL GRID RESOURCE MANAGEMENT

The main requirements of Grid systems can be formalized through the definition of a primary set of services which the system should provide. Following is a list of services composing the resource management system for a modern Grid architecture:

- resource discovery,
- access to information about resources,
- monitoring the task status and the environment state,
- resource allocation,
- SLA reservations/limits/contracts handling,
- task execution management,
- accounting and reporting.

The usage of virtualization techniques (with VG concept in mind) plays a crucial role. The pool of physical nodes can be replaced with a collection of VMS that fulfil the role of application execution containers. It is possible to execute many components of a distributed application in an isolated manner even within a single physical node. The virtualization technique also enables sharing the available resource pool among different, simultaneously running applications. Moreover, computer network can be dynamically constructed, based on application requirements.

A VG model that takes this approach into account has been proposed and analyzed in the context of Grid-related technology. The novelty of the proposed approach can be expressed in several important aspects:

- The VG execution environment integrates virtualized computational and networking resources,
- VG can be deployed on demand, according to application requirements, on the available physical resources in a shared infrastructure,
- Once deployed, VG can be modified during runtime so the running application can obtain or release access to physical resources during execution,
- VG runtime management can be performed manually or by a policy engine, executing policy rules defined by the system administrator,
- VG deployment and runtime management should be neutral from the perspective of Grid middleware usage.

In summary, the main task of the VGRMS components is to control a number of VGs operating upon a shared physical infrastructure [11]. Control is executed by a rule engine, processing rules that govern the usage and sharing of resources specified by the user of the infrastructure, based on application requirements. The other goal is to achieve a tradeoff between a guaranteed level of QoS for applications executing within VGs and maximizing the utilization of shared resources.

4 RESOURCE MANAGEMENT IN THE VGRMS SYSTEM

Hierarchic approaches are most often [12] taken into consideration when building resource management systems in Grid environments. This results mainly from the necessity to support management in spite of strong dispersion and heterogeneity of resources. The hierarchic concept assumes the division of RMS operation areas into domains [8]. This division can result from the geographic arrangement of managed resources, their type, membership in administrative domains, specificity of network communications, etc. A local manager is created to serve each specific domain. However, these managers are not autonomic, as their operation is controlled by a global resource manager (handling the global resource domain).

The hierarchic management approach (Figure 1) avoids many problems related to scalability and decreases the global complexity of management systems. In the proposed system the specific levels of the management hierarchy correspond to the process of mapping an application onto virtualized resources, which is performed in two stages. The first binding is realized between physical and virtual resources, while the second is the allocation of virtual resources to an application. The virtue of this two-stage mapping is the simplification of resource allocation, since operating on virtual resources rather than on physical ones allows us to express the application requirement specification on a higher level of abstraction. This solution can therefore separate the primary (business) application functionality from resource management, hence simplifying the expression, maintenance and modification of resource management mechanisms according to a given application's state and the phase of its execution.

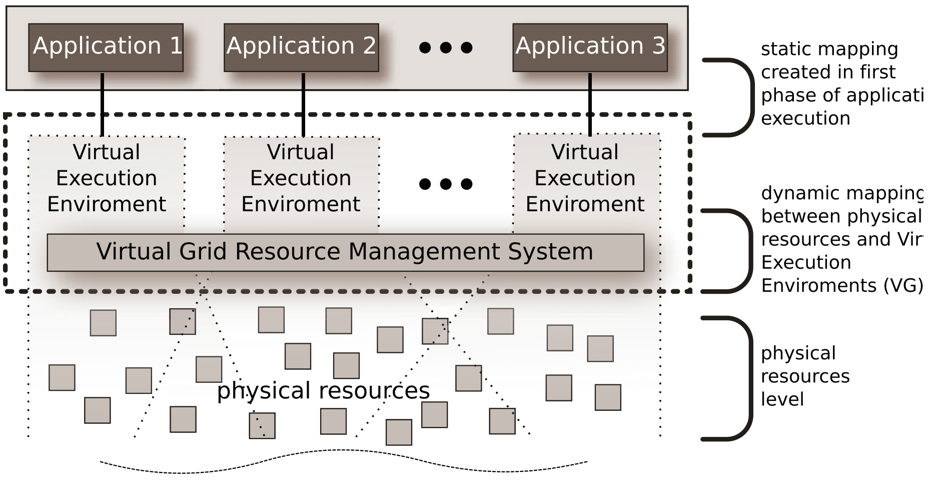
Application 1
Application 2
Application 3
Virtual Execution Enviroment
Virtual Execution Enviroment
Virtual Execution Enviroment
Virtual Grid Resource Management System
physical resources
static mapping created in first phase of applicati execution
dynamic mappinç between physical resources and Vir Execution Enviroments (VG)
physical resources level

