Comprehensive Formalization and Modeling of the Network and System Virtualization Process

Dimitrios Kontoudis

School of Information Sciences

Department of Applied Informatics

University of Macedonia

A dissertation submitted for the degree of

Doctor of Philosophy

Thessaloniki, Greece

May 2016

This Dissertation Advisory Committee (DAC) consists of the following members (listed alphabetically):

Associate Professor Alexandros Chatzigeorgiou
(Dept. of Applied Informatics, University of Macedonia, Greece.)

Associate Professor Konstantinos Lambrinoudakis
(Dept. of Digital Systems, University of Piraeus, Greece.)

Assistant Professor Panayotis Fouliras (*PhD Supervisor*)
(Dept. of Applied Informatics, University of Macedonia, Greece.)

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Assistant Professor Panayotis Fouliras, for the opportunity and for all his help and guidance over the years.

Last, but not least, I would like to thank my family for supporting me during this effort.

This research was supported in part by a grant provided under the Basic Research Funding Program (n.80698) of the University of Macedonia.

Abstract

The virtualization concept has been of interest to the academic community and IT business sector for more than forty years, providing a different approach to the realization, administration and provision of physical resources. A paradigm is introduced by which the end user is not aware of the details of the underlying physical infrastructure. This approach has been warmly adopted in many fields of computing, including computer networks. The Internet – being the prime example of a large scale and complex network instantiation [1] – has been the driving force behind the adoption of network virtualization as one of the key technologies in the field [2][3][4]. In a parallel development, the convergence of communications and computing – two, previously, distinct worlds – has introduced the use of computing servers acting as active network elements (e.g., routers) [5][6][7][8]. The core networking support in these *Virtual Network Environments* (VNEs) is based on the IEEE 802.1Q VLAN implementation, where virtual network segments are established on top of physical switches – the latter being provided by the server's hardware features. A thin software layer - the hypervisor - works as a virtual Ethernet switch, supporting queues for each VLAN in the system's memory. Consequently, the network's last-hop switch has been shifted from a dedicated active network element to become a characteristic of the hypervisor or of the physical server's hardware [9][10]. The aforementioned facts, along with the application of new technologies that lead to demand for novel services (wireless/mobile networking, Cloud computing, Networking As A Service, etc.) [11] have considerably increased the complexity of VNEs. Cisco Systems projects that thirty seven billion intelligent devices (including smart fabrics and pills) will connect to the Internet by 2020 – dramatically increasing the traffic load and operational complexity of the "The Internet of Everything" [12].

Efficient management of these architectures presumes the application of suitable information models. The latter provide the required standardization of abstraction of VNE elements and facilitate the construction and deployment of management methods. Despite the availability and benefits of several existing solutions, there are very few proposals that address the problem and the details of the introduction of computing servers in the network architecture, and of the hypervisor as a manageable

entity, in particular. Moreover, existing solutions do not treat the hypervisor as a whole entity; rather they only, indirectly, reference its involvement via abstractions of hosted virtual machine operations. This results in impediments in managing modern VNEs; thus, in assuring the quality of the offered service on an end-to-end basis.

In this dissertation we propose an information model that can conceptually abstract and describe physical or logical infrastructure elements participating in hypervisor host-based virtual network environments. In summary, the contribution of this dissertation is multifold: we initially provide a detailed review of existing modeling approaches employed in managing both VNEs as a whole, as well as autonomous system and networking components involved. We expose their pros and cons in a comparative review of the examined models and mechanisms using a conceptual categorization. We, then, describe our proposed solution: an information model that fits well within the context of a highly virtualized, host-based VNE, as those met in modern Cloud datacenters; then we apply a verification method for the validation of our approach. In particular, we draw upon one of the most important issues in Cloud environments – that of efficient resources allocation – and extend our model in order to include appropriate control logic and methods based on Statistical Process Control for managing hypervisor provisioned resources to a networking architecture. We test and validate our proposal extensively against actual VNE implementations. The verification provides a successful proof-of-concept of our proposed technique. Our proposed information model facilitates managing a VNE's element, be that a networking, computing system or other hypervisor-provisioned resource. The management application does neither need to be aware of the specifics of the underlying virtualization platform, nor to understand it.

Contents

List of Figures

List of Tables

Acronyms

AIX	Advanced Interactive Executive
APIC	Advanced Programmable Interrupt Controller
AS	Aggregation System
AUTOI	Autonomic Internet Project Framework
CIM	Common Information Model
CIMOM	Common Information Model Object Manager
CMIP	Common Management Information Protocol
CPU	Central Processing Unit
DBMS	DataBase Management System
DEN-ng	Directory Enabled Network next generation
DMI	Desktop Management Interface
DMTF	Distributed Management Task Force
HMC	Hardware Management Console
I/O	Input/Output
IaaS	Infrastructure as a Service
ICN	Information-Centric Networking
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
INDL	Infrastructure and Network Description Language
ISP	Internet Service Provider
IT	Information Technology
IX-MR	Individual & Moving Range control chart
KF	Kontoudis-Fouliras
LAN	Local Area Network
LCL	Lower Control Limit
LICL	Logical Infrastructure Composition Layer
LPAR	Logical PARtition
LPAR	Logical PARtition
MCUSUM	Multivariate CUmulative SUM control chart
MEWMA	Multivariate Exponentially Weighted Moving Average control chart
MIB	Management Information Base
MOF	Managed Object Format
MPLS	Multiprotocol Label Switching
NAAS	Networking As A Service
NAT	Network Address Translation
NDL	Network Description Language
NFV	Network Functions Virtualization
NIC	Network Interface Card
NML	Network Markup Language
NOVI	Networking innovations Over Virtualized Infrastructures

OGF	Open Grid Forum
OS	Operating System
PBNM	Policy Based Network Management
PBR	Policy Based Routing
PCI	Peripheral Component Interconnect
QoS	Quality of Service
RAM	Random Access Memory
RDF	Resource Description Framework
SDN	Software Defined Networking
SID	Shared Information/Data Model
SLA	Service Level Agreement
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol
SPC	Statistical Process Control
SSH	Secure Shell
TMForum	TeleManagement Forum
UCL	Upper Control Limit
UML	Unified Modeling Language
VFR	Multi-Virtual Route Forwarding
vgDL	Virtual Grid Description Language
VLAN	Virtual Lan
VM	Virtual Machine
VMM	Virtual Machine Monitor
VNE	Virtual Network Environment
VNMI	Virtual Network Management Information Model
VPN	Virtual Private Network
VXDL	Virtual Resources and Interconnections Description anguage
W3C	Word Wide Web Consortium
XML	eXtensible Markup Language
XSD	XML Schema Definition

Chapter 1 - Introduction

The convergence of communications and computing to a common design and operational entity is an inevitable reality, introducing new elements and technologies to network engineers that need to be taken into account when designing and implementing *Virtual Network Environments* (*VNE*s). VNEs are extensively based on virtualized infrastructures – both from the networking as well as from the computing realms. This concept is depicted in *Figure 1. A VNE based on hypervisor-provisioned resources* where a host computing server and its hypervisor provide the underlying networking infrastructure and resources upon which virtual servers along with their virtual networks operate. Some of these new elements can be quite novel and totally outside the traditional network design scope. In "traditional" physical or virtual network designs and operation approaches, network engineers had to consider dedicated network elements and characteristics (e.g., routers, communication connections and bandwidth, etc.) This has changed with the introduction of computing servers in the networking infrastructure, exploiting specific capabilities in VNE implementation and operation, while maintaining a cost/performance balance [5][6][7][8]. In the context of this dissertation *computing servers* refer to virtual systems (regardless of the virtualization platform) that employ a UNIX/Linux based operating system, along with a special configuration for allowing networking activities (e.g., a routing subsystem for a network protocol) to be served and managed from the node. These systems are commonly referred to as virtual machines. This has been made possible due to advances in server virtualization (also referenced to as system or machine virtualization) and the introduction of the hypervisor (implemented by a specific thin software layer also known as the virtual machine monitor).

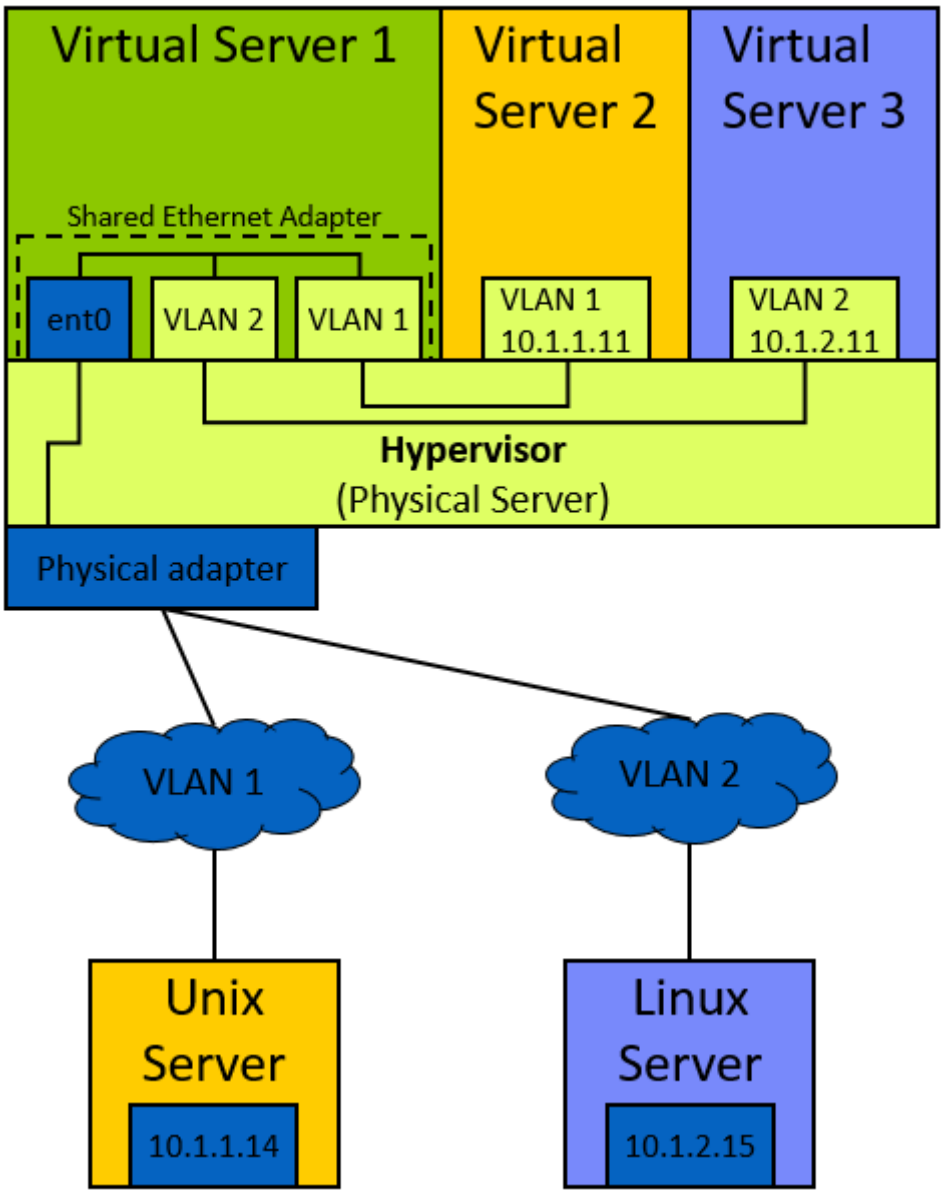


Figure 1. A VNE based on hypervisor-provisioned resources

Managing such network environments presents an increasing need for the infrastructure detailed characteristics to be represented in a formal, standardized and structured manner, regardless of the representation stakeholders (researchers, providers, end-users). This is facilitated by the use of suitable information and data models that allow for better description of the infrastructure components involved and for the organization of their characteristics and interrelations. Standardization enables, among else:

- The development of suitable management software.
- The formulation of accurate SLA agreements as precise requirements are depicted in a formal (therefore, less error-prone) way.
- The creation of supporting environments in order to enable collateral actions, such as accounting, auditing, standards compliance, etc.

To tackle these problems, from standardization and management points of view, it is essential that the infrastructure's architectural, technological and operational complexity be semantically represented, allowing for hypervisor support. The latter element has been largely overlooked in current research.

This dissertation is concerned with the requirements analysis, implementation and verification of an information model for managing modern collaborative networking architectures where computing systems and hypervisors are deployed as active network elements, as found in Cloud datacenters. In the remainder of the chapter, we present the motivation of our dissertation and its outline.

1.1 Motivation

Communication networks present an ever increasing use of computing servers and hypervisors assuming the role of active network elements. This fact, along with the adoption of the virtualization concept in computer networks, introduces new challenges that need to be accounted for when managing such networks. Advances in server virtualization (also referenced to as system or machine virtualization, implemented by a specific thin software layer, the *hypervisor* – also known as *virtual machine monitor*) have facilitated the outsourcing to the server hardware of network-related infrastructure elements and their characteristics (e.g., routers, communication connections and bandwidth, etc.) [5][6][7][8]. Particular implementations of communication networks, for example in *network testbed projects* [8][13], in *network simulation environments* [5] and in *scenario-based infrastructure management* [6], increasingly rely on computing servers to act as active network nodes, as these servers allow for flexible experimentation on new architectures, protocols and services.

In managing such infrastructures, several modeling approaches have been proposed that enable the creation of management solutions. The state-of-the-art information models available to the systems and computer network domains are the Common Information Model (CIM) [14] (proposed by the Distributed Management Task Force – DMTF [15]), the Shared Information/Data (SID) model [16] (proposed by the TeleManagement Forum – TMForum) and the Directory Enabled Network next generation (DEN-ng) [17] (proposed by the Autonomic Communications Forum [18]). Each of these models, as discussed in later sections, focuses on network aspects from a different perspective. CIM applies a holistic approach conceptualizing computing systems and networks in general, whereas SID and DEN-ng spawn from the telecommunication industry and better represent business (as well as technical details), in the telco context. CIM has been used as the basis for the creation of other information models that focus on specific application areas, such as hypervisors and virtual machines [19], and virtual network environment provisioning [20]. An overview of the three main information models can be found in [21] and their use in a network virtualization context is discussed in [22] and [23].

Two other related research areas can be, collectively, identified as presenting modeling elements. The first area refers to network description languages (NDLs) [24], some of which do contain small information models and other modeling approaches. These languages have been designed with the goal of imprinting network characteristics in a structured and hierarchical manner. Code developed in any of these languages can be used in a diverse array of applications (e.g., as input to special-purpose software). Consequently, NDLs are used as modeling tools for the design and application of abstractions on the physical and logical representation layers of the networking infrastructure.

The second area refers to work based on the Management Information Base (MIB) concept [25] – databases storing management information about devices and applications. These databases are populated and used by management applications, using specialized protocols such as the Simple Network Management Protocol (SNMP). MIBs result in data models of managed entities and this concept can be applied to abstract physical and logical device state and configuration, as well as application specific information.

Furthermore, modeling elements are apparent in testbed network implementations, pursued in academic and/or private sector (mainly EU-funded) projects. These introduce some level of formalization across different layers of their architecture. Indicative projects are: GENI, Emulab, Planetlab, VINI, Federica, Nitos, Etomic, PanLab, Wisebed, Geysers, Novi and SAIL. Information models can be found in the Novi and Geysers projects (Novi [22] and LICL [26] proposals, respectively). Finally, special mention should be given to Software Defined Networking [27], a new paradigm in network architecture and management. This is a promising approach [28], with its own data model, simplifying change facilitation in the network control logic [29]. SDN enjoys broad industry support given the flexibility it offers in data center fabric management.

1.2 Objectives

Based on the aforementioned information, it is clear that a considerable amount of work has been done in creating information models in the computing and networking areas (cf. Chapter 2). Nevertheless, in the context of hypervisor-based architectures, a suitable model necessary for VNE detailed characteristics to be represented in a formal, standardized and structured manner, regardless of its stakeholders (researchers, providers, end-users), has not been proposed. Such a model will allow better comprehension of a VNE as an entity and the qualitative reasoning about its characteristics and their interrelations (e.g., how a part or a VNE characteristic will be affected when a new change is introduced in it). Moreover, this standardization will enable better communication among all parties involved. The VNE model will, therefore, pertain to all stages of a VNE lifecycle (design, provision, operation and administration, abolition), enhancing supporting actions (e.g., monitoring) and collaterals (documentation). A well designed and instrumented model will make specific artifacts possible, such as:

- the objective evaluation of VNEs and their characteristics against specific criteria (e.g., performance, monitoring or operations-wise).
- the formulation of more accurate SLAs between providers and users as each side's precise requirements will be depicted in a formal and unambiguous (therefore, less error-prone) way.

- the creation of supporting environments in order to implement collateral actions, such as performance monitoring, accounting, auditing, standards compliance, etc. Standards compliance is becoming more and more popular in several implementations due to the ever increasing Government and other authorities' guidelines compliance requirements.
- the creation of software automation solutions (such as the creation or abolition of a VNE, SLA conformance monitoring, etc.)

The main objectives of our work can be summarized as follows. In the context of the model creation; firstly, we define the requirements for an information model targeted at the hypervisor-based VNE architectures. Secondly, we design and construct the model that will integrate the desired functionality regarding features, and we implement the required platform dependent instrumentation.

Finally, the last objective is to provide a proof-of-concept of our work against an actual architecture, as found in modern Cloud datacenter VNEs, to verify the correctness of the defined model.

1.3 Research areas

The contributions of this dissertation are centered on the following topics:

- Requirements analysis.
- Design and implementation of an information model in the context of a systems virtualized networking landscape, accounting for the hypervisor layer in the architecture.
- A verification technique to verify the correctness and applicability of the model against an actual Cloud data center based testbed.

In the remainder of this section we present the contributions in more detail. The original contribution of our work can be summarized as follows.

Information model requirements, design and implementation. The detailed design of a holistic hypervisor-based VNE information model which provides a standard

concept, scheme structure and vocabulary for describing the VNE and all its constituent components. This is facilitated in a way that allows for the support of management (or other) applications, regardless of the context in which the VNE is deployed (i.e., a particular industry sector, business environment, etc.) Thus, using the model as a single point of information makes it possible to define and abstract, in a common manner, similar concepts used in different implementations.

Verification method. The second contribution is a verification method, realized via a proposed model's implementation on a virtualized datacenter environment. We have chosen an actual problem apparent in Cloud architectures: the infrastructure dynamic resources allocation for meeting the served workload varying demand. Our approach involves a statistical method, Statistical Process Control (SPC), facilitated via the model's instrumentation (implemented, platform-specific, provider). We have introduced the method and pursued a thorough performance evaluation study. Based on test results, our study shows that the proposed approach has a ground foundation and practical applicability for real-life environments. Our statistical approach, although applied to other contexts, is unique in the resource allocation work realm and constitutes a further, autonomous, research direction.

In the remainder of this section, we present the contributions in more detail.

1.3.1 Information model requirements, design and implementation

A model is a developed representation of a real world system. Different types of models exist that can be used for representing the structured and unstructured information, relationships and elements in a given environment [30][31]. This diverse variety of information does not always fit into a particular type of model. In many real life cases it may be necessary to employ or combine different approaches to reach some suitable, per case, representation. For example, a knowledge model is a way to describe what data means and where it fits. It allows knowledge abstraction and, thus, an appreciation of how different pieces of information relate to each other. Likewise, a semantic model (a kind of knowledge model) can allow for the qualitative processing of modeled entities, and can help in answering questions on patterns of behavior and in discovering relationships between pieces of information. Several

types of models exist: conceptual, mathematical and statistical, entity-relationship, information and data, business, and so on. Each of those can serve a particular purpose, although overlapping functionality is apparent across different model types [23][31][32].

A VNE holistic information model must adhere to certain criteria – modeling, as well as context-scope wise. The fundamental characteristics are that of a) extensibility, and b) neutrality with respect to implementation platform, language and protocol. This will allow for the created model to be adaptable to changing information abstraction requirements, thus to different contexts. By definition, a holistic model cannot be limited to a particular context and must address all constituents of a VNE. The model will form the common source for information definition upon which implementation specific data models will be constructed and used by management (or other) applications. Overall, the holistic VNE model, by its purpose and design, will need to provide a standard concept, scheme structure and vocabulary for describing the VNE and all its constituent components. This must allow for the support of management (or other) applications, regardless of the context in which the VNE is deployed Consequently, the model will form the common source for information definition upon which implementation specific data models will be constructed and used.

1.3.2 Verification method

In order to illustrate the applicability of the proposed information model we have pursued a real-world feasibility application. It has become apparent [33][34] that resource management is a multi-factored, important challenge, apparent in different infrastructure configurations, ranging from stand-alone application provisioning to diverse networking architectures and commercial cloud environments. Resource management, however, is a crucial element in avoiding service level violations while operating in a ubiquitous, cost-saving context. The mathematical science of Statistics is considered a potential source of powerful tools for computer performance evaluation and related applications [35] and can be used in the server resource management context, enabling data-driven decision making.

Although we designed an information model that, in theory, can abstract VNEs and facilitate management solutions, we had to assure the model's correctness. As part of the verification approach, we engaged in this dissertation in two directions:

- A model schema and provider (instrumentation) extension for including a method to dynamic (adaptive) computing server resource provisioning based on Statistical Process Control (SPC).
- The application of the model on a Cloud datacenter testbed serving real-world workloads.

Our contribution is twofold: *i)* the proposed information model is verified as a correct tool for application in VNEs, and *ii)* we present a complete solution for monitoring and action mechanisms that provide for:

- Continuous server resources monitoring.
- Data analysis based on SPC.
- Automatic initiation of corrective procedures for bringing server performance within acceptable levels.

We have applied our proposal on a testbed consisting of IBM pSeries AIX Unix infrastructure [36] using our own developed code and executed a two series of tests:

- One test based on the Temenos T24 Core Banking software [37], serving batch processes for a commercial financial institution, and additionally.
- Another test, further investigating on the same hardware architecture, a hypervisor-based virtual network architecture instantiating a virtual router configuration (using the 'gated' routing daemon) under stress-test via simulator tools (Apache Foundation "jmeter"[1] and IBM "nstress"[2] software packages).

Resource-wise, the VNE operation can be characterized by the allocation patterns of the core infrastructure resources (CPU, memory, etc.) and several actions can be based on monitoring these (SLA monitoring, accounting, VM operations based on resource consumption, etc.) The goal is to measure these resources both at the VM and at the VNE level so that proper decisions can be taken. Indicative resulting

[1] jmeter.apache.org
[2] www.ibm.com/developerworks/community/wikis/home/wiki/Power+Systems/page/nstress

management actions include node migration, resources addition/removal, SLA alert triggering, and other.