Fig. 1. Hierarchical resource management in VGRMS

Individual levels of the management hierarchy are defined through the specification and deployment of policies regulating the access of applications to resources. This leads to the concept of a rule engine [9] in the implemented system, which was implied by the necessity to provide a flexible mechanism implementing management of selected resources. The use of a rule engine enables us to attain the following nonfunctional requirements:

- separation of application logic from its implementation,
- ability to change the management algorithm and its parameters without the need to recompile the application source code,
- the existing rules, implementing a given optimization strategy, do not need to be modified when enlarging the set of available optimizations.

The system policies, specified by an administrator, differentiate between management layers. The implementation of a system policy requires the controlled system state representation as facts. Facts are information about the system's actual state, parameters of available resources and their usage. They are aggregated in the working memory [6] of the rule engine [15]. The need to expose system state as facts requires a suitable sublayer implementation, which is done by the VGRMS system.

The autonomic system's decisions regarding the adustment of environment configuration to application requirements (realized by the rule engine) should be modifiable by the user. Such capabilities make it possible to correct errors in resource configuration and enable the application to achieve better performance. This is why the proposed structure of VGRMS offers support for manual control of resource allocation.

In such cases, the administrator is able to perform both direct resource allocation modifications and to modify the rules associated with their autonomic regulation (Figure 2). Hence, the management loops depicted in this figure can be executed automatically (loop feedback closure via VGRMS management layer elements) as well as by the operator (manual loop feedback closure) observing the resource state and reacting to certain situations via changes in resource configuration.

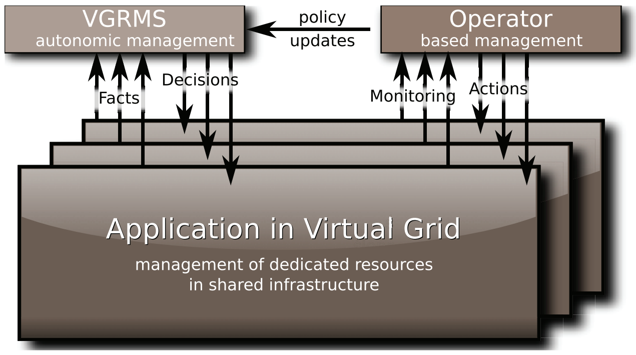

Fig. 2. Possible management flows in the VGRMS resource management environment

5 VGRMS ARCHITECTURE

The architecture of the VGRMS system, presented in this section, is constructed according to current trends in building distributed systems. One of the most important assumptions is the decomposition of the environment into smaller components, each of which exposes the functionality of a single service.

The designed Virtual Grid resource management system architecture consists of five[1] layers (Figure 3). These layers result from the established concept[2] of using virtualization as a management technique. The layers are as follows:

[1] Three upper layers comprised of VGRMS components and two lower layers related to components that represent resources and realize virtualization.

[2] Resource management, as presented in Section 4.

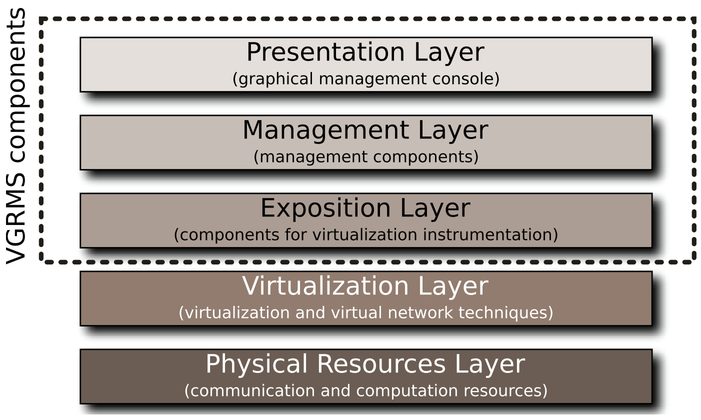
VGRMS components
Presentation Layer
(graphical management console)
Management Layer
(management components)
Exposition Layer
(components for virtualization instrumentation)
Virtualization Layer
(virtualization and virtual network techniques)
Physical Resources Layer
(communication and computation resources)

Fig. 3. Layered architecture of VGRMS components

Physical resource layer – the heterogenous resources constituting the infrastructure that typical computing centers are equipped with.

Virtualization layer – this layer creates an abstraction of physical resources through proper mechanisms. Components in the virtualization layer are responsible for:

- Virtual Machine (VM) creation,
- enabling dynamic modification of virtual machine configuration (changing the mapping and the level of physical resource usage). This is accomplished by modifying the dynamic VM parameters,
- virtual network creation,
- enabling dynamic modification of virtual network parameters.

The presented virtualization layer is also responsible for provision of an infrastructure enabling distribution of the operating system for VM instances.

Exposition layer – components added to adjust and unify virtualization layer mechanisms to simplify the interfaces realizing basic VGRMS system functions related to resource management:

- representing monitoring data and system configuration using facts,
- provision of services implementing search for and localization of resources (specifically, VGRMS system components realizing resource management); registration and retrieval of other components,
- resource management policy realization through exposed effectors.

Management layer – the main task of this layer is to process information about environment state and running procedures related to resource allocation for distributed applications. During runtime, information about the state of system components is collected as facts. This information is then processed by a rule engine in order to take proper actions.

Presentation layer – this layer consists of components enabling the end user to affect the state of other system elements. It is realized through a graphical interface in the form of the VGRMS system console. Each console can be connected with one or more component instances representing and managing virtual Grids. The presentation layer components gain access to other VGRMS components through lookup services located in the exposition layer.

The VGRMS components can be located at various points in the physical infrastructure as they communicate via the network. The presented layers do not impose any restrictions on component allocation.

6 IMPLEMENTATION DETAILS

The need for resource management based on complex heterogeneous mechanisms has led us to choose the Java Management Extensions (JMX) [20, 21] technology for developing the prototype of VGRMS. By using this technology we can create and unify diverse methods of distributed object management in the form of an uniform and clear programming interface.

All VGRMS node components are represented as JMX MBeans. While initializing the VGRMS node, a `MBeanServer` is instatiated on it with a `NodeMaster` registered. The other node components that are also MBeans – `VM-M` representing a virtual machine and `VG-M` representing a virtual Grid – are created and registered by the `NodeMaster` in the `MBeanServer`. At a lower level of abstraction, the `VM-M` functionality is accomplished through the Xen [2] mechanisms. The MBeans wrap the Xen Management API, an interface enabling remote configuration and management of virtual environments.

The node also contains other MBean components that collect information about the system state. These components are `NetMonitor`, which uses `Jpcap` library [10] for analyzing network communication packets, and `OSMonitor` for collecting information about operating system state and exposing it through a JMX interface. `OSMonitor` is built on top of monitoring systems based on Common Information Model (CIM)[3]. This standard was used as one of the possible sources of information about the host operating system. CIM capabilities related to control and management were not used.

Regarding the technology for building virtual networks over physical nodes located in different LANs, a virtual Ethernet switch was used. This concept is taken from the Virtual Distributed Ethernet (VDE) [23] project. It relies on creating virtual *switch*es on nodes connected with a tunnel. The type of tunneling method can

[3] The open-source OpenPegasus (`http://www.openpegasus.org/`) implementation standard was used in the execution environment.

vary as the VDE *switch* communicates with tunneling software through standard input/output streams.

The rule engine which implements system logic was developed using a commercial (the software is available for academic use at no cost) implementation of Java Specification Request (JSR)-94 („*Java^{TM}Rule Engine Application Programming Interface (API)*" – a rule engine for the Java platform) specification. Jess is a rule engine and a scripting environment, equipped with an extensive language for building and representing rules (based on Lisp). The environment is a compact, lightweight and highly efficient [1, 24] implementation of a rule engine for Java. The built-in scripting language enables direct access to Java API, facilitating easy integration both with the application and with the Java packages implementing system functionality (e.g. statistical packages, scheduling services, handling of complex structures, Graphical User Interface (GUI) interface, etc.).