The proposed information model correctly abstracts the VNE constituents and characteristics, as well as the aforementioned core resources and their utilization metrics. Providers designed for these tasks are deployed at the target VMs. The model is not limited to the above metrics or scenario and can be extended to include other situations, as long as these can be instrumented by the providers at the VM level. Given that our proposal is based on CIM, the restrictions imposed by CIM are inherited. Thus, the management applications can be local or remote but, at present, CIMOM and the providers must be located on the machine that is being instrumented. If a VNE spans more than one VM or extends beyond the VMs run on a single hypervisor, then it is necessary to introduce an aggregation system (AS) where the management application will reside. The CIM client running on this system will collect the management data from the "N" VMs and perform the necessary calculations as per the applied scenario. In the case of multiple VMs running on a single hypervisor it is possible (in platforms that support this functionality) to host a CIM client on one of the VMs and monitor the resources at the hypervisor level, which is sometimes desirable as it may produce more meaningful results [38].

1.4 Structure of the dissertation

This chapter has discussed the motivation and the main objectives of our work and outlined the major contributions of this dissertation. The remaining chapters are structured as follows.

Chapter 2 discusses the state of the art of information modeling related to the objectives of the dissertation. Specifically, it presents information regarding modeling approaches in computing and networking systems and architectures, and elaborates on applicability in existing proposals and mechanisms in those domains.

Chapter 3 presents the proposed information model and describes its design along with its logical and functional characteristics.

Chapter 4 presents the proposed model checking technique as a mean to verify the correctness of the proposed model. Detailed implementation and verification results are presented and discussed.

Chapter 5 summarizes the contributions of this dissertation and outlines future work.

Chapter 2 – Survey of Related Models and Approaches

2.1 Introduction

This chapter provides an introduction to the system and network virtualization concepts, discusses current information modeling approaches available in computing and networking, provides a detailed qualitative analysis of these efforts, and presents the problem of the inadequate support for abstracting system hypervisor-based network architectures.

Current modeling research in Computing and Networking (as introduced in Chapter 1) presents two distinct groupings among available proposals: the state-of-the-art information models, offering elaborate and advanced approaches, and all other work which incorporates some form of modeling elements. In order for the diverse nature and characteristics to be presented in a structured manned we introduced a taxonomy with proposals classified according to:

- Their overall positioning based on their general characteristics.
- The modeling aspects addressed.
- The system virtualization aspects addressed.

This chapter is organized as follows. Section 2.2 introduces the virtualization concept and its effect on computer networks, provides definitions of terms and concepts used in this thesis and presents the classification schema used for providing a taxonomy of current proposals. Section 2.3 refers to the main information models. Section 2.4 introduces the Management Information Bases. Section 2.5 discusses the Network Description Languages. Section 2.6 presents other, smaller in scope, proposals. Section 2.7 provides a detailed discussion and comparison among the proposals. Finally, Section 2.8 concludes the chapter.

2.2 Background concepts

2.1.1 Physical resource abstraction and virtualization

With the introduction of the virtualization concept it has been made possible to logically divide a physical resource into several virtual ones, based on the available resources and their characteristics [39]. This concept was first introduced in the server hardware context; it is now prominent in most computing infrastructures layers and information technology areas, such as in networking, storage, operating systems, application and other (cf. *Figure 2. Indicative virtualization areas*).

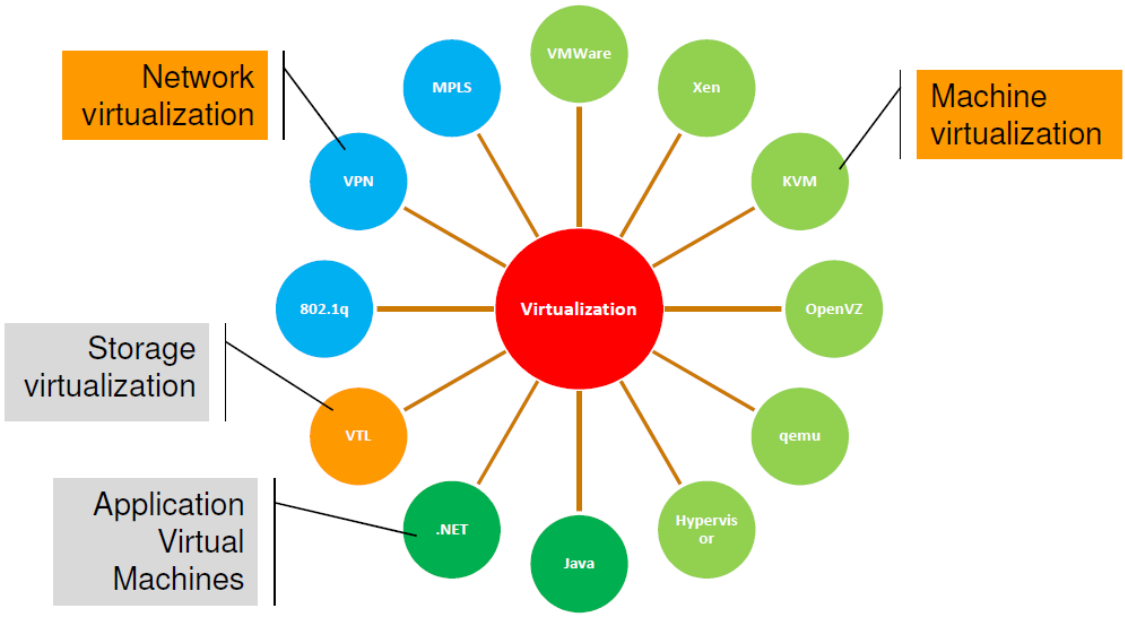


Figure 2. Indicative virtualization areas

In essence, virtualization increases flexibility by decoupling the hardware resource and the services and applications supported by that resource, from a specific physical hardware platform. It allows the establishment of multiple virtual environments on a shared hardware platform. Virtualization is commonly defined as a technology that introduces a software abstraction layer between the hardware, and the operating system and applications running on top of it. This abstraction layer is, generally, called *Virtual Machine Monitor* (VMM) or *hypervisor* and, basically, hides the physical resources of the physical computing system from the operating system (OS) running on it. Since the hardware resources are directly controlled by the hypervisor

and not by the OS, it is possible to run multiple (possibly different) OSs on the same hardware, simultaneously.

The virtualization layer resides at a higher privilege level than the clients, and can interpose between the clients and the hardware. This means that it can intercept important instructions and events and handle them in a special way before they are passed to the hardware. For example, if a client attempts to execute an instruction on a virtual device, the virtualization layer may have to intercept that instruction and implement it in a different way on the real resources under its control. Each client is presented with the illusion of having exclusive access to its resources, thanks to the management performed by the virtualization layer. The virtualization layer is responsible for maintaining this illusion and ensuring correctness in resource multiplexing. Consequently, virtualization promotes efficient resource utilization via sharing among clients, while maintaining isolation among clients (who need not know of each other's existence). Virtualization also serves to abstract the real resources to the client, thus decoupling the client from the real resources and facilitating greater architectural flexibility and mobility in system design.

Virtualization is unquestionably one of the major trends in information technology today, used in enterprise systems, networks, service providers, home desktops, mobile devices, and production systems, among other venues [40].

2.1.1.1 The Hypervisor

The *hypervisor* forms the foundation of the virtualized system software stack [41]. Also referred to as the VMM, it virtualizes all the resources of a real machine, including CPU, devices, memory, and processes, creating a virtual environment known as Virtual Machine (VM). Software running on the virtual machine has the illusion of running on a real machine, and has access to all real machine resources through virtualized interfaces. The hypervisor manages the real resources, and provides them to the virtual machines. Since it supports one or more virtual machines, it is responsible for making sure all real machine resources are properly managed and shared, and for maintaining the illusion of virtual resources presented to each virtual machine. The hypervisor provides high-level memory, page table management as well

as partition scheduling policies. Some hypervisors handle virtualization of I/O devices and resources, while others delegate this responsibility to special-purpose I/O partitions. Interrupts are handled in a variety of ways. They can preempt the running partition and force a context-switch to the interrupt destination partition. Other hypervisors mitigate interrupts by preempting the running partition, acknowledging the interrupt, scheduling its delivery to the appropriate logical partition and then returning to the preempted partition. Finally, some hardware has interrupt routing, allowing hardware assisted interrupt mitigation [42]. A high level depiction of the aforementioned core tasks and architecture, specific to the XEN hypervisor, is given in *Figure 3. High level architecture of the XEN hypervisor* (from http://www.xenproject.org). As illustrated in the figure, the hypervisor acts as an intermediate between the physical machine's hardware resources and the virtual machines using those resources. The hosted operating systems can be either "paravirtualized", i.e. aware of the underlying hypervisor and supporting direct calls to it leveraging better performance for certain tasks, or "unmodified", i.e. unaware of the underlying hypervisor. In the latter case the operating system does not use any special code features - it is the mainstream consumer product version.

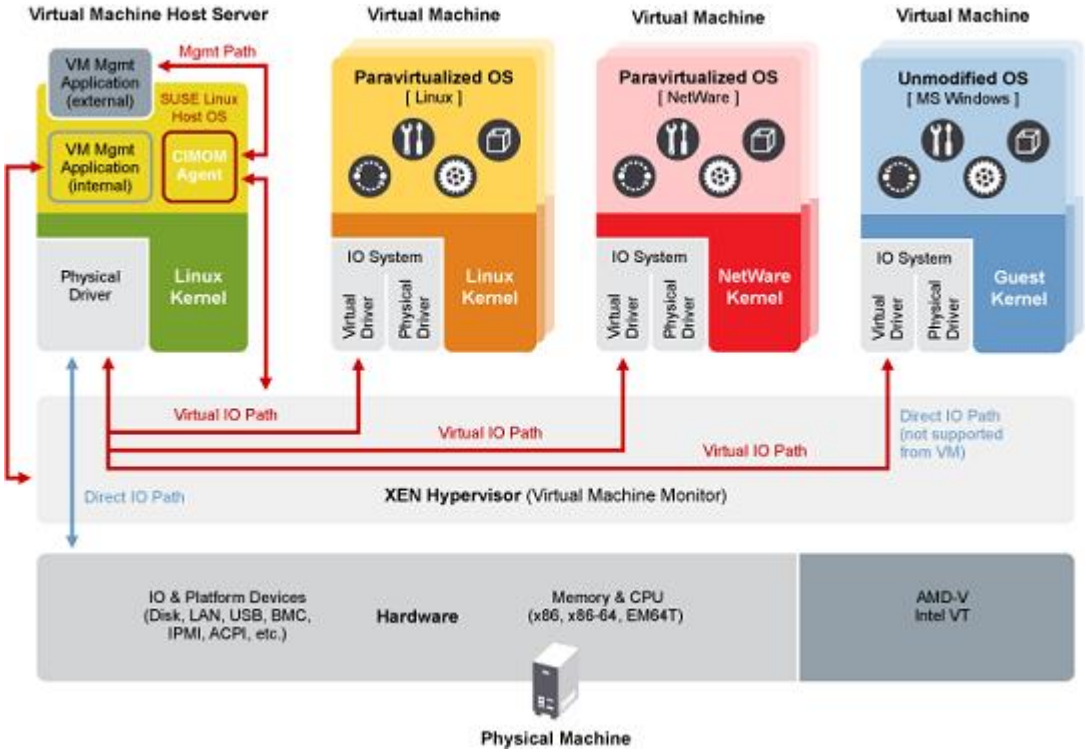


Figure 3. High level architecture of the XEN hypervisor

The main roles of the hypervisor in emulating and managing the primary resources in a modern computer system can, thus, be summarized as follows [43],[41]:

Processor cores: The first responsibility of any virtualization solution is the ability to arbitrate access to the processor cores, as well as sometimes emulate their functionality. The hypervisor needs to cope with the dynamic addition and removal of VMs as they get started and stopped, as well as the ability to schedule multiple VMs on the same system. In addition to scheduling the VMs, the virtualization layer uses the scheduler to run some of its own functionality, at every clock tick. For example, the hypervisor needs to check whether to deliver timer interrupts to the VM during a clock tick. The virtualization layer may also emulate the underlying processor, since the processor cores presented may differ in capabilities from the actual ones.

Memory: The hypervisor is also in charge of managing the host physical memory of the system. Each VM is assigned some guest physical memory that the guest OS can manage and assign as virtual memory to the applications. Here, the virtualization layer can take a number of different approaches to managing the allocation of host physical memory among the various VMs. One popular use of virtualization is server consolidation. This relies on the virtualization layer to dynamically manage the allocation of physical memory to different VMs in the system.

I/O devices: Another important functionality of the virtualization layer is dealing with I/O devices such as a network interface card (NIC). Today's virtualization solutions commonly create virtual devices that they assign to the guest VMs. The virtualization layer needs to manage these virtual devices and emulate their behavior. There is also the option of dedicating physical devices to the VMs. In this case, the virtualization layer only needs to assign and reclaim the devices; it does not need to emulate their behavior. In both cases, however, the emulator is needed to emulate the PCI (Peripheral Component Interconnect) bus. The guest OS interacts with the PCI bus during boot-up to discover and configure the available devices.

Interrupts and Timers: The virtualization layer must also emulate the interrupt subsystem and timers for the guest VMs. For interrupts, the main parts of the underlying hardware interrupt subsystem emulated by the hypervisor include the I/O APIC (which routes interrupts from devices to processor cores) and local APICs (which are attached to each core as an interface to the interrupt subsystem for sending and receiving interrupt messages to and from other local APICs or the I/O APIC). These APICs are emulated so that the guest OS can use them to configure and manage interrupts inside the VM, while physical APICs are controlled by the hypervisor for the management of the physical platform. As stated earlier, the hypervisor utilizes the physical timer devices to get periodic clock ticks itself. Hence, it can emulate the timer device for each guest VM. Today, VMs can be scheduled on different cores and as such, the interrupts and timers must first go through the hypervisor which will deliver the interrupts to the VM by utilizing its knowledge of the VM's current location.

While the basic function of a Hypervisor is to virtualize a physical host to enable running of multiple virtual hosts (or VMs), commercial hypervisor offerings come with differing feature sets. The modules that provide the constituent features are given different names in different product offerings. Hence, for clarity purposes, it is necessary to identify *from a functional perspective* a set of baseline features of a hypervisor. These features support all the necessary and required functionality of a virtualized host. The required functionality is also called Virtual Machine Manager functionality. These baseline features or functions are [44]:

- Execution Isolation for Virtual Machines: Scheduling of VMs for execution, management of the application processes running in VMs such as CPU and memory management, context switching between various processor states during the running of applications in VMs, etc.
- Devices Emulation & Access Control: Emulating all network and storage (block) devices that different native drivers in VMs are expecting and mediating access to physical devices by different VMs.
- Execution of Privileged Operations for Guest VMs: Certain operations invoked by guest operating systems, instead of being executed directly by the

host hardware, may have to be executed on its behalf by the hypervisor, because of their privileged nature.

- Management of VMs: Setting configuration parameters for VMs (VM images) and control of VM states (start, pause, stop, etc.).
- Administration of Hypervisor Platform and Hypervisor Software: This involves setting of parameters for user interactions with the hypervisor host as well as hypervisor software and configuration of virtual network inside the hypervisor.

2.1.1.2 The evolution of virtualization

The roots of virtualization are best seen in the computer "time sharing" practices of the late 1950s and early 1960s. Time-sharing was necessary in those distributed computing environments because technology was extremely expensive. It was not practical to dedicate a computer system to a single user; thus, a scheme for dividing the resources among many users was developed. These schemes often used "executive programming" which employed a combination of software and hardware in order to delegate (based on a specified time interval) which user would receive attention from the central processing unit at a particular time. This process is similar to what at present we refer to as virtualization, i.e., that a layer of abstraction is created in order to logically assign the use of a computer asset.

In 1967 IBM announced the IBM\360 version 67 [45], which was the first computer system to contain *virtual memory*; a method in which disk space is used to expand the available memory (RAM) size of a computer system. Later, in the 1970s, the notion of the *virtual machine* emerged. With the virtual machine, an entire system (software and hardware) could be emulated in a contained environment. This virtual machine was the first and closest form of virtualization, as we know it today. With the introduction of Intel's recent release of its "Vanderpool" technology (and AMD's subsequent "Pacifica"), which provides hardware native server virtualization functionality, the concept of virtualization has been taken one step further toward becoming a high-impact, common practice in the enterprise [46].

The above descriptions all contain some allusion to either "combining" or "multiplying" a computer asset. By inserting nonfigurative layers in between hardware and/or software components, more control can be exercised on one or both of the separated assets. This middle layer is usually some sort of specialized software (system firmware) that allows for the manipulation of the assets through logical separation or combination.

Similarly, virtualization techniques can be applied to other IT infrastructure layers – including networks, storage, laptop or server hardware, operating systems and applications. This blend of virtualization technologies provides a layer of abstraction between computing, storage and networking hardware on one side, and the applications running on it on the other. The deployment of virtual infrastructure is non-disruptive, since user experience is largely unchanged. A few areas where virtualization is employed are [47]:

- Server Virtualization: Server virtualization enables multiple operating systems to run on a single physical machine, yet remain logically distinct with consistent hardware profiles. In addition, server virtualization can often take the place of the costly practice of manual server consolidation, by combining many physical servers into one logical server. The focus of server virtualization is on maximizing the efficiency of server hardware in order to increase the return on investment for the hardware.

- Operating System Virtualization: Through virtualization of operating systems, a single computer can accommodate multiple platforms and facilitate their operation simultaneously. This description is similar to the aforementioned server virtualization, but server virtualization alone does not necessarily provide the ability to run multiple platforms on a single server. OS virtualization allows different operating systems (or versions of the same one) to be executed on a single physical or virtual server. Also, OS virtualization is focused more on flexibility than efficiency. OS virtualization can be used to facilitate what is known as universal computing, where software and hardware

work together seamlessly regardless of the architecture or language for which they are designed.

- Application Virtualization: While most of the prevalent virtualization strategies focus on hardware infrastructure, an important and often overlooked method is application virtualization. With application virtualization (also commonly referred to as service virtualization) end-user software is "packaged," stored, and distributed in an on-demand fashion across a network. This virtualization strategy goes hand-in-hand with the standardized web services initiative that is making waves in the IT industry today. Virtualized applications use a common abstraction layer, which defines a protocol, allowing them to communicate with one another in a standard messaging format. Thus, applications can invoke one another in order to perform requested functions. A virtualized application is not only capable of remotely invoking requests and returning results, but also ensuring that the application's state and other data are available and consistent on all resource nodes executing the application across a grid.

- Storage Virtualization: Being one of the most widely deployed virtualization practices, storage virtualization allows separate storage devices to be combined into a perceived single unit. Storage virtualization attempts to maximize the efficiency of storage devices in an information architecture. In a general sense, it means providing a logical, abstracted view of physical storage devices, hence anything other than a locally attached disk drive.

- Data/Database Virtualization: Data virtualization allows users to access various sources of disparately located data without knowing or caring where the data actually resides. Database virtualization allows the use of multiple instances of a DBMS, or different DBMS platforms, simultaneously and in a transparent fashion regardless of their physical location. Such practices are often employed in data mining and data warehousing systems.

- Network Virtualization: The term network virtualization is used to describe a number of different things. By virtualizing a network, multiple networks can be combined into a single network, or a single network can be separated logically into multiple parts. Perhaps the most common of the involved ideas are those of the virtual private network (VPN) and of the Virtual LAN (VLAN). VPNs abstract the notion of a network connection, allowing a remote user to access an organization's internal network just as if she were physically attached to that network. VPNs are a widely implemented idea, and they can use various technologies. VLANs, on the other hand, present a method of logically segmenting a single Layer-2 network into multiple, distinct, broadcast domains (referred to as VLANs). The latter are mutually isolated so that packets can only pass between them via one or more routers.

Many parties, industrial, commercial and academic alike, are actively involved in network virtualization research, occasionally in joint ventures. Research spans a wide variety of topics, ranging from very specific technical issues (interfacing, signaling and bootstrapping, resource and topology discovery, resource allocation, admission control, virtual nodes and virtual links, naming and addressing) to broader interest areas such as mobility management, monitoring, configuration and failure handling, security and privacy, and interoperability issues. A concise survey of network virtualization research is provided in [48] and [49]. Active [50], programmable [51] and overlay [52][53] networks also benefit, as well, from advances in system and network virtualization. Network virtualization architectures are discussed in detail in [54][55][56][57][58] and [59].

Software-Defined Networking [27] (SDN - described in section 2.6 of this chapter) is, at the time of writing of this dissertation, the latest mainstream approach and technology toolbox that affects, and is affected-by, network virtualization. SDN has emerged as a promising paradigm for making the control of communication networks flexible. SDN separates the data packet forwarding plane, i.e., the data plane, from the control plane and employs a central controller. Network virtualization allows flexible sharing of physical networking resources by multiple users (tenants). Each tenant runs its own applications over its virtual network, i.e., its slice of the actual physical network. A critical component for virtualizing SDN networks is an SDN hypervisor

[60] that abstracts the underlying physical SDN network into multiple logically isolated virtual SDN networks (vSDN), each with its own controller. The virtualization of SDN networks promises to allow networks to leverage the combined benefits of SDN networking and network virtualization. Consequently, it has attracted significant research attention in recent years [61].

2.1.1.3 Network virtualization and the use of computing servers

The features provided by server and operating system virtualization, along with ease of deployment and low costs, have provided network engineers with new options and capabilities. Nowadays, virtualized computing servers are used in data centers for performing a diverse array of networking functions [62]. To name but a few typical examples, computing servers are widely used for managing the network control plane as virtual routers and virtual switches, especially in the SDN paradigm. The use of such servers in networking projects enables, among else, rapid infrastructure deployment, ease of experimentation and excellent adaptation to shifting load demands (thus, optimizing the cost-performance balance in the infrastructure).

Over the years, specialized network appliances known as *middle-boxes* have become a fundamental part of today's networks. These devices perform a wide array of network or network-related functions: security (firewall, IDS, traffic scrubber), traffic shaping (rate limiter, load balancer), dealing with address space exhaustion (NAT) or improving performance (traffic accelerator, cache, proxy). Despite their usefulness, *middle-boxes* come with a number of problems, many of which arise from the fact that they are hardware-based: they are costly, difficult to manage, and their functionality is hard or impossible to change [63]. To address these issues, there is a new, recent trend towards network function virtualization (NFV). This proposes turning these *middle-boxes* into software-based, virtualized entities running on computing servers, taking advantage of system and OS virtualization. These servers are based on inexpensive, commodity hardware (e.g., off-the-self x86 servers with 10Gb NICs, running Linux-based distributions) virtualized, mainly, by the use of open-source hypervisors, such as XEN [64] and KVM [65]. NFV has already gained a considerable momentum: seven of the world's leading telecoms network operators, along with 52 other

operators, IT and equipment vendors and technology providers, have initiated a new standards group for the virtualization of network functions [66]. Operators want to streamline their infrastructures through NFV and this predicates a consolidation of network functions onto industry-standard servers, switches, and storage hardware located in data centers. For carrier functions to be virtualized, monolithic systems are decommissioned and their functions broken and re-provisioned to components running on virtual machines.

The core networking support in server virtualization and hypervisor environments is based on the IEEE 802.1Q VLAN implementation. Here, virtual network segments are established on top of physical switches – the latter being provided by the server's hardware features (i.e., the hypervisor works as a virtual Ethernet switch and supports queues for each VLAN in the system's memory) [10]. In this way it is possible to establish network communication across different virtual servers, implementing virtual Ethernet adapters without routing network traffic outside the physical system that hosts them (cf. *Figure 4. Hypervisor-based IEEE 802.1Q implementation*). The illustration shows four virtual machines interconnected via two virtual LANs, VLAN2 and VLAN3. All network traffic on these networks is handled by, and flows through, the physical system's hypervisor. The use of this mechanism provides increased VLAN security and much more performance and flexible network deployment compared to physical network devices. These technologies are now offered by most server virtualization products. Examples are VMware [67], IBM [68], and Oracle [69] with the "vSphere", "PowerVM" and "Crossbow/SPARC/Oracle VM" hypervisors, and the XEN hypervisor [64]. Furthermore, a user community has been active on developing operating system code and related software, to be run on such virtual servers, from a networking perspective.

The network's "last-hop switch" has, consequently, been shifted from the pure active network elements to become a characteristic of the hypervisor or of the physical server's hardware [9]. Overall, a paradigm change is in progress in networking by which the *user to IP address ↔ to active network element's network port ↔ to physical location*, no longer applies [70].

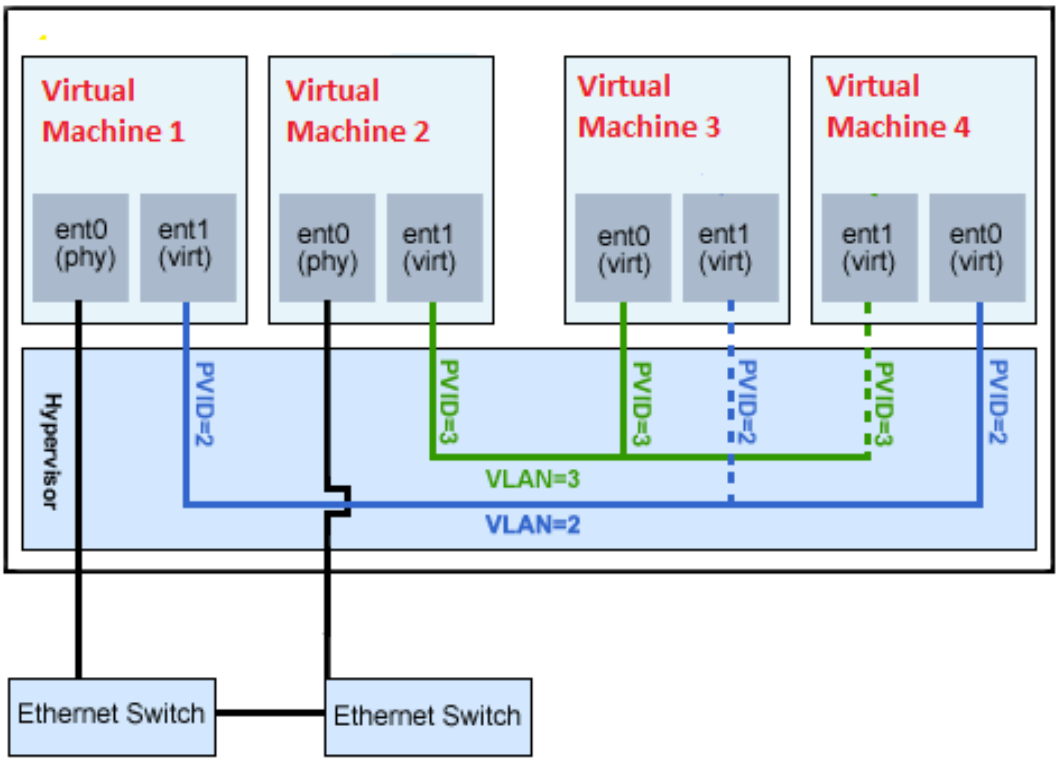


Figure 4. Hypervisor-based IEEE 802.1Q implementation

2.1.1.4 Relevance to our research

The use of the systems virtualization mechanism in network implementations provides increased LAN security and much more flexible network deployment compared to traditional network devices. However, new issues are raised and need to be considered in network design, operation, monitoring and administration: the involved physical server hardware resources (capacity of processors, memory, virtual switch, etc.), the optimization of these resources so as to reach a required system performance and behavior, etc. More complexity is introduced given the new approaches to computing and to IT/networking services delivery [71][72][73]. In the Cloud Computing and in the Networking and Computing Infrastructure as a Service (IaaS) paradigms [11][74], indicatively, the server loads – thus, now, the active network elements – can dynamically shift between physical servers in the same data center or, even, in data centers at different geographical locations.

At the time of writing this dissertation, the application of server virtualization concepts and technologies in the networking landscape is not reflected accordingly in

the available information models for virtual network environments. The hypervisor as an entity, its functional and operational specifics, its internal virtualization layer pertaining to networking (virtual switch, virtual network interfaces), the state of each hypervisor component as well as other relevant aspects are poorly addressed by current models, if at all. This introduces a modeling-related challenge that must be dealt with: the hypervisor and its specifics must be properly modeled. Some of the current models only barely reflect on this matter, as discussed in detail later in this chapter. Only recently has this need been indirectly acknowledged in the modeling and management field, in [75], where the authors address the problem introduced in management operations given the use of different hypervisor types in the infrastructure: it is, very accurately, stated that "*management attributes are the foundation of any management operation*" and "*without current information about the managed objects, sensible and useful management actions cannot be executed*". The authors then provide an analysis of virtual machine attributes and their representation across different hypervisors, in parallel with a possible CIM-based abstraction for each. This analysis, although focusing on the virtual machine side and not attempting to model the hypervisor, provides a clear indication of the complexity and the detail involved in environments where hypervisors are used, should one wish to engage in modeling (as most virtual machine related aspects correspond to a hypervisor-side counterpart resource and attribute). Three examples, two from the industry sector and one from the Open Source community, have evolved around the hypervisor virtualization modeling for the purpose of providing management implementation capabilities to respective products. These products have resulted in CIM Schema extensions focusing on virtual machine monitoring and management. Microsoft Corporation's Hyper-V WMI Provider [76] exposes a CIM-based interface permitting users to monitor and control virtual machines hosted on a Hyper-V server. VMWare's CIM SMASH/Server Management API [77] is the equivalent product for the ESXi hypervisor and allows CIM-compliant management applications to manage the hypervisor and its virtual machines. Finally, Libvirt-CIM [78] is a CIM provider (originating from the Open Source community) for managing Linux virtualization platforms using the *libvirt* library. This product implements the virtualization class model from DMTF CIM's Experimental Schema and focuses on managing the XEN [64] hypervisor and the virtual machines hosted by it.

2.2.1 Model classification schema

It is, therefore, evident that modern VNEs can be very complicated in their design, incorporating a variety of physical and virtual resources (among else servers, hypervisors, routers, nodes, protocols) as well as related services (routing, monitoring, etc.) and member elements (providers, users, SLAs, etc.). Current relevant research presents two distinct groupings among available modeling proposals:

- The state-of-the-art information models, offering elaborate and advanced approaches.
- All other work which incorporates some form of modeling elements. These elements are isolated, autonomous attempts to abstracting specific aspects and information of a network virtualization environment.

Several areas of interest (technical, business, etc.) are tackled in these proposals (i.e., network architecture, communication protocols, business roles and so on). Nevertheless, these modeling approaches do not abstract the VNE as a whole entity.

Qualitative analysis of examined work presents difficulties due to the fact that there is no objective and commonly accepted standard by which comparisons can be made. Our classification schema uses a number of criteria that, intuitively come into mind and are in use in the field, attempting to distinguish items found in literature based on three main areas:

i) The overall positioning of the proposed work.

ii) The modeling aspects.

iii) The system virtualization perspective.

Towards that direction we have defined thirteen distinct criteria, presented below.

2.2.1.1 Overall positioning

Based on a logical categorization we attempt to form a high level grouping, examining the scope and maturity of each proposal. Thus, it is possible to obtain the

following generalized assessment, with the respective possible values listed in the parentheses:

a) **group** (main/MIB/NDL/other): overall categorization of each proposal. This can be main information models-based, MIB-based, NDL-based or other.
b) **resources** (T/O/B): items that can be modeled/abstracted/standardized by the proposed approach. These can be of technical, operational or business nature.
c) **maturity** (high/low): assesses the current status of the proposals with regard to actual applications as well as industry and community adoption.

2.2.1.2 Modeling aspects

As discussed earlier, modern virtual networks can be very complicated in their design, incorporating a variety of physical and virtual resources (servers, hypervisors, routers, nodes, protocols, etc.), as well as related services (routing, monitoring, etc.) and member elements (providers, users, SLAs, etc.). The diverse variety of information does not always fit into a particular type of model. Each proposed model type can serve a particular purpose, though overlapping functionality is apparent across different model types [23][31] and [32]. Furthermore, a certain number of semantics will enable the model to be in a position to span different contexts and allow for handling varying infrastructure items and operational scenarios [31][32]:

d) **model type** (*I/D*): as per the criterion name (informational, data).
e) **representation** (*various*): indicates the representation method used.
f) **views** (*no.of*): support for different perspectives on the abstracted objects.
g) **policy** (*yes/no*): support for establishing conditions and actions.
h) **context** (*yes/no*): support for performing actions based on application-specific information.
i) **capability** (*yes/no*): support for handling and processing different data types and structures.
j) **state** (*yes/no*): support for capturing different states of the managed environment and for enabling the triggering of specific actions based on state information.

2.2.1.3 System virtualization perspective

The use of hypervisors and virtual machines complicates the handling of several aspects of the infrastructure even further [79][41]. Resources provisioned via the system virtualization layer need to be addressed in a system-level environment. The specific characteristics also need to be incorporated in the model's semantics. Thus, the model needs to include design and logic to account for hypervisors and virtual machines:

k) **virtualization** (*yes/no*): support for system (host) based virtualization (virtual machines, virtual CPU/memory, etc.).

l) **hypervisor** (*yes/no/indirectly*): as per the criterion name – if the proposal explicitly defines semantics for hypervisors.

m) **hypervisor agnostic** (*yes or N/A*): whether the model can abstract any hypervisor (i.e., not limited to a specific product).

From a presentation perspective, a direct grouping is possible as four high level categories can be used to span all efforts in the field: *i)* the main information models, with CIM and CIM-based proposals, SID and DEN-ng, *ii)* MIB-based work, *iii)* Network Description Languages, and *iv)* un-generalized approaches (i.e., not belonging to a collective category). This presentation path is adopted in the rest of the Chapter.

2.3 The main information models

2.3.1 CIM and CIM-based approaches

The **Common Information Model** (CIM) [14], proposed by the Distributed Management Task Force (DMTF), is a conceptual, object-oriented, information model for describing the management entities in computing environments. The model is not bound to any particular implementation. Hence, it enables platform-independent and

technology-neutral exchange of management information, providing a consistent definition and structure of data. CIM consists of a specification and a schema:

- The specification describes the model's integration aspects, core architecture and basic interrelationships.
- The schema consists of an extensive set of modeled entities covering areas such as systems, networks, devices, virtualization, applications, metrics, etc.

All components of CIM (the specification, the schema and the schema extensions) are expressed and maintained in the Open Management Group's Unified Modeling Language (UML) [80]. By using the CIM concept it is possible to manage systems through the use of management applications and the interchange of management information among them, through the Common Information Model Object Manager (CIMOM). The latter is an object management engine that exists between the managed system and the management application. The primary example of explicit network-related modeling can be found in the CIM Network model [81]. This characterizes a network as a type of administrative domain, which may contain other networks or sub-networks in a recursive relationship. The model covers both generic aspects required to represent connectivity between systems and relationships to the underlying physical components, as well as network technology and protocol specifics. The CIM Network schema can be extended to model specific network instantiations and architectures. Virtual system configurations can be modeled in several contexts using the CIM System Virtualization and Virtual System profiles [82][83]. These are intended for host systems and discovery of hosted virtual systems used as active network nodes, and for in-depth representation of a virtual system and its components, respectively. Detailed modeling includes hardware and logical device resources (CPU, memory, networking adapters, etc.) and some methods against those entities (basic control operations).

In [20] the authors propose the **VNE-CIM** information model for the formal specification of Virtual Network Environments (VNEs). These are described as a collective of interconnected virtual devices in a certain topology, regardless of the underlying system virtualization platform (Xen, VMware, KVM, etc.) The model includes mapping of VNE creation, provision and administration procedures as well

as several different system virtualization platforms (used as hosts, i.e., virtual nodes, in VNEs). The model has been applied, as an initial proof of concept, on a system setup based on XEN and VMware virtualization platforms, together with the use of open-source libraries for DMTF CIM instrumentation. The current version of the VNE-CIM approach does not model virtual link characteristics (bandwidth, delay, packet loss, etc.) or cross-node VNE deployment based on each node's available resources.

2.3.2 Shared Information/Data (SID) model

The **Shared Information/Data** (SID) model [16] has been proposed by the TeleManagement Forum (TMForum), an international consortium of communication services providers, network operators and relevant suppliers of equipment to the telecommunications domain (hardware and software). Nowadays, the model is part of TMForum's NGOSS Frameworx suite [84]. While SID spawns from the networking world, it mainly covers business entities and processes along with the information flow between them, targeting the enablement of operational support systems and the flawless integration of systems and telecom operation processes. Essentially, SID is a composition of various industry models and presents a common information language for describing management data pertaining to the telecommunications industry.

SID is an information and a data model along with a common business and system vocabulary. It allows coverage of telecommunication business context by means of a high level classification of entities (realized in a framework consisting of different concept areas known as "domains"), together with their attributes and interrelations. SID defines domains at a progressing level of detail, with each domain designed as being self-contained including links to other domains. Class breakdown (the model includes approximately 1000 classes) ranges from general, abstract concepts ("Product", "Service", "Customer", "Resource", "Partner", etc.) to very specific ones. The increasing detail level is represented via the Aggregate Business Entities (ABEs) and their tied information and operations. The implementation of the actual functionality can make use of standardized input and output (e.g., XML files) resulting from the use of the model. Consequently, the data fed to different applications is very well formalized. Cross-area functionality benefits from this

common description of concepts and assets. However, the large number of subclasses used under the root class, as in the case of DMTF CIM, can lead to confusion and possibly complicate the use of the model. Elements and the relationships among them are expressed in UML, the widely adopted standard for building information models. Furthermore, model definitions do include XML schema definition representations (XSD), thus providing the ability for reusable data models. The SID model is data representation agnostic and does not follow or endorse any particular approach (e.g., a database system or class diagrams).

2.3.3 DEN-ng

The Autonomic Communications Forum **DEN-ng** (Directory Enabled Network next generation) information model [17] has its roots in the network management (specifically in policy based network management – PBNM [32]) and in the autonomic networking [85] areas. The focus of PBNM has been on constructing information models for the representation of policy and its specifics. On the other hand, autonomic networking concentrates on creating networks that are self- adjusting so as to adapt to changing needs, based on policy and context. Policy can be conceptualized as a control entity with specific attributes related to the managed environment [86].

DEN-ng is an object oriented information model which provides a common way for the representation of management information (devices, services, users, etc.). It allows for policy and context application to be taken into account. Managed entities are described from different perspectives (views): business, system, implementation and deployment. DEN-ng uses software patterns and roles for modeling managed entities. Information in the model is organized by using a single root class with three subclasses (Entity, Value and MetaData). These form the top-level hierarchies via which general semantics are gradually fine-tuned. They can be made specific by subclassing and by adding more detailed data if necessary. Special classes and associations (Context, PolicyConcept) allow for handling policy [87][88] and context [89] in the detail desired. It is important to note that DEN-ng models any handled entity characteristics, but not the entity itself as a self-contained item (the latter approach is used in CIM model). This, primarily, allows for reusability of the created

components. The DEN-ng model is extensible and, thus, can theoretically cover all aspects of a network and its operational environment, including network virtualization features and technologies. A specific application in a virtual network context was pursued under the Autonomic Internet Project framework (AUTOI) [90][91]. In this project, management overlays were defined for controlling virtualized resources and related services [92].

2.4 Management Information Bases (MIBs)

A **Management Information Base** (MIB) is a logical information store consolidating entity details, organized in a hierarchical (tree-structured) manner. While mostly found in the network management and monitoring context [93][94] MIBs are also used in other areas such as computer systems and high-availability cluster management. Accessing an MIB involves using a specialized protocol, most often the Simple Network Management Protocol (SNMP). The actual properties of the managed objects are populated into the MIB information store by means of specialized software (the MIB module) that normally implements the SNMP protocol. Once the MIB database is populated, management applications can access it and retrieve the device data (cf. *Figure 5. The MIB management concept*). This is performed via issuing requests (SNMP commands) and receiving the relevant responses (SNMP traps) from the managed device. The management application can then parse the responses and take proper actions.

In network management, MIBs provide a tool towards abstracting the operational information, statistics and status of the physical or virtual devices layer from the part of the infrastructure that needs to access it. The resources that can be modeled include physical [95] and logical [96] devices as well as the software that runs on them. Functional details and interrelations, such as user roles, cannot be handled. The outcome is a management information model which describes technical and operational aspects of the network infrastructure. Numerous network virtualization approaches make use of MIBs, including virtual networks and virtual routers, overlays, Multiprotocol Label Switching (MPLS) VPNs, Multi-Virtual Route Forwarding (VRF) and Policy Based Routing (PBR) [97][98][99].

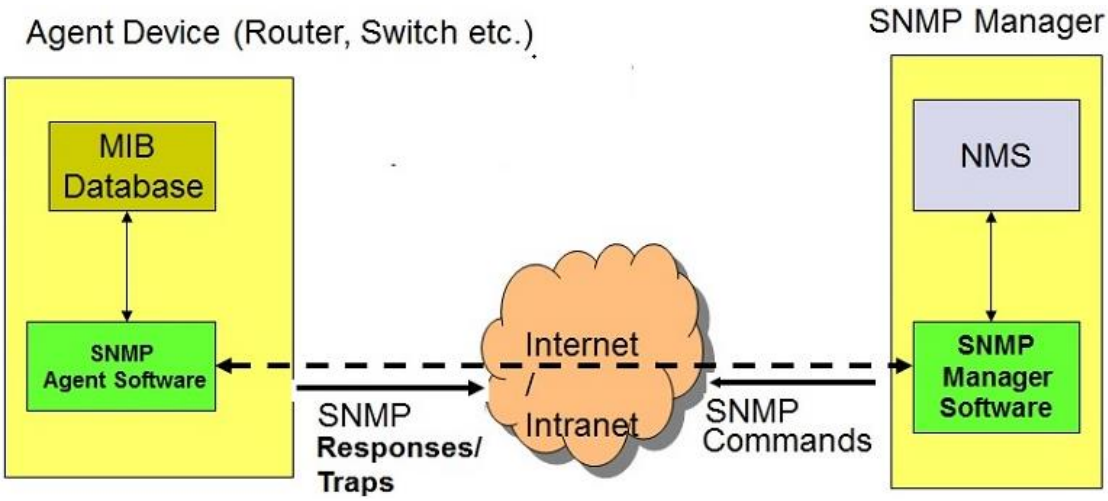


Figure 5. The MIB management concept

If one needs to apply MIBs for managing a virtual network in a system virtualization context, the most prominent approach is the combination of different MIBs. To better illustrate this limitation consider IETF's Entity-MIB [96]. This is a well suited common candidate for hardware infrastructure targeted management as it can handle information between logical and physical entities on one network element, as well as information about the hierarchy among physical entities. However, the detailed architecture of the physical infrastructure cannot be thoroughly represented. To overcome this limitation, another MIB, the Physical Topology MIB [95], can be related to the Entity-MIB and introduced into the model so as to provide the missing semantic coverage. This approach is necessary for a MIB-based model's scope to be broadened.

2.5 Network Description Languages (NDLs)

Description languages, in general, are tools (formal languages) that allow high-level system description and its properties. These languages aid in the planning, systemic analysis and resource specification of the described system. The output of a description language is, most often, XML or XML-based [100] code that can be used as input in certain parsing software. Several such network description languages (NDLs) exist in the computer networks field, primarily focusing on physical, rather than virtual networks [24]. Virtual networks, though, present special attributes (as it

has been discussed in the previous sections) that are, currently, not within the scope of available NDLs. Nevertheless, the latter can be of use in modeling, since as any standardization approach presupposes a formal and structured element description. It is towards this direction that NDLs can be employed using their representational abstractions, like ontologies and logical schemas that support conceptual infrastructure imprinting.

The **Network Description Language** (NDL) [101] is the most prominent among the relevant proposals. It is being maintained by the System and Network Engineer Research Group (SNE) of the University of Amsterdam in the Netherlands. NDL is based on the RDF language - a metadata data model, part of W3C consortium's specification for the theoretical description and modeling of Internet resources [102]. Essentially, NDL is an information model which includes ontologies and schemas (represented in UML) allowing for simple, yet, comprehensive network descriptions. Network information is categorized in network topologies, technology layers, device configurations, capabilities and topology aggregations. The language defines several schemas for the description of different aspects of the network infrastructure, as per the aforementioned categories (topology, layer, capability schemas). A domain schema describes administrative network domains and abstractions for contained devices. The physical network infrastructure is described by the relevant physical schema. NDL has found particular application in the definition of network topologies [103][104] and the relevant processing of those definitions for creating network maps [105] or discovering specific resources (such as paths) [106].

The **Network Markup Language** (NML) has been proposed in [107] by the Open Grid Forum's (OGF) NML Working Group [108][109] and is a description language whose conceptual design consists of flexible and extensible schemas. This allows for the inclusion of network attributes and the construction of new ontologies, as new elements and demands become available. NML focuses on connection-oriented topologies of physical and virtualized networks, especially, on service discovery and provisioning issues in such architectures. The language does not describe aspects such as policy, scheduling and reservation (currently, these types of aspects are outside NML's design scope). The UML language schema describes layer-independent network topologies, together with the properties common to different technologies

employed in the infrastructure [110]. A network description in NML is expressed in XML and RDF. Example uses of the language include optical path finding and resource/topology inventory description.

The **virtual grid Description Language** (vgDL) [111] was developed at Rice University under the VGrADS project and pertains to the Grid Computing area. vgDL provides a framework for the abstractive description of resources, especially related to Grid applications. Resource specification in vgDL consists of a core resource description, along with a ranking function describing the conditions under which the resource is needed. Specific associations indicate resource interdependencies. vgDL example-uses include the description of the attributes and characteristics of workflows in Grid environments [112][113].