The graphical VGRMS console, implemented in the Spring-RCP technology, simplifies system management. This console enables, among others, modification of the policy of system operation, tracing application execution and monitoring the level of resource usage.

Apart from the above mentioned basic technologies, such as JMX, Xen, VDE and Jess, used to implement VGRMS system functionality, frameworks supporting the process of software creation should be mentioned. The complexity of VGRMS software would make it very difficult to manage without the Spring[19] framework. The ability to invoke VGRMS with the use of configuration files supporting inversion of control (Inversion of Control (IoC)[7]) techniques and dependency injection Dependancy Injection (DI)[7] simplifies code maintenance and increases its flexibility.

7 CONCLUSIONS

The design and evaluation of the VGRMS system demonstrate the capabilities of the presented resource management model and selected virtualization technologies. Prototype installation has confirmed that complementing the available technologies with additional components (mentioned in Section 4) and providing the functionality enumerated in Section 3 results in a system which satisfies most of the stated VG requirements.

The implemented system provides a foundation for VGRMS further development and extension of current capabilities. A significant improvement could be achieved by introducing more complex analysis and processing of information concerning system state. This would allow us to construct better heuristics[4] for VG system management. Another field of improvement concerns the techniques of exposing state and making decisions based on actual system information represented as facts for the rule system. In some situations it might be desirable to extend the fact set

[4] By heuristics we mean diverse algorithms for resource allocations. These algorithms are based on performance data collected by system.

representing the system state with information derived from event streams. This would make the system more reactive and able to identify complex symptoms by processing many parallel event streams, using them as policy rule parameters.

Acknowledgement

The research presented in this paper has been partially supported by the European Union within the European Regional Development Fund program No. POIG.01.03. 01-00-008/08.

REFERENCES

[1] Adamczyk, J.—Chojnacki, R.—Jarząb, M.—Zieliński, K.: Rule Engine Based Lightweight Framework for Adaptive and Autonomic Computing. In ICCS '08: Proceedings of the 8th International Conference on Computational Science, Part I, Springer-Verlag 2008, pp. 355–364.

[2] Barham, P.—Dragovic, B.—Fraser, K.—Hand, S.—Harris, T.—Ho, A.—Neugebauer, R.—Pratt, I.—Warfield, A.: Xen and the Art of Virtualization. In SOSP '03: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, New York, NY, USA 2003, pp. 164–177.

[3] Bent, J.—Venkataramani, V.—LeRoy, N.—Roy, A.—Stanley, J.—Arpaci-Dusseau, A.—Arpaci-Dusseau, R.—Livny, M.: Grid Resource Management. Kluwer Academic Publishers 2003.

[4] Buyya, R.—Chapin, S. J.—DiNucci, D. C.: Architectural Models for Resource Management in the Grid. In GRID 2000, pp. 18–35.

[5] Drotz, A.—Gruber, R.—Keller, V.—Thimard, M.—Tolou, A.—Tran, T. M.—Cristiano, K.—Kuonen, P.—Wieder, P.—Wldrich, O.—Ziegler, W.—Manneback, P.—Schwiegelshohn, U.—Yahyapour, R.—Kunszt, P.—Maffioletti, S.—Sawley, M. C.—Witzig, C.: Application-Oriented Scheduling for HPC Grids. Technical Report TR-0070, Institute on Resource Management and Scheduling, CoreGRID – Network of Excellence, February 2007.

[6] Forgy, C. L. Rete: A Fast Algorithm for the Many Patterns/Many Objects Match Problem. Artificial Intelligence, Vol. 19, 1982, No. 1, pp. 17–37.

[7] Fowler, M.: Inversion of Control Containers and the Dependancy Injection Pattern. Technical report, ThoughtWorks, January 2004.

[8] Funika, W.—Korcyl, K.—Pieczykolan, J.—Skitał, L.—Bałos, K.—Słota, R.—Guzy, K.—Dutka, L.—Kitowski, J.—Zieliński, K.: Adapting a HEP Application for Running on the Grid. Computing and Informatics, Vol. 28, 2009, No. 3, pp. 353–367.