The **Virtual Resources and Interconnection Networks Description Language** (VXDL) [114][115] originates from the Grid Computing area (like vgDL). Grid implementations present special operational characteristics: in a Grid environment it is imperative that resources involved can be dynamically re-allocated, without disturbing any running applications. These resources can be of many types: virtual machines acting as nodes, interconnections, network bandwidth, etc. The language allows for the description of the virtual infrastructure resources, the network topology and their specific attributes. It is worth noting that elements of VXDL and NDL were used as base components for the LICL information model [26] of the Geysers project.

The proposal in [116] is the **Infrastructure and Network Description Language** (INDL), a recent effort aiming at describing computing infrastructures in a technology-independent way. The language provides semantic descriptions for physical resources along with their virtualization mechanisms and the underlying networking infrastructure. It can be extended to include other aspects, such as infrastructure federations, resource behavioral aspects, etc. INDL is related to research results and activities from the NML-WG [109], the GEYSERS [117] and NOVI [22] projects. One of the main approaches used for INDL's conceptual schema was the Geysers Information Modeling Framework [26].

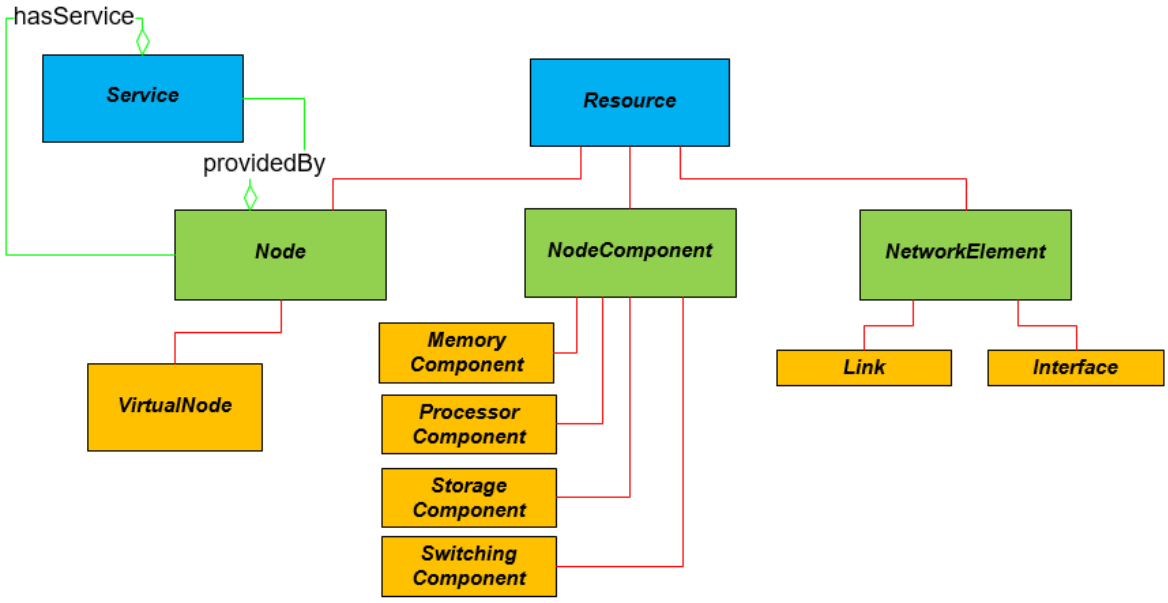


Figure 6. The INDL class hierarchy

INDL is based on an ontology which defines a hierarchy of abstract classes, their associations and properties (cf. *Figure 6. The INDL class hierarchy*). Two main classes (Resource and Service) are used for describing the core components and characteristics of a network setup. Three further sub-classes (Node, NodeComponent, NetworkElement) allow for modeling physical/virtual nodes, their characteristics (CPU, memory, etc.) and network connectivity respectively. Virtualization specifics are modeled via Node's special subclass VirtualNode. The concept of a service is abstracted at the participating node level which is considered as either owner or provider of a service, depending on the view applied on the class.

2.6 Other proposals

In [22] authors propose **NOVI** (Networking innovations Over Virtualized Infrastructures), an information and data model targeted at describing virtual resources in federated, heterogeneous Future Internet platforms. The model facilitates management of such infrastructures by enabling data handling, resources control, provisioning and monitoring. Rich in modeling features NOVI includes semantics and support for resource state handling, context-awareness and managements policies, in a virtualized infrastructure context.

The **LICL** (Logical Infrastructure Composition Layer) information model, developed under the Geysers project, is proposed in [26] and focuses on physical resource to virtual infrastructure provisioning and management. The infrastructure layer is described via a series of physical to logical resource abstractions, incorporating host-based virtualization as end-point network elements. The core aim of the LICL model is to allow decoupling of the infrastructure (specifically, resource management) from service provisioning.

The **Node Architecture** is proposed in [118] (under the 4WARD project [119]) which provides a framework for network virtualization, incorporating design for commercial environments and business scenarios. An RDF-based data model has been developed for the construction of the VNet virtual network, along with its characteristics and available resources. Several networking physical resources can be virtualized in the context of the VNet, such as routers, wired and wireless links. Provisioning, management and control of the VNet has been demonstrated in different scenarios. Finally, a flexible XML-based scheme has been provided for network description. 4WARD, and other Future Internet research projects, are based on the Information-Centric Networking (ICN) concept [120]. The EU-funded SAIL project (Scalable and Adaptive Internet soLutions) [121] adapts to the ICN concept and builds upon the research artifacts, primarily, of 4WARD in order to design architectures, technologies and techniques for adapting current network infrastructures to Future Internet concepts.

Authors of [122] proposed the **Virtual Network Specification Schema** (VN-SLA), an XML-based schema for abstracting a network architecture's business resources and their interrelations. This includes virtual network provisioning scenarios, along with parameters inherent in their design. A top-level VN-SLA class is defined which contains the abstraction and details of a Service Level Agreement (SLA) in a virtual network provision scenario. The focus is on resources provisioned by infrastructure providers (InP) and virtual network providers (VNP). It provides a basis for describing virtual resources and virtualization services at the level required for automated virtual network provisioning.

The **VNMI** proposal presented in [123] and submitted as a working document to the Internet Engineering Task Force introduces an information model for the management of virtual switches. The model describes the physical layer (connections between physical switches) and the virtual layer (connections between virtual switches) in the networking infrastructure. These layers represent the association of the virtual switch with the corresponding physical switch. The approach is focused on device management and, from this perspective, can model physical and virtual resource and hierarchy (mapping) information among members, along with the relevant technical characteristics. Model implementation would result in a MIB-based approach. Consequently, usage of other MIB elements would be necessary in order to extend the model's scope.

Software-Defined Networking (SDN) [27] presents a new, industry popular, paradigm in network architecture by which the network control and data planes are decoupled [124]. The network's intelligence and state are separated from the core components (routers and switches) and are consolidated on control entities (computing servers) [125]. This decoupling into separate *control* and *data* planes (cf. *Figure 7. Logical view of the SDN architecture*) provides the benefit of having different distribution models and actual implementations of the two planes [126][127]. SDN has evolved in close progress with developments in server virtualization and recent trends in computer networks (ossification of the Internet, cloud computing, extensive user mobility, etc.).

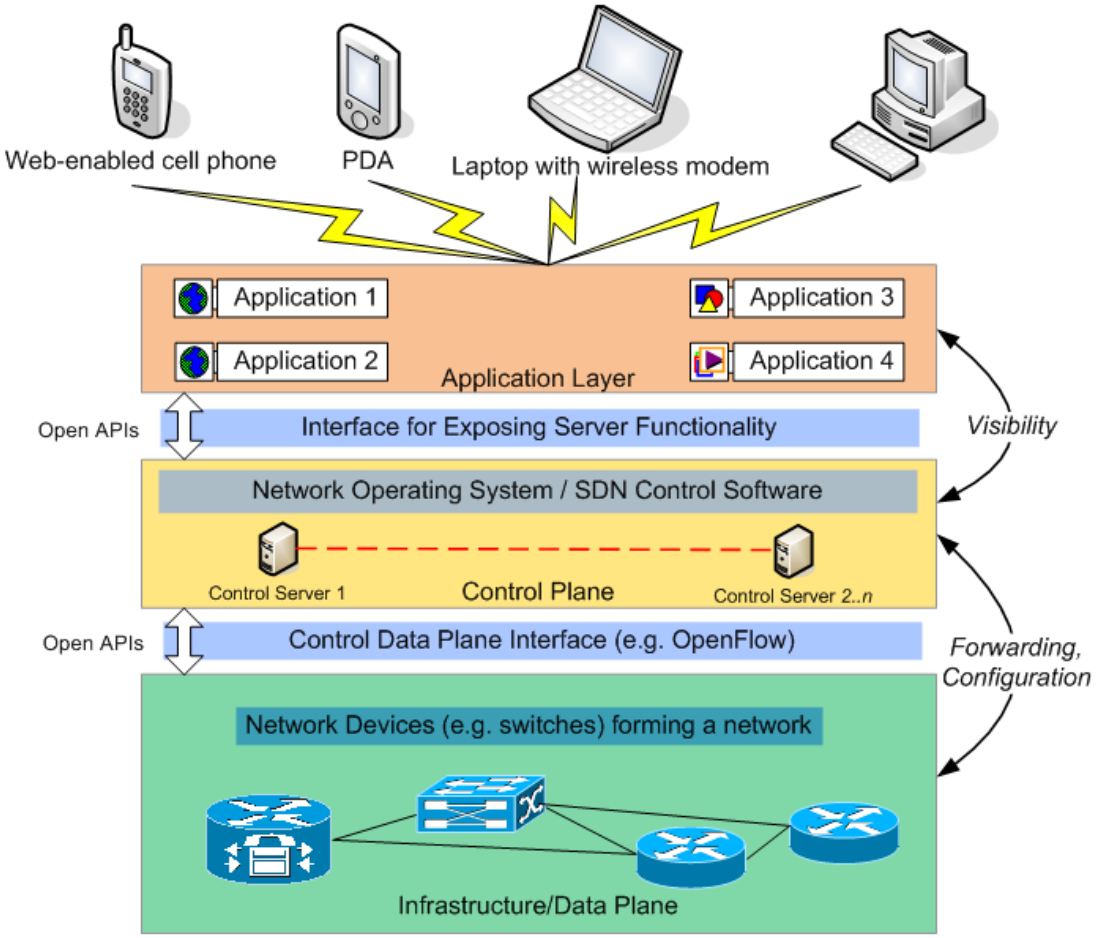


Figure 7. Logical view of the SDN architecture

Configuration and management protocols in the SDN paradigm include OpenFlow [128] and OF-config. The latter is based on NETCONF [129], an XML-based protocol that provides mechanisms to install, manipulate, and delete network device configuration. The underlying data model is based on YANG [130] and describes the resources that can be configured along with the relations among different resource configurations. SDN is somewhat close to the systems hypervisor concept, in that the latter is run on actual hardware to abstract it from the operating system. Hence, SDN has been referred to as the network hypervisor [131][132][60].

2.7 Discussion

A model is a developed representation of a real world system. Different types of models exist that can be used for representing structured and unstructured information, relationships and elements in a given environment [31][30]. Choosing a

proposal to apply in a new project involves at first determining the target use of the model to be created in conjunction with the modeling capabilities offered. Three main factors affect the selection:

i) The kind of resources that need to be modeled.

ii) The requirement for actual implementation in real applications (i.e., the need for instrumentation).

iii) The desired model capabilities for describing the infrastructure in question.

Virtual network environments can be multidimensional entities with respect to the nature, role and function of their elements. Every proposal surveyed in this paper presents different advantages and disadvantages. From a pure modeling perspective it is clear that the three main information models (CIM [14], SID [16], DEN-ng [17]) are superior in modeling concepts and features than those found in other proposals. Nevertheless, these models are complex; they require greater learning effort as well as increased development and maintenance costs, still enjoying broad industry support [133][134]. Given the diverse nature of infrastructure resources and the variety of available modeling approaches, it remains to be determined which approach can be best applied for abstracting this large collective of information, relations, and entities. Ultimately, choosing a model will depend on a number of factors, both modeling-related and others (technical, cost, etc.). An overview of the assessment of these proposals is presented in *Table 1. Summary of current proposals*.

2.7.1 Variety of modeled resources

All proposals do abstract various resources, ranging from technical elements [19], [25], to SLAs [122], relationships and roles [16][118], and business contexts [58]. Some proposals employ a broader scope abstracting a larger array of resources [14], [16][17], whereas others are more limited in that respect [97]. Assessment based on the modeled resources exhibits a higher degree of commonality. A common drawback of all surveyed work is the lack of proper (explicit and detailed) hypervisor [41] abstraction. This is a serious drawback given recent trends where networking converges with computing. Most models treat hypervisors as transparent elements of the virtualization layer. Hence, they begin abstracting from the virtual system or virtual network point. Partial and indirect support can be found in CIM, DEN-ng as

well as in MIBs [25]. In CIM, a hypervisor (not a virtual machine) can be instantiated as a subclass via the OperatingSystem class and the built-in hypervisor virtual switch, respectively, via the UnitaryComputerSystem class. Although CIM (in the System Virtualization Model [83]) elaborates on modeling and management actions on a virtual machine and on the host computer system, it does not account for the hypervisor layer. In a similar fashion, DEN-ng could be extended via subclassing from either the PhysicalResource and LogicalResource or the VirtualSystem and VirtualImage classes. In the MIBs domain the only relevant references are the VM-MIB [43] and the VMM-MIB [135], both at IETF draft status. These MIB objects can store basic hypervisor information (list of guest virtual machines, virtual CPU information and mappings of logical storage and network interfaces). Hypervisor technologies incorporate more operational specifics than what can be abstracted by available models [65][67]. Only recently has the need of modeling the hypervisor been acknowledged, indirectly [75]. Three implementation examples, evolved around hypervisors, have resulted in CIM Schema extensions focusing on virtual machine monitoring and management. Microsoft's Hyper-V WMI Provider [76] exposes a CIM-based interface, permitting users to monitor and control virtual machines hosted on a Hyper-V server. VMWare's CIM SMASH/Server Management API [77] is the equivalent product for the ESXi hypervisor. It allows CIM-compliant management applications to manage the hypervisor and its virtual machines. Finally, Libvirt-CIM [78] is a CIM provider (originating from the Open Source community) for managing Linux virtualization platforms using the libvirt library. This product implements the virtualization class model from DMTF CIM's Experimental Schema. It focuses on managing the XEN [64] hypervisor and the virtual machines hosted by it.

	CIM [14]	**VNE-CIM** [20]	**SID** [16]	**DEN-ng** [17]	**MIB** [25]	**NDL** [101]	**NOVI** [22]	**LICL** [26]	**NODE A.** [118]	**VN-SLA** [122]	**VNMI** [123]	**SDN** [27]
	Overall Positioning											
group	main	main	main	main	MIB	NDL	other	other	other	other	other	other
resources[a]	TO	T	TBO	TB	T	T	TO	TO	TO	B	TO	TO
maturity	high	low	high	high	high	high[b]	low	high	high	low	low	high
	Modeling Aspects											
model type	I	I, D	I, D	I	D	I, D[b]	I, D	I	D	I	I	D
representation	UML, XML	UML, XML	UML	UML	SMI	RDF, XML, UML	RDF, OWL	NDL, RDF, VXDL	UML, XML, GRDF	XML	XML	YANG, XML
views	one	one	two	four	one	one	one	two	one	one	one	one
policy	limited	limited	limited	yes	limited	no	yes	no	no	limited	no	yes
context	no	no	no	yes	no	yes[c]	yes	yes	no	no	no	no
capability	yes	no	no	yes	no	no	yes	no	no	no	no	no
state	limited	limited	no	yes	limited	no	yes	yes	no	no	no	yes
	System Virtualization Perspective											
virtualization concepts	yes	yes	no	yes	yes	yes[c]	yes	yes	no	no	no	no
hypervisor	indirect[d]	no	no	indirect[d]	indirect[d]	no	no	no	no	no	no	no
hypervisor agnostic	yes[d]	N/A	N/A	yes[d]	yes[d]	N/A	N/A	N/A	N/A	N/A	N/A	N/A

[a] **T**: technical, **B**: business, **O**: operational

[b] for NDL, NML

[c] not featured in all NDLs

[d] no explicit notion of a hypervisor

Table 1. Summary of current proposals

2.7.2 The need for actual implementations

If there is a need for the target model to be applied in real world applications, it is clear that its instrumentation must be supported on the actual hardware and software environment where it will be applied. Therefore, selecting one of the proposed approaches, based on instrumentation possibility, requires a mature model with collateral device support, while simultaneously using well adopted and accepted representation methods. Here the choices are clearer: the broadest industry support is enjoyed by CIM, SID, SDN, DEN-ng and MIBs, in the sense that there already exist management applications, employing the aforementioned models, and expandable to suit custom infrastructures. Likewise, particular NDLs have been widely adopted and applied in various environments [101][107]. By elaborating on the maturity of each proposed model, we can see that well-established methods can increase adoption possibility. Restrictions still apply: CIM-based solutions are bound to the availability of specific operating system providers that materialize the instrumentation of a model element. If these providers are not available they need to be developed, imposing an overhead in model adoption and solution provisioning. In a similar fashion, MIBs are implementation-dependent and any solution relies on suitable MIB modules availability, as discussed in previous Sections. Finally, SDN is enjoying considerable industry support, making it a positive choice, implementation-wise.

2.7.3 Modeling perspective of proposals

From a modeling perspective, the main models have been designed to cover a broad scope. Hence, they incorporate greater detail and more modeling mechanisms than the rest of the proposals. SID partially originates from an older release of DEN-ng, with the two models sharing, up to a certain extent, common concepts in model creation (relationships, attributes, etc.). Both models differ from CIM in that different modeling approaches (such as meta-models) are used. SID and DEN-ng are pure information models whereas CIM is not, since it is not entirely technology agnostic [32]. CIM provides for information abstraction that can be extended to include new items of the infrastructure and can be adapted to changes in management protocols. The model, however, does not include semantics for business processes and logic as the other two main models do. On the other hand, MIBs are technology-dependent data models that represent virtual containers for managed objects and their

information. MIBs are too focused [136] and limited in scope, most often abstracting the specifics of a particular device or protocol. The three main models are object-oriented in design, whereas MIBs are hierarchical tree views of the managed objects. This implies that extending SID, DEN-ng and CIM presupposes a clear understanding of parts of the model, their class inheritance and associations. Extensions in MIBs are realized by adding sub-trees to the hierarchical structure. This is a simpler process, partly accounting for MIBs and SNMP popularity.

Representation methods vary in the proposals: most use UML (Unified Modeling Language) and XML (or clones), both mature and widely accepted standards [80], [100]. MIBs employ IETF's SMI standard (Structure of Management Information Version 2) [93], a subset of the ASN.1 standard (Abstract Syntax Notation One) [137]. SMI includes module, object and notification definitions for describing information semantics, managed objects' description, and management information transmission, respectively. SMI is inherently limited in semantic representation and cannot support complex data structures; hence, MIBs are bound by this restriction. This is hindered even more as MIBs use vendor-specific data (by sub-tree addition for SNMP extensions). The result is less standardization, even though SMI has been defined as a standard. On the other hand, SID and DEN-ng use UML. UML is far richer in semantic capabilities and techniques than SMI and can be used to construct and support complex models. CIM, although expressed and maintained in UML (leveraging its advantages), does not result in fully compliant UML models. Information and concept expression in UML leverages the advantages of XML representation, increasing the probability of being used by a wider array of management (or other) applications. The NOVI [22], LICL [26] and SDN [27] models are exceptions to the general rule and use RDF/Owl, NDL/VXDL and YANG, respectively.

SID and DEN-ng provide more modeling features, such as patterns and roles. Patterns provide for design reuse and roles provide abstractions for managed elements based on functions that they can perform. Several elements can be abstracted via roles, ranging from people to infrastructure item, such as devices. CIM does not support patterns or roles [92]. Another important quality characteristic among the three main models is their support for state (defined as the condition of any managed object at

any given instance), context and policy. DEN-ng outweighs CIM [122][92] and SID in these features. In terms of state, most information models are referred to as current state models; that is, they abstract and capture a managed resource state at a particular point in time and do not incorporate mechanisms for handling state variations. Via the CIMState extension, together with specific properties and their enumerators, CIM incorporates the current operational state of a managed resource. A common state model of CIM objects is provided via predefined states, where state pertains to managed object's attributes that can be queried or measured. DEN-ng includes advanced semantics and mechanisms [32][92]. It can also provide for context- and policy-aware systems with varying object state, depending on changes in context and policy. SID does not provide advanced support context or policy. A detailed discussion of these features and limitations can be found in [86][87][89]. Compared to the other two main information models (DMTF CIM and TMF SID), DEN-ng is superior in the sense that it is truly implementation-independent and does include extensive features for policy, state and context representation. Additionally, DEN-ng provides a well-designed metadata model, whereas the other two models lack that feature. The minimal approach in class structure enables clearer design and better understanding of the model (in contrast to CIM and SID where thousands of classes are used). The NOVI information model [22] is also rich in features as it supports policy, context, capability and state. LICL [26], too, provides context-aware decisions and state capturing of infrastructure resources.

The rest of the surveyed approaches are limited from a modeling perspective, which is to be expected, as these models focus on a specific research aspect and have not been designed for a broader information abstraction scope. Information models are proposed in [122][123] for the management of SLA-based provisioning scenarios and virtual switches, respectively. An information model, based on CIM, that abstracts system and network components found in a virtual network environment is proposed in [20]. Virtual machines and hypervisors are in the scope of work proposed in [19]. The semantics in these models allow for the description of different parts of the network virtualization environment; each model applies a different view on the modeled environment. None of these proposals, however, includes appropriate support for policy, context and capability; moreover, view support is limited and available only in certain cases. Finally, network description languages are mainly

network-oriented [24]. Consequently, they do not provide semantics and mechanism for describing computing architectures. This has changed recently [116] with the proposal of INDL, which provides support for basic computing infrastructure.

2.8 Chapter summary

In this Chapter, we introduced the topic of modeling in the Computing and Networking domains. We presented in detail the characteristics of current proposals, thus establishing the state-of-the-art at the time of writing of this dissertation. We also provided a classification schema by which current proposals can be organized and qualitatively assessed. All the aforementioned information is important when designing and building a new information model for virtual network environments in Cloud datacenters. Lastly, we provided a detailed discussion on the suitability of the proposals presented for any new application undertaking.

Chapter 3 – The proposed information model

3.1 Introduction

A holistic model for VNEs in a hypervisor-based environment has not, yet, been proposed. Current research proposals (cf. Chapter 2) incorporate modeling elements to a greater or lesser extent. The creation of such a model involves overcoming certain difficulties, as discussed in the later sections of this Chapter, and taking informed decisions on different aspects pertaining to this undertaking. It is reasonable to assume that existing proposals will be reused, forming either the basis for model creation, or for extending/enhancing design and features of a newly created model. Such a reuse would incur obvious benefits (knowledge and skills transfer, lower initial costs, etc.) compared to the overwhelming effort needed for a large scale information model to be built from scratch. It is important to note at this point that the three main information models available today (CIM [14], SID [16] and DEN-ng [17]) have already been available to stakeholders, and have remained under constant enhancement for several years. Their development has been the outcome of the joint effort of several companies and organizations, with continuous funding and competent project management.

The aforementioned information models claim holistic nature as they do have a broad scope and cover a wide variety of elements. This claim, however, is not accurate as these models have been built taking into consideration the most important data sources in the vertical market in whose environment they have been developed. SID, for example, has been constructed within the telecommunication context, with business processes as its primary focus. Likewise, CIM targets IT infrastructure from an operations perspective. It is, therefore, debatable whether a single information model covering all possible context scenarios and data sources can be created. This model will, obviously, be very expensive to build and the required development effort and cost might not be justified by the anticipated advantages.

This dissertation proposes a new information model that attempts to cover the lack of hypervisor support in other proposals, while in the virtualized networking landscape. The proposed model, essentially a CIM schema extension, is based on CIM, thus

inheriting the latter's design features and advantages. The overall positioning, in the infrastructure modeling landscape, of the work presented in this dissertation is illustrated in *Figure 8. Positioning of the proposed information model*.

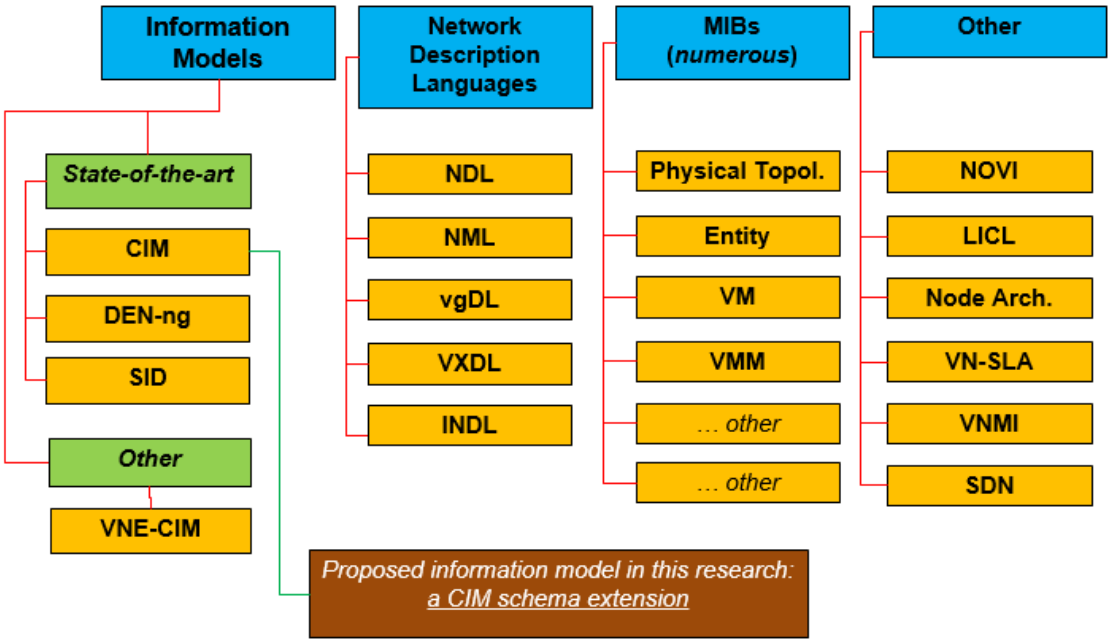


Figure 8. Positioning of the proposed information model

The remainder of the chapter is organized as follows. Section 3.2 introduces the requirements for the new model. Section 3.3 outlines our proposal's relevance to the (parent) CIM model. Section 3.4 presents the new model's architecture and details its characteristics. Finally, Section 3.5 concludes the chapter.

3.2 Relevance to other proposals

As stated earlier, a model is a developed representation of a real world system. Different models exist that can be used for representing the structured and unstructured information, relationships and elements in a given virtual network architecture. This diverse variety of information does not always fit into a particular type of model. In many real life cases it may be necessary to employ or combine different approaches to reach some suitable, per case, representation. The proposals discussed in Chapter 2 cannot efficiently abstract a virtual network infrastructure based on hypervisor-provisioned resources. This is due to those proposals' lack of

explicit design for hypervisor entities and their characteristics. In the VNE context, the creation of a suitable model needs to address in detail the virtualization layer and the concept of hypervisors, in particular. The latter has been overlooked in all available models and no proper abstraction has been proposed. Sporadic support can be found in some proposals but is limited in scope compared to the complexity and details involved in managing a hypervisor. The model design must ensure that any specific hypervisor platform is supported (vendor-independence and implementation-agnostic), providing the ability to abstract and manage virtualized network connectivity both within a single hypervisor, as well as in cross-hypervisor architectures (where more than one type of hypervisors are used in the VNE). This feature will also enable easier model introduction in VNEs in the Cloud Computing context, where shifting workloads among nodes (i.e., rerouting traffic across hypervisors) is an essential characteristic [11]. Consequently, a clear and parallel challenge is evident: direct hypervisor modeling.

Combination of different surveyed approaches, aggregated in a new model, is neither possible nor desirable for several reasons. Current models differ in their nature (others are informational models, others are data models, etc.) and incorporate different approaches and abstractions for similar items of the infrastructure. This causes cross model incompatibility issues. Furthermore, it is not always an easy (or even possible) task to extend current models in order to allow for new infrastructure items. This hinders their use as a basis for a VNE holistic model. On the pure modeling side, each surveyed approach presents both advantages and disadvantages (discussed in detail in section in 2.7 of Chapter 2). These should also be evaluated in any decision for partial or whole approach adoption. A similar conclusion is presented in [23] on the applicability of the main information models in the context of the federation of Future Internet platforms. Overall, both the feasibility of building a single, context-wise all-inclusive, information model, as well as the possible adoption of existing models (or parts of) and their extension for use in new context areas, constitute open research questions.

In this dissertation, in order to account for semantic representation of such resources, we propose the KF model, a CIM-based conceptual representation of the different components that constitute a virtual machine-based network. Since our proposal is

based on CIM, it enjoys the benefits offered by the parent model. The KF model can cover physical and logical components supporting the virtual network, along with its settings, modes of operation and statistical elements of the hypervisors and virtual machines involved. The model is extensible so as to include new elements that need to be introduced, in a hardware-agnostic way. As a result, it can be applied to a wide variety of scenarios and does not depend on any particular hardware implementation. On the design side, at the logical level, the model semantically incorporates a virtual network spanning a number of virtual server hosts (which act as active network elements and provide its core resources), along with the virtualization techniques (physical nodes, hypervisors, virtual machines – VMs) [65] employed in such a design. System provisioned resources (such as CPU, memory and I/O), as well as other relevant operational parameters are included in the model. Given the agnostic nature of the model various virtualization platforms are supported, as long as proper providers are developed adhering to the CIM approach.

The CIM virtualization model, from which our proposal inherits, addresses the concepts of resource allocation, system virtualization and virtual devices. This is achieved by extending the existing CIM system modeling, and by reusing system and logical device classes to model virtual systems. Our proposal enhances on that and fills the current gap by directly addressing the hypervisor, its functional and operational specifics and the internal virtualization layer pertaining to networking (i.e., virtual switch, virtual network interfaces). As described in previous sections, these aspects are, poorly, if at all, addressed by other information models. In the industry sector and in the Open Source community, three software solutions have been made available for the purpose of providing specific management implementation options to respective management products. These efforts have not resulted in generic hypervisor models rather than on limited CIM Schema extensions, focusing on virtual machine monitoring and management for each vendor's hypervisor. Microsoft Corporation's Hyper-V WMI Provider [76] exposes a CIM-based interface, permitting users to monitor and control virtual machines hosted on a Hyper-V server. VMWare's CIM SMASH/Server Management API [77] is the equivalent product for the ESXi hypervisor and allows CIM-compliant management applications to manage the hypervisor and its virtual machines. Finally, Libvirt-CIM [78] is a CIM provider for managing Linux virtualization platforms using the libvirt

library. This product implements the virtualization class model from DMTF CIM's Experimental Schema and focuses on managing the XEN [64] hypervisor and the virtual machines hosted by it.

3.3 The proposed model

3.3.1 Model requirements

A VNE holistic information model, as approached in this dissertation, refers to a, context-wise independent, complete information model of a virtual network environment hosted in a hypervisor-based virtualized architecture. It is a construct that can represent all aspects of a VNE, whether of technical, business, operational or other nature. This common model is sufficient to theoretically abstract most, if not all, aspects in the VNE: technical infrastructure (hardware, software, connections, operations), members and their roles (users, providers, management entities), business context of application, data manipulated by all parties, etc. (cf. *Figure 9. The VNE holistic model constituents*). Given recent developments in networking (discussed in Chapter 1), where hypervisors are now used as network active elements, the common model must be able to abstract hypervisor-specific information and state. Due to the fact that network implementations employ technologies and products from different vendors it is crucial that the model is vendor-independent. The neutrality and extensibility characteristics allow the model to be adaptive and used in current and future, multi-vendor, VNE environments.

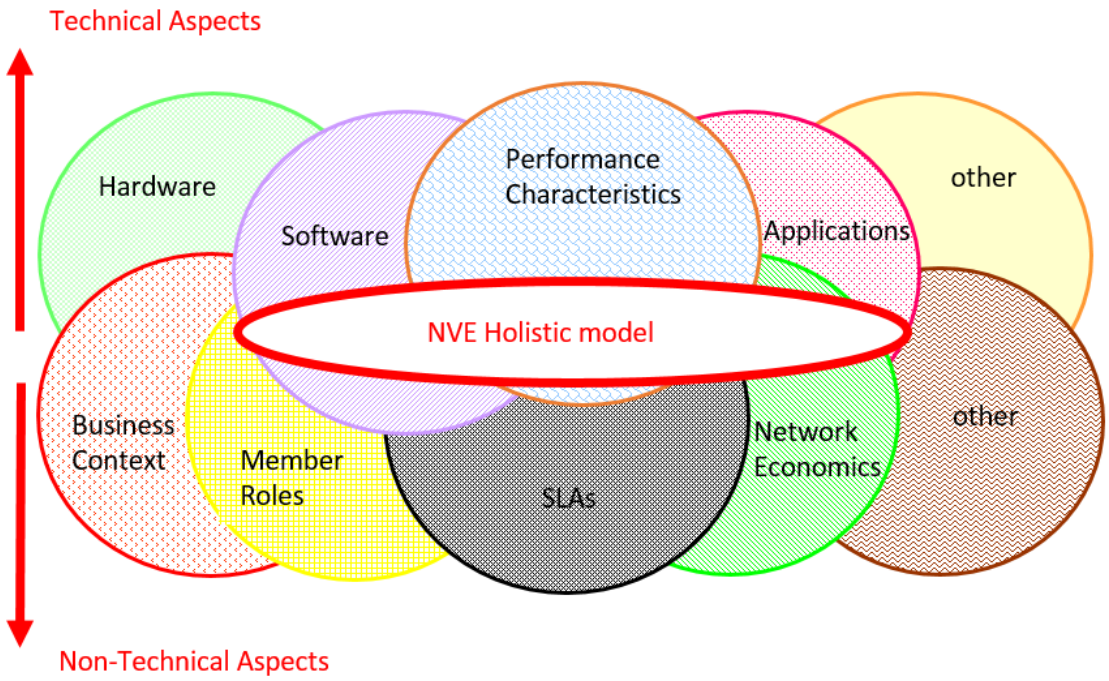


Figure 9. The VNE holistic model constituents

It has been suggested in [23] that an information model, in the virtual infrastructures federation context, will need to support:

i) Virtualization concepts.

ii) Vendor independence.

iii) Monitoring and measurement concepts.

iv) Management policies.

v) Context-aware decisions.

vi) Description of stateful resources.

We acknowledge the need for those features in a holistic VNE model and (along with the adoption of hypervisors) we augment the argument, stating that the holistic model must incorporate extensive semantic support for virtualization across all suggested concepts. By that, we refer to definitive and explicit handling of hypervisors and their specific information, attributes and operational characteristics, state and parameters [41]. The use of hypervisors complicates even further the handling of several aspects of the infrastructure. Every resource provisioned via the hypervisor needs to be addressed in a resource competition environment and the overhead imposed by the virtualization layer has to be incorporated in the model's semantics. Thus, the model needs to account for new logic and metrics [79] introduced to account for hypervisors.

A VNE implementation, whether entry level or full scale, includes several constituents, ranging from infrastructure (hardware, software) to services and user roles. Each constituent part accesses some resource and, in certain scenarios, can be the owner of the resource. Furthermore, the same resources may be used via different methods and interfaces (e.g., accessing a resource via the virtualization's management interface is different than via the external client interface). The holistic model must address these issues and provide clear abstractions and methods to account for multiple resource ownership and varying access interfaces of the same resource. The model must include policy support pertaining to authorization (who is allowed to access the resource), conditions (based on rules/situations allowing resource access) and access methods (how the resource can be accessed). Obviously, policy support must span the hypervisor layer and the virtual devices provided by it (e.g., virtual switch) [41]. Semantics must also be provided to support state information for all VNE resources (including the hypervisor itself and the physical system it operates on) so as to allow for decision making based on the managed environment in which it is applied. This feature will enable management applications to react based on a dynamically changing infrastructure environment (e.g., in the Cloud computing area) [139].

The basic elements of a system virtualization environment are shown in *Figure 10. Basic elements of a system virtualization environment*. Resources that make up the virtualization environment typically are supplied by one or more host computer systems (physical servers), storage devices and networking devices (the latter, also, being hypervisor-provisioned where needed). A virtual infrastructure layer manages the lifecycle of virtual computer systems, which are composed of resources allocated from the host computer systems. A virtual computer system may be active and running an operating system and applications with a full complement of virtual devices defined and allocated, or it may be inactive with no software running and a subset of the virtual devices actually allocated. The proposed model enables the client to manage the virtualization layer and the full lifecycle of the virtual computer systems hosted.

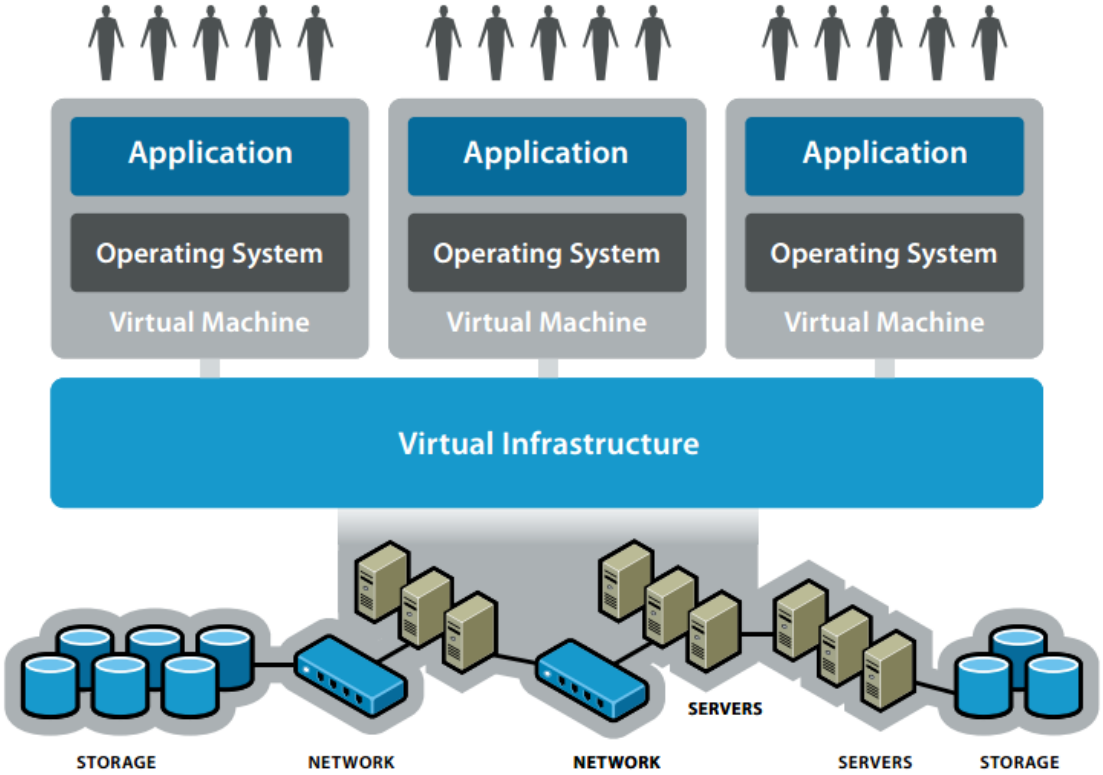


Figure 10. Basic elements of a system virtualization environment

3.3.2 Model architecture

The KF model consists of ten classes (cf. *Appendix B for UML schemas*) representing the hypervisor, virtual machine, network, configuration and operation parts of the virtual network architecture[3] (cf. *Table 2. KF model classes*). These classes, along with the extensibility characteristics of the model, are adequate for representing the basic design of any virtual network employing virtual machines hosted on a hypervisor (regardless of the choice of hypervisor). Additional features and facilities can be abstracted by extending the model, thus eliminating the need for initial classes over commitment which would incur difficulties in the model design.

An exclusive namespace has been applied to class naming, under which each class name is prefixed with a "KF_" convention. In the current version of the model the following elements of a virtual host based network are referenced:

- Computing (systems) nodes.

[3] Detailed UML diagrams, complete textual class descriptions as well as MOF files are available at http://users.uom.gr/~kontoudis/research/

- Hypervisors.
- Hypervisor virtual switches.
- Virtual machines.
- Processes.
- Applications.
- Virtual networks along with their settings and statistics.

All these elements are semantically represented at the logical level. The virtual network environment, as a complete entity, is conceived as a collection of the referenced entities. This design approach simplifies handling and consolidates global characteristics, such as settings, statistics, naming, etc. The virtual machine part uses six classes, handling physical server infrastructure as a hardware node with a running hypervisor, a number of participating VMs, and the processes and applications running on these VMs. Intra-node networking is, in part, represented by a specific "KF_HypervisorVirtualSwitch" class which details the hypervisor's in-memory IEEE802.1Q VLAN compatible virtual switch. Moreover, networking information is shared with the "KF_Network" class; the latter includes the properties needed for mapping a virtual host based network notion [81] as a whole entity. A special class is used for handling statistical data, resulting in a total of three classes abstracting networking information. A number of associations have been designed which, being double-ended references, return specific operational data depending on the invocation method. These associations report the amount of virtual Ethernet adapters allocated per VM, the physical node's running hypervisor, the applications operating per VM and per virtual network, etc. Thus, typical Cloud datacenter virtualized environments, such as single-node and multi-node IEEE802.1Q virtual Ethernet switch based architectures (cf. *Figure 11. Virtual Ethernet Switch based architecture*), can be easily described using the proposed model. In such architectures, as discussed in previous sections, some network traffic (VLAN1 and VLAN2 in the illustration) is routed through the hypervisor's IEEE802.1Q virtual Ethernet switch and never leaves the physical host. This yields certain advantages such as increased performance and security given that the network traffic is handled *in-memory* of the host computer system.

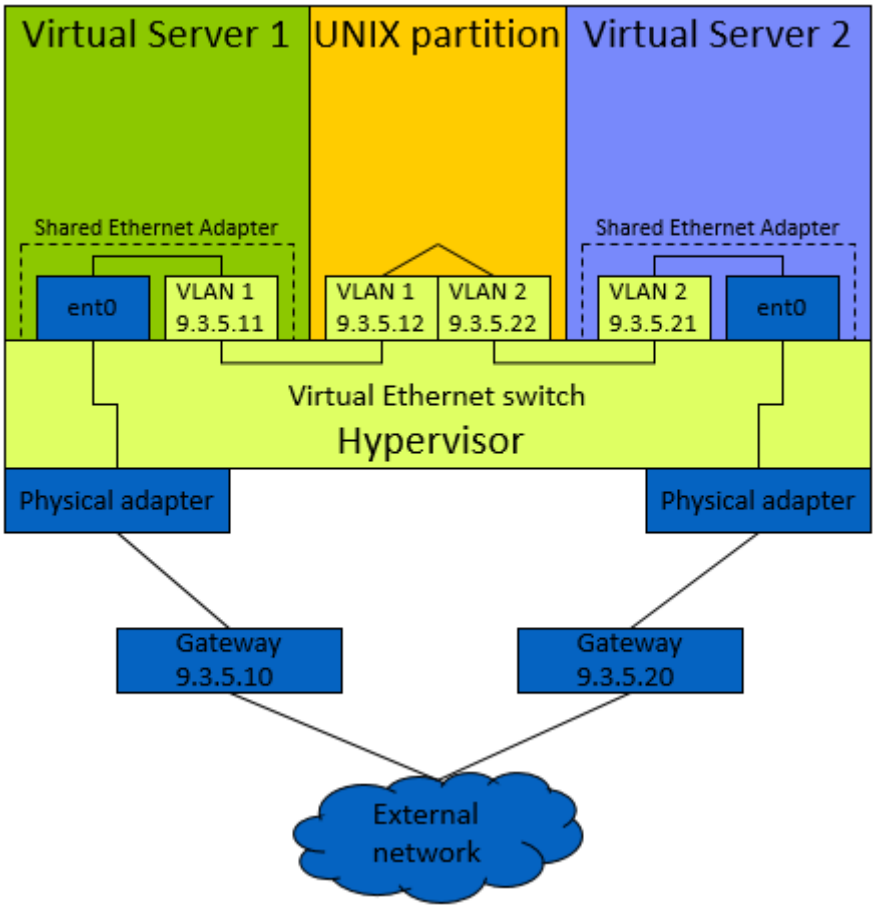


Figure 11. Virtual Ethernet Switch based architecture

3.3.3 Modeling abstractions

The management modeling abstractions designed in the proposed model evolve around the typical managed environment of host-based VNEs, as found in Cloud datacenters (cf. *Figure 10. Basic elements of a system virtualization environment* and *Figure 11. Virtual Ethernet Switch based architecture*). The abstractions incorporate the attributes of the most important managed elements, resources, characteristics and relations in those environments. An overview of the abstractions for the basic entities is outlined in *Table 2. KF model classes* and, for the resource allocation related methods of Chapter 4 in *Table 5. KF_VM class SPC methods*, respectively. Since the model is based on CIM, it is built on object-oriented design philosophy. It is made up of classes, which reference each other by inheritance and association. If a class inherits another, it includes all properties of the parent class. A reference means that

the data type of a property of a class is an instance of another. An instance of a class is usually called an *object*.