[9] Friedman Hill, E.: Jess in Action: Java Rule-Based Systems. Manning Publications Co., Greenwich, CT, USA 2003.

[10] Jpcap library manual. Online. `http://http://netresearch.ics.uci.edu/kfujii/~jpcap/doc/`.

[11] Kosiński, J.—Zieliński, K.: Enhancing Grid Computing With Virtual Grid Concept Model. In Marian Bubak, Michal Turaa, Kazimierz Wiatr (Eds.): Cracow Grid Workshop 2007, Kraków, Poland, October 2007, pp. 291–298.

[12] Krauter, K.—Buyya, R.—Maheswaran, M.: A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing. Software – Practice and Experience, Vol. 32, 2002, No. 2, pp. 135–164.

[13] Krsul, I. V.—Ganguly, A.—Zhang, J.—Fortes, J. A. B.—Figueiredo, R. J.: VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing. In Proceedings of Supercomputing Conference, November 2004.

[14] Kryza, B.—Dutka, L.—Sota, R.—Kitowski, J.: Dynamic VO Establishment in Distributed Heterogeneous Business Environments. In G. Allen, J. Nabrzyski, E. Seidel, G. D. van Albada, J. Dongarra, and P. M. A. Sloot (Eds.): Computational Science – ICCS 2009, 9th International Conference, Baton Rouge, LA, USA, May 2009, Springer LNCS Vol. 5545, pp. 709–718.

[15] Sandia National Laboratories. Jess, the Rule Engine for the Java Platform. Online.

[16] Livny, M.—Raman, R.: High-Throughput Resource Management. In Ian Foster, Carl Kesselman (Eds.): The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann 1998.

[17] Schwiegelshohn, U.—Yahyapour, R.—Wieder, P.: Resource Management for Future Generation Grids. Technical Report TR-0005, Institute on Resource Management and Scheduling, CoreGRID – Network of Excellence, May 2005.

[18] Seidel, J.—Wldrich, O.—Ziegler, W.—Wieder, P.—Yahyapour, R.: Using SLA for Resource Management and Scheduling – A Survey. Technical Report TR-0096, Institute on Resource Management and Scheduling, CoreGRID – Network of Excellence, August 2007.

[19] Spring framework. Online. `http://www.springframework.org/`.

[20] Sun Microsystems, Santa Clara, CA, USA. Java Management Extensions (JMX) Specification, Version 1.4, JSR 160, 2006.

[21] Sun Microsystems, Santa Clara, CA, USA. Java Management Extensions (JMX) Specification, Version 2.0, JSR 255, February 2008.

[22] Tonellotto, N.—Wieder, Ph.—Yahyapour, R.: A Proposal for a Generic Grid Scheduling Architecture. In Sergei Gorlatch and Marco Danelutto (Eds.): Proceedings of the Integrated Research in Grid Computing Workshop, pp. 337–346, Università di Pisa, November 2005, CoreGRID Technical Report TR-0015.

[23] Vde, project homepage. Online. `http://sourceforge.net/projects/vde/`.

[24] Young, C.: Microsoft's Rule Engine Scalability Results – A comparison with Jess and Drools. Technical report, Solidsoft, September 2005.

Joanna Kosińska is a research assistant at the Department of Computer Science, AGH University of Science and Technology, Kraków since 2001. Her main interests focus on distributed environments and portal technologies based on Java. She has participated in several national and international research projects, mainly EU-funded.

Jacek Kosiński is a research assistant at the department of Computer Science, AGH University of Science Technology, Kraków since 2000. His main interests focus on computer networks and operating systems. He is a CCNP instructor. He has participated in several national and international research projects, mainly EU-funded.

Krzysztof Zieliński is a Full Professor and Head of the Institute of Computer Science at AGH-UST. His interests focus on networking, distributed computing, object-oriented distributed system engineering and SOA-related technologies. He is the author/co-author of over 150 papers in the above mentioned areas. He served as co-chair and PC member of many conferences and workshops worldwide. He is a member of the IEEE Computer Society, ACM, and the Polish Academy of Sciences (Computer Science Chapter).