KF class	Purpose	Properties
KF_Node	Describes a physical node (computing server) participating in a VNE	SupportedHypervisors: string RunningHypervisor: string
KF_Hypervisor	Describes the physical node's hypervisor [4] (Type 1 or Type 2) [33]	HypervisorName: string HypervisorDescription: string SupportedOSs: string NumberofRunningVMs: uint16
KF_ HypervisorVirtualSwitch	Describes the hypervisor's in-memory IEEE802.1Q VLAN compatible virtual switch	NumberofVirtualPorts: uint16 VirtualPortSpeed: uint32 VirtualEthernetFrameSize:uint64 NumberOfSupportedNetworks: uint32
KF_VM	Describes virtual servers running on top of the hypervisor, along with a UNIX-like operating system	NodeId: uint16 RunningHypervisor: string NumberofServicedApplications: uint16 AllocatedVirtualProcessors: uint16 AllocatedEthernetBandwidth:unit64 VirtualEthernetAdapterType: uint8
KF_Process	A Process running in the virtual server	NodeId: unit16 BelongsToApplication: string VirtualProcessorUtil: uint32 ComputationalMemoty:uint64 EthernetAdapterUtil: uint32
KF_Application	An application composed of 1..N processes	NodeId: unit16 ProcessList: uint32 VirtualProcessorsUsed: uint16 MemoryUsed: uint64 AggregatedEthernetIO:uint64
KF_Network	Describes a virtual network	ParticipatingNodes: string Participating VMs: string ServicedApplications: string

[4] Examples of Type 1 hypervisors include IBM's {Power Hypervisor/PHYP, Processor Resource/System Manager, z/VM}, VMWARE's ESX server, Microsoft's HYPER-V, xen.org's XEN and KVM (found in several Linux distributions), whereas Type 2 hypervisors include Microsoft's Virtual Server and VMware's {GSX server, Workstation}

KF_NVECollection	Refers to all elements comprising an VNE	*Collection Name inherited from parent class*
KF_NVEStatisticalData	Holds statistical elements about an VNE	HypervisorCPUres: uint32 HypervisorMEMres: uint64 HypervisorIOres: uint64
KF_Settings	Consolidates VNE-wide settings	NVEName: string NVEDescription: string NVEid:uint16 MemberNodes: unit32 RunningApplications: unit32

Table 2. KF model classes

3.3.3.1 Resources

The basic element in model design is the abstraction of a managed environment's core resource and the resulting definition in UML. Resources are the basic units of abstraction. As described in previous sections, a VNE is composed of a set of resources alongside their attributes and interrelations. Efficient operations on those resources require their configuration and state information to be readily available to management applications. In theory, the target environment may be composed of an infinite number of those resources. The classes included in the model represent the core resources provisioned in VNEs in question. It is possible that many resource occurrences represent instantiations due to the same resource's different states, e.g., in configuration. For example, consider a virtual machine with a CPU allocation of one core at a specific time point. After some time, the CPU allocation may have changed to two cores, due to dynamic reconfiguration. This variation is reflected as a different resource *state*.

3.3.3.2 CIM inheritance

The management environment structure cannot be sufficiently handled by static definitions in the KF model. The proposed CIM extension schema leverages the functionality of certain CIM classes, reusing attributes and methods. These are inherited and reused by child classes from parent classes (cf.
Table 3. CIM class inheritance).

KF class	CIM parent class
KF_Node	UnitaryComputerSystem
KF_Hypervisor	OperatingSystem
KF_HypervisorVirtualSwitch	UnitaryComputerSystem
KF_VM	OperatingSystem
KF_Process	UnixProcess
KF_Application	SoftwareFeature
KF_Network	Network
KF_NVECollection	CollectionOfMSEs
KF_NVEStatisticalData	SystemStatisticalInformation
KF_Settings	SettingData

Table 3. CIM class inheritance

Leveraging the CIM inheritance provides for the following, general requirements:

- Enables clients that are unaware of virtualization to manage virtual systems. That is, after a virtual computer system is created, most management operations (such as list, install, configure, show devices) are available similarly on virtual or physical systems.
- The model is flexible and general enough to support all types of platform virtualization including hypervisor-based virtualization, logical and physical partitioning, and operating system containers. The general patterns developed to model resource virtualization are applicable as new types of virtualization become available.
- Because the capabilities of system virtualization implementations vary widely, the model supports the runtime inspection of a system's capabilities. In this way a client does not need to know a priori about an implementation's capabilities for the system to be managed effectively. This includes the ability to determine supported resource types, resources, and lifecycle capabilities.
- Management operations are modeled such that reasonable defaults are made available wherever possible.
- The model is extensible, with clear mechanisms for adding implementation-specific capabilities and for allowing a client to discover these capabilities.

3.3.3.3 Dependencies, associations and aggregations

Resources can have varying number of bindings defined between them; a level of complexity necessary to reflect actual managed environments where architecture components are, very rarely, used as isolated entities. Because of this very fact, bindings cannot be ignored. A simple example would be that of an operating system (described by the KF_VM class). This presupposes the existence of a virtualized computer system hosting a hypervisor on which it runs (cf. *Figure 12. Bindings example*). This high-level UML illustration shows the KF_Hypervisor and KF_VM classes along with their intra relationships. An operating system without an associated host providing the hardware base for the operation can never exist in a real environment.

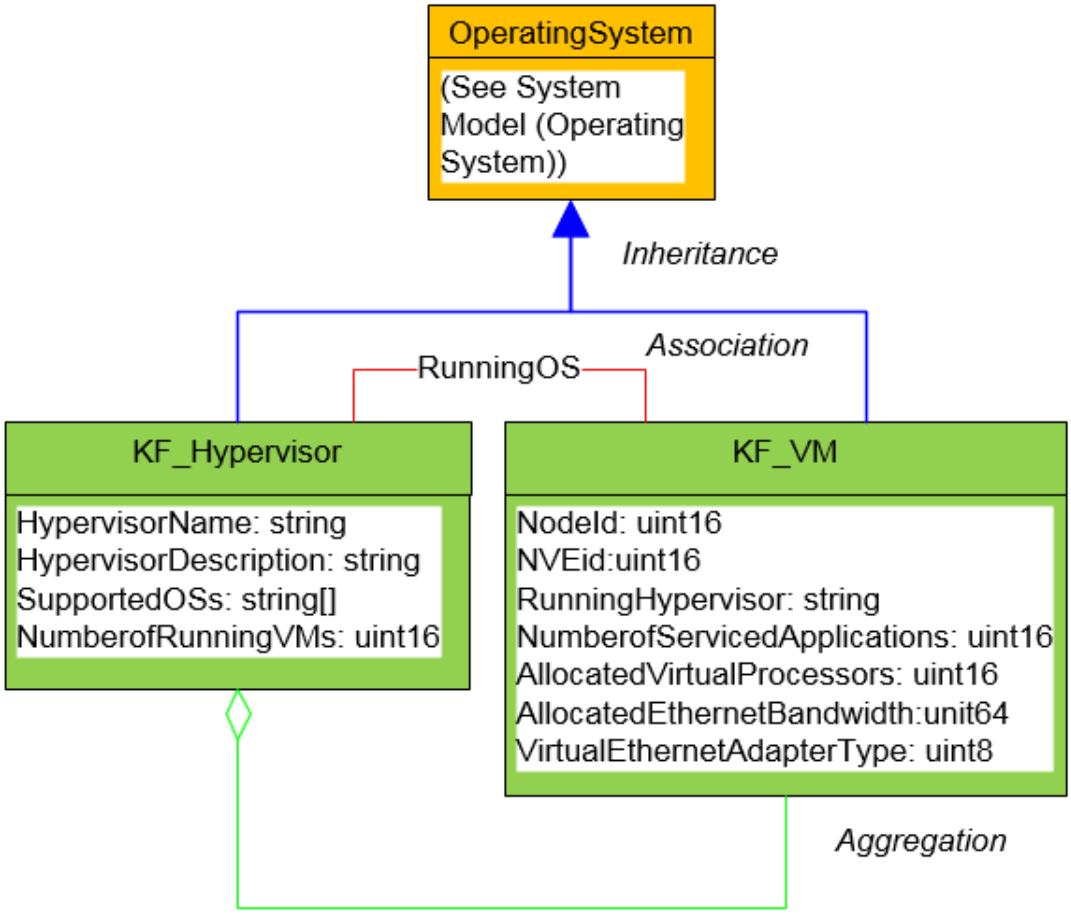


Figure 12. Bindings example

3.3.4 Model extensibility

The KF model design allows for the inclusion of any manageable entity by implementing proper extensions which can augment the model's scope and, thus, the managing application's (that makes use of the model) functionality. For example, suppose that a need arises for handling transaction-based performance characteristics or for the management of a virtual router instantiated by a virtual server. The former need can be tackled by a CIM_UnitOfWork derivative subclass [140], whereas the latter via extending the KF_VM class to include the required management methods. This extensibility of the KF model derives from its design logic and allows for the wieldy inclusion of new features and elements. CIM schemas are expressed in UML and their syntax description (textual definition of the class name and attributes) is composed in the Managed Object Format (MOF) [141].

3.3.5 Modeling language

The modeling language adopted for defining the modeling elements of the proposed model is the Unified Modeling Language (UML) [80]. UML class diagrams constitute powerful modeling concepts, not only for the information artifacts, but for also characterizing their relationships via the abstractions of composition, association and inheritance. The object-oriented metamodel adopted is expressed using the Meta Object Facility (MOF) [141]. This language is designed to define metamodels belonging to the structure of information models. It allows for any level of complexity to be expressed in a concise and structured manner. Both UML and MOF are the core languages adopted by CIM, thus their potential and suitability have been extensively verified.

The enabling technology for the data exchange across the different model components is the eXtensible Markup Language, XML [100] (a W3C standard [102]). This is a widely adopted technology which allows for a common, standardized, way of information expression. It is intended as a structured data definition language, easily interpreted by a computer system, while remaining human-readable. The XML standard itself defines markup rules, but the content and semantics are up to the user to decide. Accompanying standards define further rules for more specific use cases.

Being computer interpreted and having multiple implementations for the core parts of the standard, XML can easily be used in an application even without any knowledge of the accompanying standards. XSD is the preferred file extension of XML schemas, which is an XML-based markup for defining the allowed elements in an XML file. This information is primarily used for validating the contents of the XML file, but can be utilized by code generation tools, as well. The main benefit of using XML schemas is that data validation can be provided with little additional work.

3.4 Chapter summary

In this Chapter, we presented the proposed information model. We analyzed the high-level requirements for an architecture that can be applied in highly virtualized, hypervisor-based virtual network environments. We discussed the proposal's relevance to other modeling approaches, as presented in Chapter 2. Lastly, we provided a detailed description of the model's internal characteristics, in particular, of the classes abstracting the core components of a typical modern VNE.

Chapter 4 – Model verification

4.1 Introduction

This chapter presents our methodology for verifying the proposed information model's correctness. We draw upon research and experience in the Cloud Computing paradigm, where resource management is of key importance in a wide array of relevant computer and network environment implementations in Cloud datacenters [33], [34]. Failure to effectively allocate the resources necessary for smooth infrastructure operation may result in impediments regarding successful service provisioning. This condition is more prominent in commercial infrastructures due to the very fact that over- or under- provisioning of computing and network resources can incur negative financial impact. To mention but a few examples, a major challenge faced in successfully administering commercial Cloud environments is the processing of the enormous amount of data, thus, in effect, the dynamic management of the infrastructure's resources - mainly, processing power and memory capacity, virtual networks and network bandwidth, data storage and virtual machines [74], [142]. This allows for cloud elasticity, facilitating infrastructure adaption to varying customer demand (dynamic leasing and releasing of resources). In a related example, data centers employ virtualization at several layers in order to host cloud, online, distributed and other enterprise, commercial or research applications. This is facilitated via heterogeneous networked computing servers, storage arrays and active network elements. Proper planning of resources, accurate provisioning along with targeted monitoring ensures the performance and guaranteed operation of application and services hosted by the data center [33].

4.1.1 Verification approach rationale

In this dissertation we proposed (cf. Chapter 3) a new information model that can be used for describing host-based virtual network environments. As a result, our proposal can lead to the creation of suitable infrastructure management solutions. The key differentiator or our proposal, as compared to other available work (cf. Chapter 2), is the effective handling of virtualized servers and their hypervisors as managed elements of the target infrastructure.

As part of the model's verification approach, we pursued a real-world application of our proposal in an attempt to yield actual results. The ultimate proof-of-concept of the effectiveness and suitability of our information model would be its ability to be applied on actual datacenter infrastructures. Success in this undertaking would, also, validate the proposal's possible application outside of the theoretical domain.

The main information model characteristic that needs to be verified is its ability to describe the infrastructure that it was designed for [30][31]. Towards this direction, we introduced a typical Cloud datacenter virtual network testbed, consisting of IBM UNIX systems and the IBM Power Hypervisor, running different workloads. The particular architecture and choice of systems is not random: they are used in most modern datacenters with the core implementation variation being the choice on specific vendors and their technologies. We used the KF model to describe the aforementioned environment.

In attempting to describe the deployed environment, several of the proposed model's features were verified. Given the inheritance from the CIM parent model it was expected that most characteristics of our proposal would manage well, since these have been repeatedly applied and tested in numerous cases since the CIM model was introduced [15][83]. Modern datacenter infrastructures, however, are constantly changing [62][60] due to new technologies, economic factors, etc. Thus, a model that cannot evolve in order to adapt to the changes introduced in the infrastructure would be rendered obsolete in a very short time frame. **Therefore, the second characteristic that needs to be verified is the proposed model's extensibility**. To that end, we chose something totally absent from the initial model design: an infrastructure resource management method. The choice to test the model's extensibility by adding a new method rather than a new hardware (or other) infrastructure object was that the former posse a more complex task – one that, should it prove successful, would highlight the capabilities of the proposal. We extended the proposed model's schema in order to include a statistical approach for managing the involved infrastructure resources.

Our choice on a resource management method is not linked to information modeling per se but focuses on one of the very key problems in Cloud Computing [74]. Thus, it

facilitates our primary goal: **to propose an artifact that can be applied in real-world applications**. We chose Statistical Process Control (SPC) as a theoretical and practical toolbox for application in managing infrastructure resources. Our choice of SPC was driven by the following facts [143][35]:

i) SPC is a well-established technology in industrial application and could, therefore, be of use in the computing server context.

ii) At the time of writing of this dissertation, such approach does not appear in literature and constitutes a potential distinct research path, independent of the scope of our research.

The model schema extension resulted in appropriate modifications to the model's actual instrumentation (the KF controller software implementation). Once again, in order to remain as close to actual datacenter operation as possible, **we chose two distinct environments and workloads upon to test the applicability of the proposed model extension**. The first workload was based on a commercial core banking software running with real business data. The second workload was based on a typical webserver being fed with network traffic coming from a virtual router. These environments and workloads are found today in Internet Service Providers, Cloud Operators and/or commercial environments [61][34][70]. The metric we focused on for applying the method was the system CPU utilization as **this is the key metric involved in systems performance investigation as well as in related cost/benefit decisions in modern datacenters** [142][65][144]. Hence, the metrics, workloads and datasets used in the verification were as close to actual datacenter usage as possible. **The proposed information model was successful in all aforementioned undertakings.**

The structure of the remainder of this chapter is as follows. In order to establish a background framework on the concepts used, we provide an overview of resource management and SPC approaches in Section 4.2. In Section 4.3 we discuss the theoretical foundation of SPC for resource management. The schema extension implemented on the proposed information model is presented in Section 4.4. The experimental testbed and the test runs performed are described in Section 4.5. Experimental results and limitations of the method, along with metrics used, are discussed in Section 4.6. We conclude this chapter in Section 4.7.

4.2 Resource management and SPC related approaches

The research introduced in this section applies to local scheduling of virtualized resources [34] and fits under the autonomic resource management methods [33]. These approaches are of high importance as they deal with one of the very key problems [74] in the Cloud Computing context; they result in infrastructures that can have their resources dynamically modified as per the varying workload applied on them. Successful methods can have a significant positive financial and service impact on the involved infrastructure.

Several such approaches to resource management in Cloud environments, based on different foundations, have been proposed. Control theory [145][146] has been used in single-tier application environments for adjustment of web server performance and guaranteed response [147], server cache management [148], virtual machine CPU and memory management [149][150]. In multi-tier application environments, [151] proposes an adaptive control system that can dynamically adjust the virtualized resources used in utility computing virtualized infrastructure. Resource allocation in cluster environments has been addressed in [152] as a constrained optimization problem focusing on performance isolation of resource allocations in individual machines and high utilization of server resources. In [153] a pre-computed scheduling graph is proposed for reserving CPU allocation and to implement guaranteed time constraints with accurate a priori feasibility analysis. Authors of [154] argue that the behavior of application workloads is an essential characteristic driving the resource allocation of datacenter resources. To model the resulting system performance patterns the authors propose a time–domain description of a generalized processor sharing server. A constrained optimization technique is, then, used to dynamically allocate the required resources based on the patterns observed. Work in [155] allows for dynamic allocation of the provisioned data center core cluster resources, based on the infrastructure's continuous energy consumption and relevant footprint.

In the cloud computing and data center contexts, efforts are oriented primarily on CPU capacity management (as the core infrastructure resource), so as to meet QoS metrics and application related SLAs in effect. Statistical approaches, where present, are limited to virtual machine migrations and their suitable placement so as for taking

advantage of overlapping/coexistence of workloads. This, in essence, is not an application of a statistical method, but rather a best effort based on workload distribution on the infrastructure in question. Towards that direction, certain proposals rely on statistical multiplexing for consolidating and provisioning multiple virtual machines [144] or, in the same context, for modeling and improving the performance of the consolidated environments, based on the bandwidth resources managed by the hypervisor [156]. Statistical forecasting techniques have been proposed in the context of predicting future infrastructure workloads. These approaches allow for load demand profiling particularly in web applications. Time series analysis is used in [154] for estimating the workload characteristics based on requests flow. Queuing networks have been proposed in [157] and [158] for workload modeling of datacenter-hosted single and multi-tier applications. Proposals in [159] and [160] employ controllers in order to estimate the anticipated infrastructure load, under the generalization of the dynamic resource provisions problem as the cloud elasticity concept. An analytic survey detailing the aforementioned efforts is presented in [33], outlining relevant research and challenges faced.

SPC techniques have been applied outside the manufacturing domain. Consequently, it has been observed that service and infrastructure related problems can benefit from such approaches [143]. Some efforts, of diverse focus, have been proposed in Computer Science and Information Technology. Authors of [161] have introduced a hybrid intelligent tool, IntelliSPC, which is an expert system based on neural networks. IntelliSPC was designed to allow for on-line data collection and processing, providing automated SPC in a controlled environment - demonstrating the use of hybrid artificial intelligence techniques in conjunction with SPC algorithms. The application of artificial neural networks has been introduced in [162] for pattern recognition in control charts, for automatically detecting non-random patterns. A hybrid system has been built and experiment results on the prototype implementation have demonstrated the effectiveness of the method. SPC has received considerable attention from the software engineering world, with a focus on software process monitoring and related metrics [163]. In the same domain, work in [164] introduces control charts for analyzing performance counters across test runs, in order to facilitate automatic performance regression testing in a software engineering environment. Results have been promising, showing that SPC can accurately identify

performance regression in software systems. Proactive fault detection in computing infrastructures, using SPC, is investigated in [165]. Authors of [166] propose an intrusion detection system based on SPC. The system processes and analyzes audit events employing two detection techniques, one for anomalies and one for misuses. The application of SPC in the autonomic computing context, for application service quality, has been investigated in [167]. In [168] authors propose an expert system for the online construction of control charts. The proposed system does not require the human intervention of an SPC-skilled person. Finally, in computer networks, SPC has been proposed in [169] for determining the wireless channel behavior in IEEE 802.11 networks in order to facilitate better user centric management. In [170] authors apply multi-scale control charts in monitoring and analyzing communication on IEEE 802.11 and Bluetooth wireless systems, for effecting higher levels of quality of service (QoS).

Our proposed approach as compared to the aforementioned resource management work differs in several aspects, apart from the underlying mathematical foundation. From a managed infrastructure element perspective, related work can be grouped into proposals targeted on the application/middleware layer ([145][146][147][148][158]), on single virtual machines (VMs) ([150][151][153]), on clustered VMs ([152][154][144]), on specific technical characteristics such as energy profiling [155] or rack-wide network bandwidth [156], and on data center scale VM mobility and provisioning ([157][159][160]).

All these proposals have a key goal: that of achieving the performance requirements expected from the managed infrastructure. To meet the set performance goals each proposal employs different metrics for monitoring the infrastructure and for triggering resource provisioning actions (an overview of the characteristics of related proposals, including a detailed list of metrics used, is presented in *Table 4. Overview of related resource management proposals*).

Our proposal focuses on self-adaptation of single or clustered VMs and is mostly concerned with the CPU utilization along with the virtualized infrastructure's dynamic resource allocation adaptation time, as the key metrics involved in the

calculations. Our technique can complement other ones relevant to local resource management, especially those focusing on broader infrastructure behavior, such as Cloud local and geographical elasticity ([156][157][159][160]) as well as energy considerations when related to the performance of VMs ([155]). It is possible for our work to be incorporated in the aforementioned frameworks in order to support and facilitate local management decisions. However, we do not focus on Cloud services and their auto-scaling when provisioned locally or when dispersed over different Clouds (e.g., as in [171] and [172] - this is a different area of research [33]). From a workload perspective, our technique does not provide prediction mechanisms such as those found in time-series based as well as other forecasting approaches ([154][144][156][157]). In contrast, our technique is based on online workload characterization. It can, however, be combined with a forecasting approach, as part of a general management solution, in order to provide proactive actions and alerting when the resource capacity of the managed infrastructure will be nearing its limits, as per the anticipated workload pattern. Therefore, this option can be used as a failsafe mechanism in order to avoid resource exhaustion, something that would result in the failure of any resource management operation.

Approach	Foundation	Target Layer	Validation Data	Metrics
[145]	Control theory	Application	Synthetic	Transactions/s, no. of users
[146]	Control theory	Application	Synthetic	HTTP requests, TCP connection delay
[147]	Control theory	Application	Synthetic	Web server request rate, system resources utilization
[148]	Control theory	Application	Synthetic, Empirical	Proxy server request rate, cache storage utilization
[150]	Control theory	Single VM	Synthetic	VM virtual clock time
[151]	Control theory	Single VM	Synthetic	CPU {utilization, cycles}
[152]	Constrained optimization	Clustered VMs	Synthetic	Web server request rate, cluster capacity utilization
[153]	Graph theory	Single VM	Synthetic	Execution duration, CPU reservation time
[154]	Time-series analysis,	Clustered VMs	Synthetic, Trace	No./rate of HTTP requests

	Non-linear optimization			
[155]	Greedy algorithm	Energy	Synthetic	TCP connections, Server power draw
[144]	Statistical multiplexing, Time-series analysis	Clustered VMs	Real	VM capacity and utilization
[156]	Statistical multiplexing, Stochastic Bin Packaging	Network Bandwidth	Synthetic	Network bandwidth
[157]	Queuing theory, Statistical forecasting	Data Center	Synthetic	Server {response time, throughput}
[158]	Queuing theory	Application	Synthetic	Application request rate
[159]	Queuing theory	Data Center	Trace	Service request rate
[160]	Wavelet functions	Data Center	Trace	Service request {rate, type}
Our solution	*Statistics (Statistical Process Control)*	*Single, Clustered VMs*	*Real*	*CPU utilization, adaptation time*

Table 4. Overview of related resource management proposals

The technique proposed in this section essentially forms a threshold-based rule reactive approach. These approaches lack a formal evaluation methodology of their effectiveness (including the absence of a commonly accepted scoring metric). They appear simple in their design with the most difficult part being that of determining the proper threshold limits as per the distinct characteristics of individual workloads [173]. This lack of verification formality hardens the task of objectively assessing our proposal's effectiveness, as compared to other approaches. One overall factor, yet hard to quantify, is the level of difficulty inherent in applying the approach. This fact has been reported in the literature in the context of resource management in cloud environments [174][34].

Rule-based approaches, as compared to control theoretic ones, are easier to implement, simple and convenient, provided that the user can correctly determine the scaling indicators and the required performance goals. In essence, if the workload characteristics are known then a rule-based method is easier to adopt.

We have verified this fact given that a well-known feature of SPC is its ability to facilitate rapid prototyping while maintaining the robustness of the mathematical formulae for the determination of the involved thresholds. As part of the verification approach, we proceeded via application with real data and data center virtualized infrastructure running a commercial batch workload, in an attempt to yield actual results. All related proposals have been validated through experiment and simulation, but, apart from one proposal [144], did not use real data in the experimental implementation. Proposals [148][154][159] and [160], in part, used trace data from past Internet portals. We have, thus, provided a first proof of suitability of our proposal for application on production infrastructures. Further comparison between our and the related proposals, based on specific technical or other attributes, is not feasible. The lack of a formal testing and comparison framework along with the fact that each author focused on different Cloud architecture directions and used totally diverse infrastructures, metrics, testing scenarios, workloads, etc. make direct comparative assessments more difficult. This problem has already appeared in auto-scaling in the Cloud context [173].

4.2.1 Rationale of the proposed SPC foundation

We chose SPC as a theoretical and practical foundation candidate for application in managing infrastructure resources driven by the fact that SPC is a well-established technology in industrial application. This presents specific advantages (discussed below) not found in other methods [143][35].

The key differentiator of our proposal, as compared to other related work discussed, is the effective handling of virtualized managed elements of the target infrastructure via leveraging the SPC advantages. While several proposals exist in the resource management context, as presented in [33], the application of SPC to meet the same objectives offers substantial advantages, not found in related work. The key advantage

of SPC is that it focuses on a measurable process as a whole and not on any particular attributes of it: the behavior of the process is the collective outcome of variation in any of the processes' associated attributes. Thus, monitoring and controlling the process helps to minimize this variability.

Determining threshold limits in SPC is a robust, well-established procedure with a solid theoretical foundation and, most importantly, verified by decades of actual industrial application [175][176][163][143]. Related work presents a diverse array of techniques for scheduling the virtualized resources. Nevertheless, no proposal encompasses a holistic approach manifested as a single vector reflecting the impact of different factors. Local allocation of resources is addressed as a result of a primary factor (CPU utilization, network latency, etc.). Hence, the techniques employed can be successful for handling that factor alone [33]. On the other hand, the aforementioned key differentiator leads to additional individual positive repercussions: when SPC is properly applied, using the data gathered, makes it easier to analyze the process in question, i.e., which attributes affect it and in what manner. Therefore, better process understanding is achieved – something not possible when focusing on a single, particular attribute. This is of the utmost importance in actual data center infrastructures as identifying root causes can be a challenging task. Hence, it is possible to investigate a process in a top-down approach and, gradually, rule out causes of process variability. In some cases it is even possible to detect problematic factors previously considered normal and excluded from analysis.

These advantages have been leveraged for years in the manufacturing context, where SPC was first introduced, and are mature and accepted approaches. Although properly designed experiments, such as those described in related proposals [33], can quickly resolve some of the relevant issues, an initial thorough examination of the SPC data can be valuable.

Another important advantage in applying SPC in a threshold-based rule reactive approach is that the latter, at their present form, cannot handle unexpected changes in the workload pattern – this being the main disadvantage of the techniques. The inherent analytical capabilities of SPC, though, provide indication on when a process is deviating from the normal down to a problematic state. It is, therefore, possible to

eliminate or reduce infrastructure operation and quality issues before they appear. Variation in the measured process is reflected in the calculated thresholds and change in the workload patterns can be handled dynamically as long as the process pattern remains symmetrical. This is reflected in the normal distribution of the measured data. Related proposals cannot support such foregoing handling of the managed infrastructure unless they are combined with time series based methods applied for yielding predictive results [173] for the anticipated workload level. In the case of asymmetric process patterns SPC offers specific tools that can handle such cases. Although these tools are widely used in industrial applications, their applicability in the resource management context and in our proposed approach has not been verified – it constitutes an area of further research.

Overall, SPC can be considered as an embracing approach which allows for rapid assessment and maintenance of a process' quality without the need of investigating and regulating each individual process attribute. These attributes do exist and influence the final outcome, as properly suggested in related proposals. However, the statistical analysis of the process as a whole remains consistent. Consequently, we argue that SPC can provide the basis for a new resource management approach in the datacenter context. The results of our research indicate that SPC has a solid application ground.

4.3 SPC-based resource management theoretic framework

SPC is a term, collectively used, for specific analytical methods which employ statistical techniques to process monitoring and control, in order to ensure that it operates within designed limits for acceptable output. Under SPC, a process behaves predictably to produce as much conforming product as possible with the least possible waste. While SPC has been applied frequently to controlling product manufacturing lines, it applies equally well to any process with a measurable output. SPC, not only indicates when an action should be taken in a process, but it also indicates when no action should be taken – the latter when a process behaves as expected. The reader is referred to [175] for a detailed discussion on SPC. The term process, as used in the context of this paper, refers to the CPU utilization of a virtual machine (VM). Hence, undesired process (CPU utilization) levels may result in impediments in the service

offered by the infrastructure. Towards this direction the problem of using SPC to manage the VM's performance can be described, in general terms, as follows.

Let *X* represent the VM's CPU utilization while executing under a particular payload. The distribution function of *X* is indexed by *V*, a number of different factors (type of workload, processor parameters, system tuning or other workloads running on the same infrastructure, etc.). In SPC terminology, process *X* is operating at a stable state when *V = V0* and is referred to as being in-control. *V0* is the baseline for successful VM operation against which comparisons can be made. Obviously, there is no ubiquitous accepted value for *V0*; this is case-based and depends on the particular problem. Therefore, variations in *V* against the baseline *V0* are reflected as variations in *X* and allow us to determine when an action should be taken in order to bring the process back in-control.

Key tools in SPC are the Shewhart Control Charts [177] (also referred to as $\bar{X}$–S or $\bar{X}$– R charts). These charts can be used for on-line process monitoring in order to show how they perform over time and how their capabilities are affected by changes introduced to the processes. This information is used next to make quality improvements. Control charts are also used to determine process capacity by helping identify special or assignable causes for factors that impede peak performance. Control charts rely on well-set control limits. These must be defined in such way that the value of *X* is unlikely to fall outside these limits when the VM performs normally, at acceptable service levels (i.e., when V = *V0*). VM operation can be managed by the ongoing control of process *X*, essentially by monitoring CPU utilization variation over time and by detecting anomalies. These anomalies occur as a result of unwanted deviations of *V* from the established baseline *V0*. When this occurs, management action is justified and corrective procedures can be applied. Consequently, VM operation quality, resulting cost and effective capacity can be optimized. Additionally, retrospective assessment or prospective capacity planning by extrapolation are possible based on the analysis of the monitored data.

It has become apparent in recent data center implementations that, since most modern system and network infrastructure implementations are virtualized, the virtualization

overhead imposed by the hypervisor layer (present in all parts of the infrastructure, i.e., server, network, storage, etc.) needs to be assessed and evaluated [178][179]. The very use of virtualization technology introduces certain facts that need to be addressed, as traditional approaches to resource management may not apply, or be sub-optimal for use in the virtualized landscape. From the resource management perspective it is crucial that, apart from the virtualization overhead, two other information facts are taken into account in any management foundation:

i) The available (free) resources.

ii) The resource utilization patterns imposed by the workloads running on the virtualized infrastructure.

These points of interest, along with any other that affect the VM's performance, are all reflected in vector *V* where, by reciprocation, the state of *V0* is the accepted state of all resources, indicated by VM CPU utilization within the defined control limits. Our proposed approach allows adaptation to infrastructure changes as these are, ultimately, reflected in the resource utilization. By employing control charting analysis on the VM CPU utilization patterns, it is possible to identify data variation and react appropriately.

4.3.1 Setting the Control Limits

Daily server operation varies considerably and resource utilization graphs of a business or cloud virtualized server running a typical workload over a 24-hour period usually exhibit a spiked spread (cf. *Figure 13. Typical time-varying server resource utilization graph*). Here, the server's resource is consumed at lower or higher levels running time-varying workloads [180]. In heavily consolidated servers, where each hosts several workloads, the utilization graph may appear smoother with a flat, sustained, utilization. Nevertheless, spikes do exist and can be revealed if a smaller monitoring sampling interval is used.

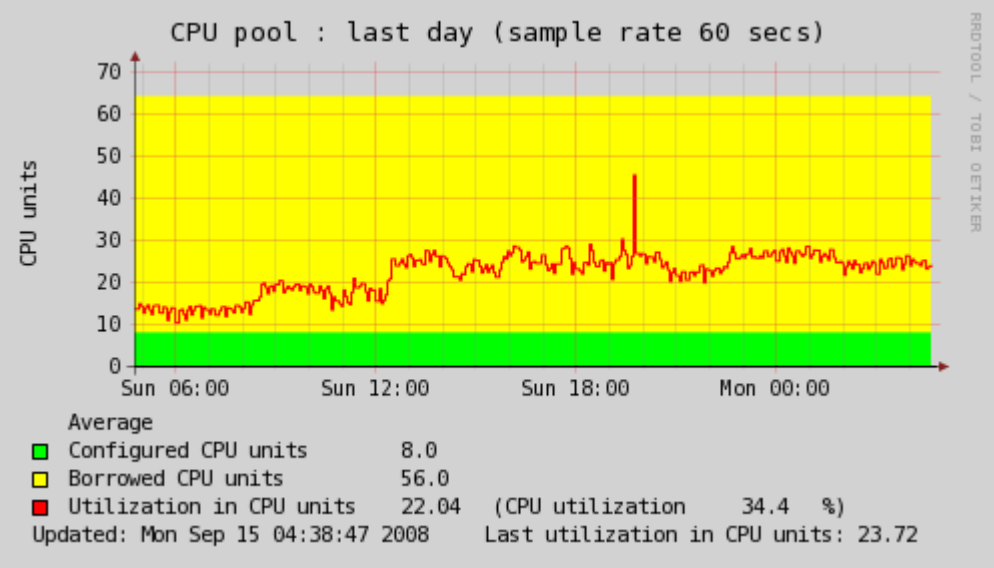


Figure 13. Typical time-varying server resource utilization graph

Whichever the case, the overall spread follows a random pattern with a standard normal distribution, characterized by specific statistical attributes, expressed by standard Statistics or SPC textbook formulae [181], as follows:

a. the Mean, $\bar{\boldsymbol{X}}$, refers to the average CPU utilization measured granularly over a time period, formulated by

$$\bar{\boldsymbol{X}} = \frac{\boldsymbol{x_1 + x_2 + \ldots + x_n}}{\boldsymbol{n}} = \frac{\sum_{i=1}^{n} \boldsymbol{x_i}}{\boldsymbol{n}} \quad (1)$$

where x_n denotes each CPU utilization measurement (sample). Similarly, the average of all sample means, $\bar{\bar{\boldsymbol{X}}}$, for several measurement time periods, is the average of each period's calculated mean value $\bar{\boldsymbol{X}}$, formulated by

$$\bar{\bar{\boldsymbol{X}}} = \frac{\bar{\boldsymbol{X}}_1 + \bar{\boldsymbol{X}}_2 + \ldots + \bar{\boldsymbol{X}}_k}{\boldsymbol{k}} = \frac{\sum_{j=1}^{k} \bar{\boldsymbol{X}}_j}{\boldsymbol{k}} \quad (2)$$

b. the Range, $\boldsymbol{R}$, is the difference between the highest and lowest recorded CPU utilization samples across the time period under measurement. The Mean Range, $\bar{\boldsymbol{R}}$, is the average of the ranges calculated for several distinct measurement time periods, formulated by

$$\bar{\boldsymbol{R}} = \frac{\boldsymbol{R_1 + R_2 + \ldots + R_k}}{\boldsymbol{k}} = \frac{\sum_{i=1}^{k} \boldsymbol{R_i}}{\boldsymbol{k}} \quad (3)$$

c. the Standard Deviation, $\boldsymbol{\sigma}$, is a measure of how widely the intervals differ from the mean $\bar{\boldsymbol{X}}$, formulated by

$$\boldsymbol{\sigma} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{X})^2}{n-1}} \quad (4)$$

d. The average sample standard deviation, $\bar{s}$, is the average of the respective values calculated for several measurement time periods:

$$\bar{S} = \frac{\sigma_1+\sigma_2+\ldots+\sigma_n}{n} = \frac{\sum_{i=1}^{n}\sigma_i}{n} \quad (5)$$

The normal distribution for the CPU utilization of a virtual server, running a specific workload will be consistent across time frames, as long as the workload pattern does neither change nor there are parameters affecting normal server operation. Should the latter occur, a different distribution will be observed and the root causes will be manifested as different utilization measurements and variances. In the absence of extraordinary conditions, the distribution pattern of the observed utilization data will exhibit a majority 95% population of recorded values at less than a $3\times\sigma$ difference from the calculated mean $\bar{X}$ (the condition often referred to as the 68%-95%-99% rule [180]). Hence, the appearance of CPU utilization at levels greater than $3\times\sigma$ might indicate a possible error state in the infrastructure.

Before any SPC method can be applied, it is necessary to setup the proper control limits. The number of recorded measurements and data points produced when monitoring VM CPU utilization is, usually, quite large (in the range of hundreds or thousands per measurement period). Therefore, the $\bar{X}$–S chart is the appropriate tool for this data processing as, for measurement data sets of more than 12 values, the $\bar{X}$–R chart is inefficient in properly handling the range (R and $\bar{R}$) [181]. The control limits for the $\bar{X}$–S chart are calculated as follows:

e. The Center Line, CL, the Upper Control Limit, UCL, and the Lower Control Limit, LCL, respectively, for the $\bar{X}$ and the S charts, by

$$CL = \bar{\bar{X}}, CL = \bar{s} \quad (6)$$
$$UCL = \bar{\bar{X}} + A_3\bar{s}, UCL = B_3\bar{s} \quad (7)$$
$$LCL = \bar{\bar{X}} - A_3\bar{s}, LCL = B_4\bar{s} \quad (8)$$

where A_3, B_3, B_4 are constants used for the construction of the control chart.

The reader is referred to [175][176][182] for detailed information on control charting.

4.4 Information model schema extension

Having introduced the experimental architecture (described in detail in Section 4.5), the proposed model was applied in order to describe the resulting managed environment. No further class creation or sub-classing was deemed necessary as the model's available classes were adequate for handling the infrastructure details involved. Next, we focused on extending the model in order to instantiate an SPC management approach on the testbed provisioned resources. The KF model's KV_VM class was extended, as this has been provided for by its design, to include the required methods (cf. *Table 5. KF_VM class SPC methods*).

Method	Purpose	Usage scope
SetControlLimits [IN] UCL: uint32 [IN] LCL: uint32	Calculates the Lower and Upper Control Limits based on the observed CPU utilization	Any VM performance fluctuations outside these limits may result from common causes inherent to the system, such as normal VM imposed load. These normal fluctuations are attributed to statistical variability
CheckStatus [IN] InControl: boolean, [IN] OutofControl: boolean	Checks whether the VM's performance is within acceptable operating parameters for the anticipated and designed load. Implements a control chart equivalent and allows for live pattern analysis	The variance of the VM's performance is compared, over time, against the upper and lower control limits to see if it fits within the expected, variation levels. If so, the VM's performance is considered in control and the variance between measurements is considered a normal random variation inherent in the process. If, however, the variance falls outside the limits, the VM's performance is considered out of control
ModCPU (): uint32	Interfaces with the hosting hypervisor in order to add or remove CPU	Modifies the allocated CPU resources so as to bring the VM in a stable and accepted performance state, as per the control limits in place. This is facilitated by calls to the hypervisor control interface

Table 5. KF_VM class SPC methods

We implemented the required method functionality into the model's provider in order to incorporate the logic for applying SPC on VM-based datacenter infrastructure. The provider processes resource utilization data and determines whether corrective actions are needed. Should this be the case, the latter are applied on the VM in order to bring the environment back in-control, under accepted performance parameters. Operation is performed in an online mode with the provider receiving and processing utilization data at a continuous rate during the desired management period. Thus, the provider assumes the role of a resource controller. A high-level illustration of the resulting experimental concept is shown in *Figure 14. High-level logical control flow architecture*.

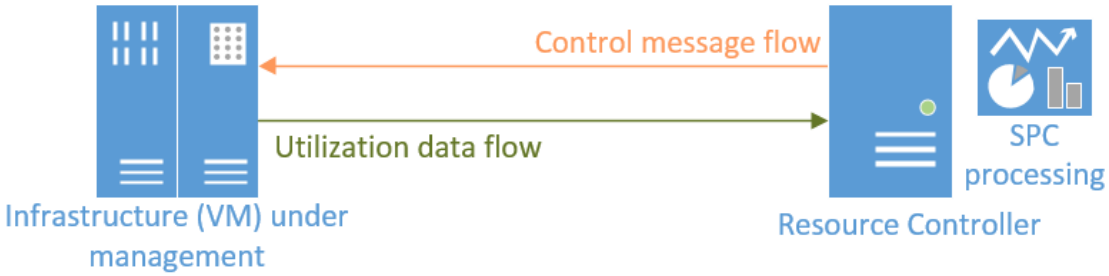


Figure 14. High-level logical control flow architecture

Using the resulting resource controller to manage the infrastructure involves three distinct phases:

i) Preprocessing.

ii) Base-lining.

iii) Online management.

The purpose of the first phase is to collect the required initial utilization data sets from the infrastructure in question and prepare this data so as to be suitable for statistical processing. Any extraordinary performance data which can be attributed to known causes needs to be removed.

The second phase performs the data analysis and produces a baseline, i.e., a depiction of the infrastructure's normal state of operation. During this phase statistical attributes

of the supplied data sets are determined (the population mean, the standard deviation and the associated upper and lower control limits).

Once the performance baseline has been obtained, as reflected by the statistical attributes, the management phase can commence. During this phase, a continuous flow (cf. *Figure 14. High-level logical control flow architecture*) of utilization data is streamed to the resource controller which, in turn, applies the SPC logic and reverts back with the appropriate infrastructure reconfiguration commands. The control loop receives the current VM CPU utilization and compares it with the actual boundaries computed by the SPC processing. Based on the analysis of the behavior observed, the controller dynamically adjusts the VM's resource allocation in response to workload changes. The control loop executes continuously for the entire duration of the desired management period. The resource controller implements an average of all data set means based approach ($\bar{X}$–S) but can be easily modified or extended to realize other workload models. The high-level algorithm for facilitating the three phases is presented in *Table 6. Algorithm for the model-based infrastructure management*.

Step	Phase
1. Collect performance utilization data sets for a number of normal infrastructure usage periods (to be used for calculating each individual sample period mean $\bar{\boldsymbol{X}}$ and the average of all sample means $\bar{\bar{\boldsymbol{X}}}$)	preprocessing
2. Transfer recorded data sets to the resource controller system	
3. Normalize performance data ***For*** each data set *(1..n):* **If** data points attributed to known causes exist, then remove those data points ***End If***	
4. Baseline normal VM operation by calculating the required statistical attributes in the collected data ***For*** *each data set (1..n):* Compute the mean, $\bar{\boldsymbol{X}}$, using equation (1) Compute the standard deviation, $\boldsymbol{\sigma}$, using equation (4) ***End For***	base-lining
5. Compute the average of all sample means, $\bar{\bar{\boldsymbol{X}}}$, using equation (2) and the average sample standard deviation, $\bar{\boldsymbol{s}}$, using equation (5)	

6. Construct the $\bar{X}$–**S** control chart and compute the Upper Control Limit (**UCL**) and the Lower Control Limit (**LCL**), using equations (7) and (8) respectively	
7. Monitor (VM CPU utilization) and manage the infrastructure resource (VM CPU capacity)	
8. *Occurrences_Counter_High* = 0 9. *Occurrences_Counter_Low* = 0 10. *In_control = True* 11. ***Repeat*** until end of management period Retrieve CPU resource utilization from the VM ***While*** *In_control = True* **If** *CPU resource utilization* > *UCL*, **then** Increase *Occurrences_Counter_High* by one ***End If*** **If** *CPU resource utilization* is < *LCL*, **then** Increase *Occurrences_Counter _Low* by one ***End If*** ***If {****Occurrences_Counter_High* OR *Occurrences_Counter_Low****}*** > 3 , **then** *In_control = False* ***End If*** ***End While*** **If** *Occurrences_Counter_High = 3*, **then** Send control message to increase VM CPU allocation by one unit *Occurrences_Counter_High = 0* ***End If*** **If** *Occurrences_Counter_Low* = 3, **then** Send control message to decrease VM CPU allocation by one unit *Occurrences_Counter_Low = 0* ***End If*** *In_control=True* ***End Repeat***	online management

Table 6. Algorithm for the model-based infrastructure management

As discussed in Chapter 3, applying the model on a target managed environment results in an object-oriented approach for describing management entities in that environment. The resulting structured set of infrastructure description is not bound to any particular implementation. Thus, it enables platform-independent and technology-neutral exchange of management information, providing a consistent data definition

and structure. Given the standardization followed, it becomes easier to port the model's provider (i.e., the resource controller) to other technology platforms.

4.5 Experimental architecture

4.5.1 Hardware setup

We implemented an experimental technical architecture for applying the proposed information model, focusing on a realistic assessment based on real-life and not synthetic data.

The technical setup consists of an IBM pSeries 740 UNIX server divided into three logical partitions (LPARs - VMs), each VM running IBM's AIX operating system version 7.1 [36]. Employing the hypervisor capabilities, each VM was assigned a certain amount of p7 CPUs and GBs of RAM memory along with a virtual Ethernet adapter realized via the in-memory IEEE 802.1Q virtual switch [41]. The precise amount of allocated resources depends on the requirements of each experiment run on the infrastructure. A high-level diagram of the hardware setup, common for all the experiments run, is given in *Figure 15. Experimental virtualized architecture*.

On **virtual machine 1** we developed and installed appropriate code (KornShell93 scripts and C binaries) that implements the model's resource controller functionality. In addition, this VM serves as an aggregation system, collecting performance information from the managed VM, gathering utilization patterns and producing consolidated statistics of the core resources provisioned by the hypervisor. Operating system specific monitoring tools gather performance related information (`sar` and `vmstat` commands) and SSH DLPAR operations, via the p740 system's Hardware Management Console (HMC), provide for the dynamic reconfiguration of resources, as per the control messages sent by the controller.

Virtual machine 3 was deployed as the target system to be managed via the model's resource controller. We applied the actual workload on this system. This architecture and systems configuration was used in both cycles of the experiment; the varying parameter was the workload imposed on the target VM system:

- For the first test cycle we deployed the Temenos T24 Core Banking System software [37] along with an actual business data set. The CPU capacity of the target VM was modified in such way as to result in less resources available than necessary, thus creating a negative impact on the VM performance. The VM was, then, managed by the controller in an attempt to return it to an acceptable performance state.
- For the second cycle we implemented an intermediate virtual router (on **virtual machine 2**) based on the *gated* daemon, routing network packets to a web server installed on the target VM, **virtual machine 3**. The load imposed on the virtual router was gradually increased in an attempt to verify the controller's appropriateness in returning the VM to an acceptable performance state (by successive CPU allocation modifications).

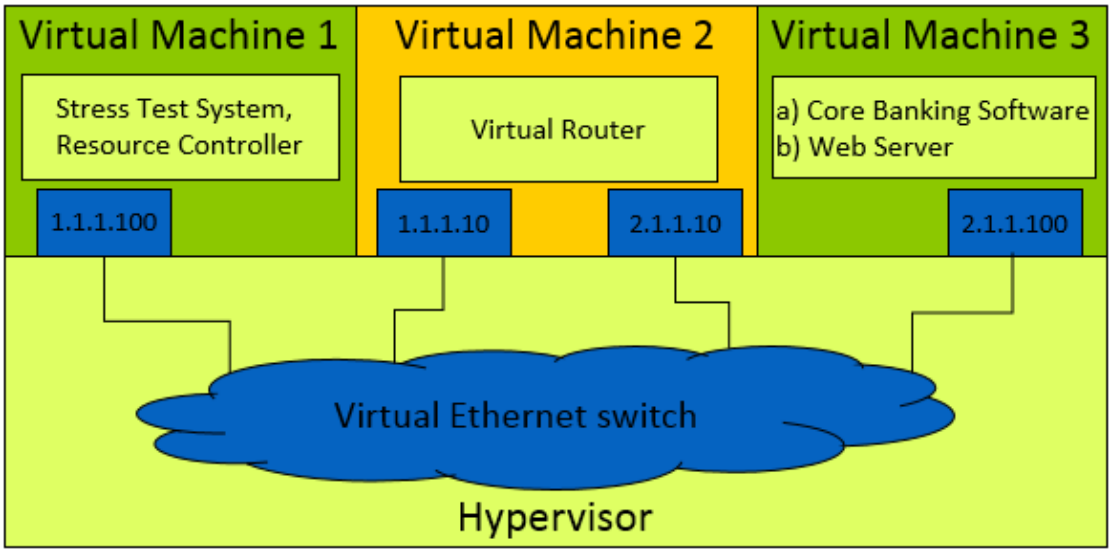


Figure 15. Experimental virtualized architecture

All data flow between different components is based on messages structured in XML, a mature and widely accepted standard [100]. Apart from the well-established benefits of the language, XML representation increases the possibility of using the created messages in a wider array of management (or other) applications. The target VM is continuously monitored and its utilization status is reported to the resource controller via suitable messages (space delimited values of primary CPU statistics). The information received is statistically processed by the controller and, if proved

necessary, infrastructure-related decisions are made. These are instantiated via control messages and infrastructure reconfiguration commands issued by the controller.

Table 7. Example of V-SPC infrastructure management messages and commands shows a brief example of such events sequence: the controller receives a resource utilization message containing several attributes of CPU state at a particular point in time. This data is routed to the SPC algorithms and processed against previously calculated baselines. A decision is made to increase the CPU capacity of the managed VM by one unit. The necessary infrastructure reconfiguration command (essentially, KornShell93 or C executable code) is constructed and executed by the resource controller. The utilization messages as well as the reconfiguration commands are platform specific and depend on the deployed hypervisor and operating system used.

Resource utilization message	Infrastructure reconfiguration command
<MESSAGE ID="2368"> <NAME="ProcUtil"> </NAME> <VALUE=” 13:37:08 %user %sys %wait %idle physc %entc lbusy app vcsw phint 47.1 20.6 2.3 29.9 8.15 108.7 24.2 1.32 11562 826 ”> </VALUE> </MESSAGE>	FRAME=Server1-SN06C0121 HMC=hmc1 PROC_TO_ADD=1 VPROC_TO_ADD=1 NODE_TO_ADD=SRVCBP1 ssh hscroot@${HMC} "chhwres -r proc -o a -p $NODE_TO_ADD -m $FRAME --procunits $PROC_TO_ADD --procs $VPROC_TO_ADD -w 1"

Table 7. Example of V-SPC infrastructure management messages and commands

4.5.2 First test cycle (core banking software)

In the first cycle, during the preprocessing phase we run five times a T24 batch processing procedure on the target VM, using typical business data of a core banking environment, resulting in similar and anticipated workload patterns for each run. The five-time cycle provides a confidence interval for obtaining statistical measurements.

The VM's processor allocation was initially capped at 10 CPU units, which resulted in a run window of one hour for servicing the particular workload. This is considered as the acceptable process output (baseline duration) for the given experiment, under a static resource allocation environment.

The VM was monitored during each run and detailed performance information was collected, resulting in the creation of five distinct performance data sets (cf. *Appendix E – Table 6*). The illustration in *Figure 16. Aggregated, non-normalized, VM CPU utilization* shows the aggregated utilization graphs, for the five run times (in different coloring). The performance data was normalized and the data points that could be attributed to known causes were removed. To that end, data from, approximately, one minute duration at the start and finish times of each run (regions 10:00:00-10:00:58 and 11:00:13-11:01:00) was deleted, since it corresponds to VM utilization at stages where no workload-related processing occurred.

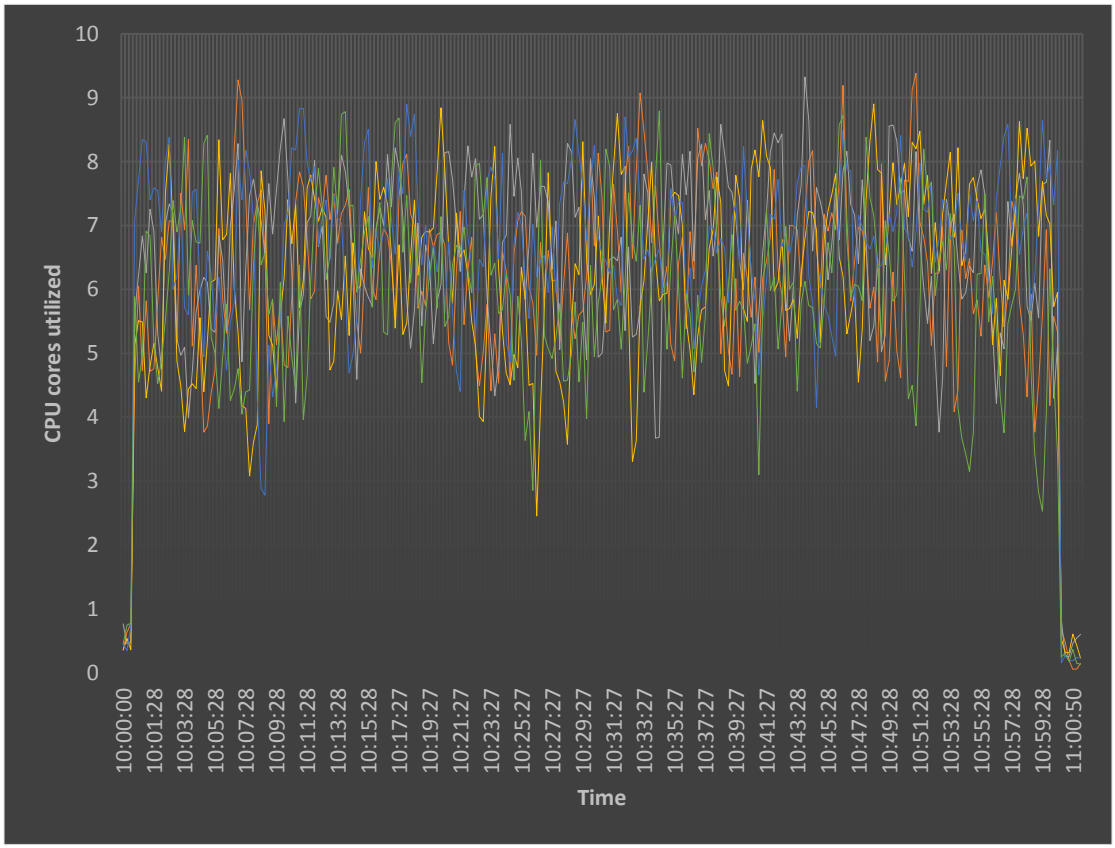


Figure 16. Aggregated, non-normalized, VM CPU utilization for the five test runs

Following data collection and preprocessing, the statistical representation of the accepted baseline was established, by calculating certain statistical attributes of the collected data sets ($\bar{X}$, $\bar{\bar{X}}$, σ, $\bar{s}$, *CL*, *UCL*, *LCL*). Control charting yielded the values of *UCL* = 71.09, *CL* = 64.52 and *LCL* = 57.95 (cf. *Figure 17. X-Bar control chart showing UCL, CL and LCL values* for the SPC *Xbar* chart) with data points exhibiting a normal distribution and a mean value of 64.52 (cf. *Figure 18. Data points distribution and mean value*). These values were consecutively used for the next stages of the experiment. The distribution observed provides an indication of the fluctuation patterns in the particular process variable (the CPU utilization) and was used as the baseline for the subsequent comparisons with newer data collection.

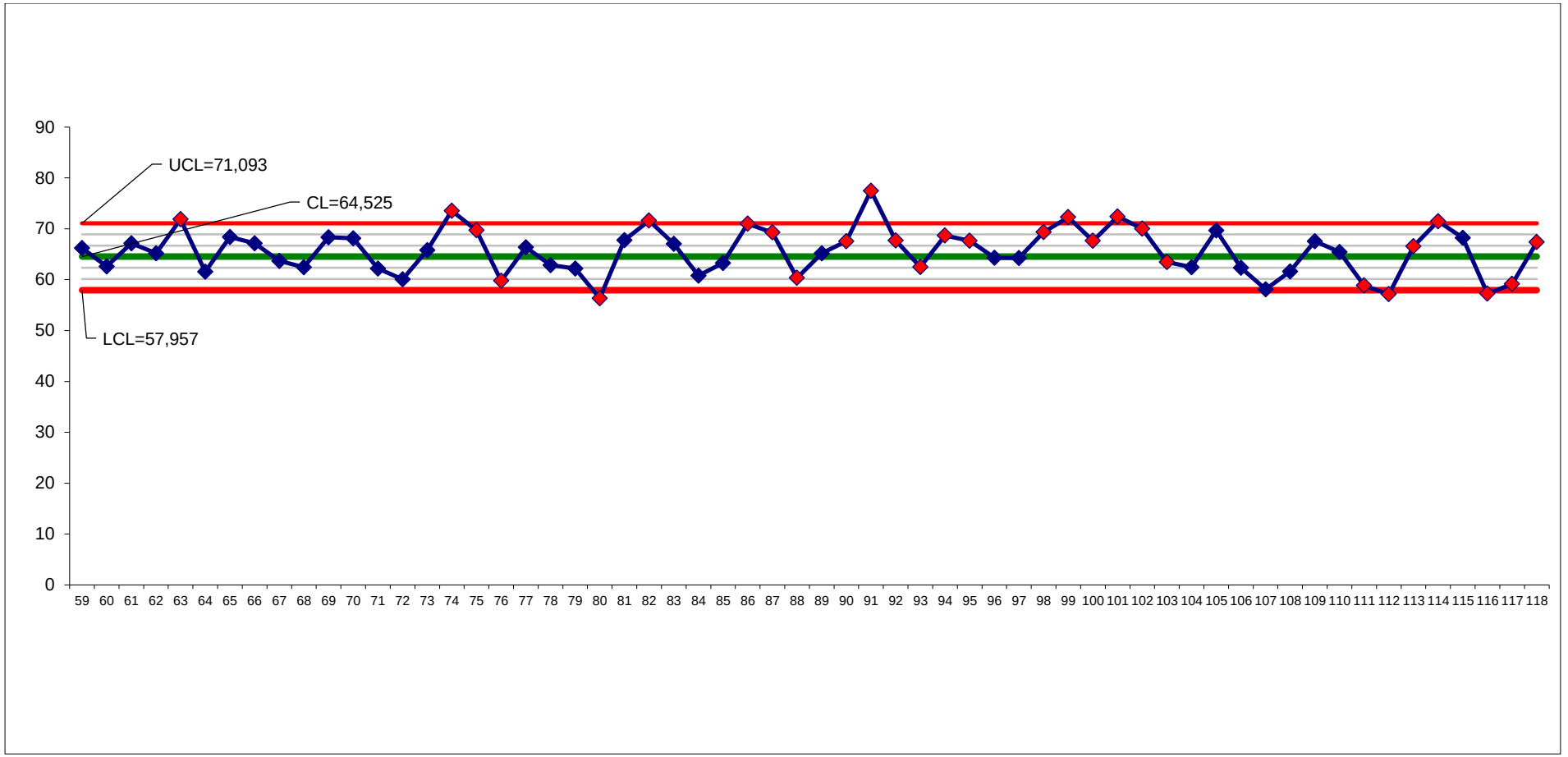


Figure 17. X-Bar control chart showing *UCL, CL* and *LCL* values

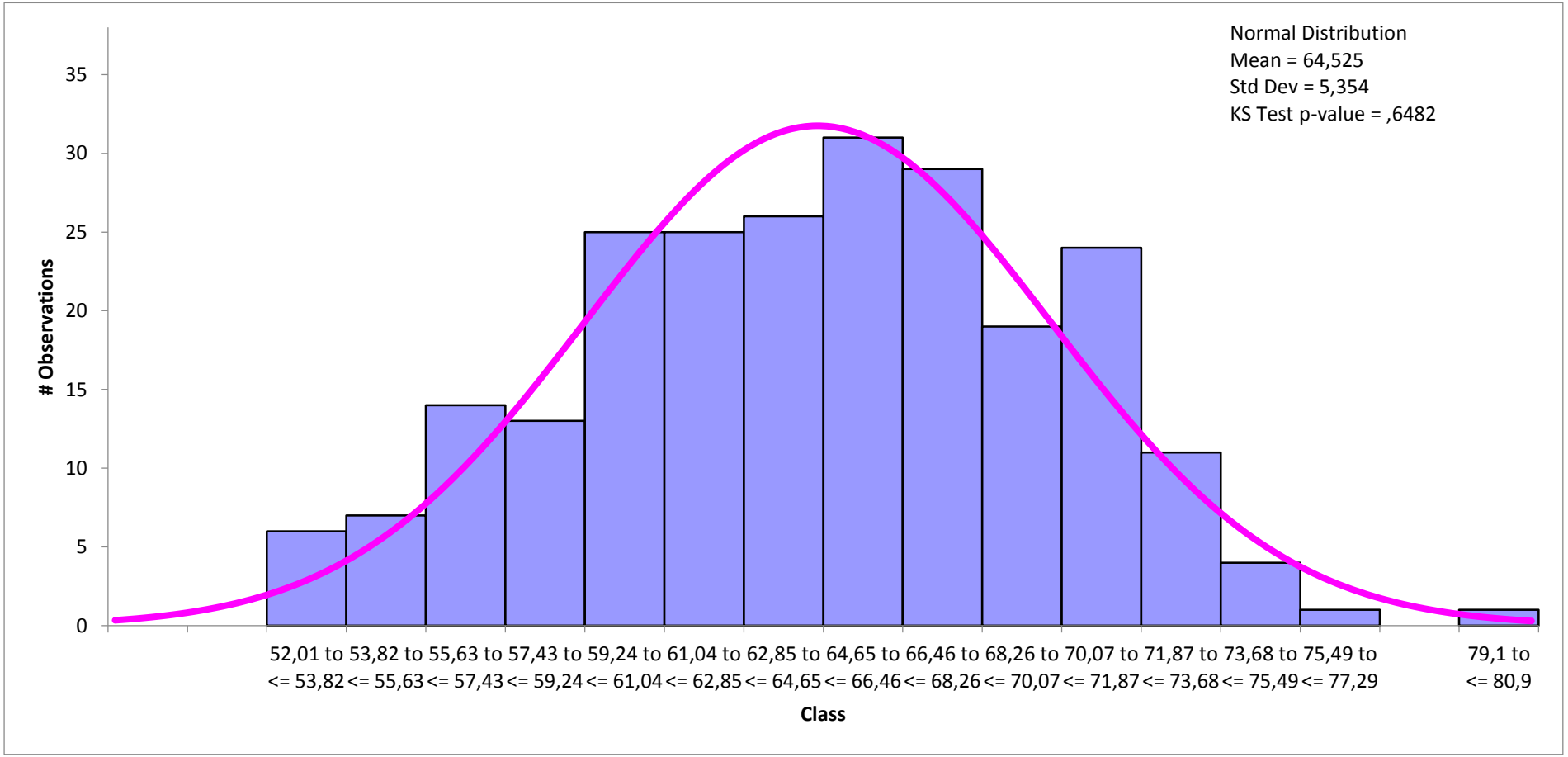


Figure 18. Data points distribution and mean value

The VM's processor allocation was decreased by 50% (5 CPU units), so as to lead to processor starvation for the applied workload and to force corrective procedure triggering. Two new test runs were initiated on the decreased CPU-configuration VM: one without any external management action, and one with the KF resource controller actively managing the VM, providing a dynamic resource allocation environment. Performance data was collected for each run.

The run on the unmanaged VM, using half of the original CPU capacity resulted in an extended duration of one hour and 12 minutes, increased by 20% as compared to the acceptable process duration of one hour. The run on the managed VM resulted in a duration of one hour and 2.5 minutes, nearly similar to the acceptable one, with the VM's CPU allocation stabilized at eight units. The aggregated, graphical experimental results and a timeline of CPU capacity modification as initiated by the KF resource controller are presented in *Figure 19. Aggregated VM CPU utilization* (showing CPU utilization for the three test runs) and *Figure 20. VM CPU capacity after each management action* (showing the number of virtual machine CPU over time), respectively. The graphs across different runs show common patterns due to the fact that the exact source application data set was used unaltered, for each run.

The choice of the workload and the virtualization architecture is typical of those found in commercial public and private Cloud datacenters (Internet Service Providers, Cloud Operators and/or commercial environments [61][34][70]). The workload used in the experiment consisted of T24 core banking jobs processing a data set, as structured for a typical nightly batch processing procedure [37]. The total data volume was sized at 500GB of storage area network provisioned capacity, of which 240GB were consumed by the business data. Each test run included the same, resource-intensive, jobs executed sequentially, indicatively:

- SYSTEM.END.OF.DAY5-EOD.UPDATE.CATEG.ENT.MONTH
- FILE.TIDY.UP-EB.CLEAR.FILES
- FT.START.OF.DAY-FT.LOCAL.DATA.PURGE
- REPORT.PRINT.REPORTING-EB.EOD.REPORT.PRINT
- UPD.DEL.CYCLES-EXTRACT.WRONG.BKTS
- RE.UPDATE.SLC.LINE.BAL-RE.UPDT.STAT.LINE.CONT

Each of the involved jobs executed during the test runs created a different stress level on the allocated processors and resulted in varying CPU utilization. Given that the data under processing remained static for each test run, the overall execution time for the entire set of jobs depended solely on the number of the allocated resources – in our case, on the number of allocated processors.

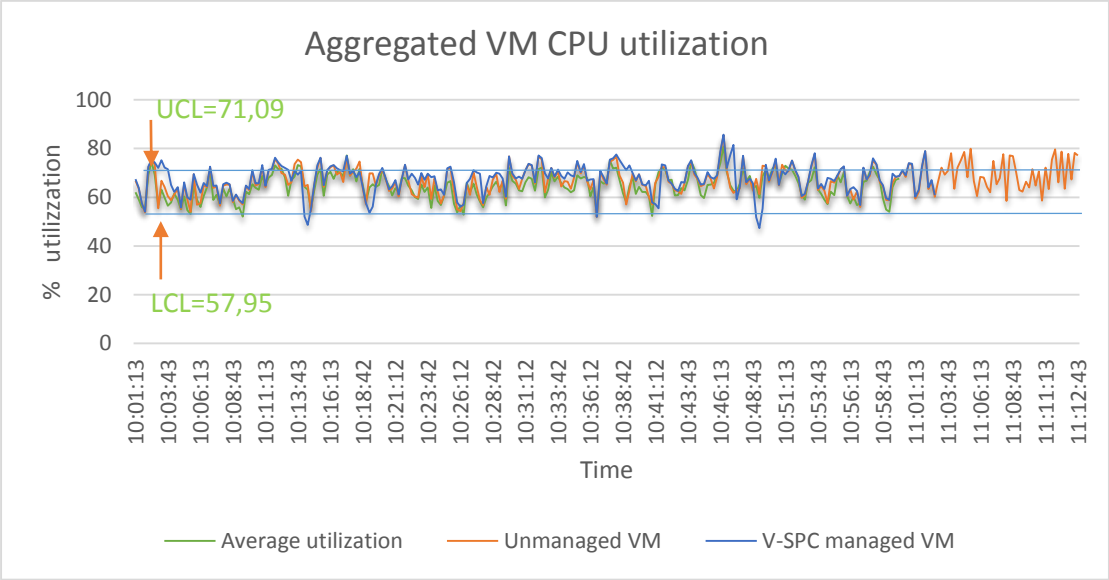


Figure 19. Aggregated VM CPU utilization

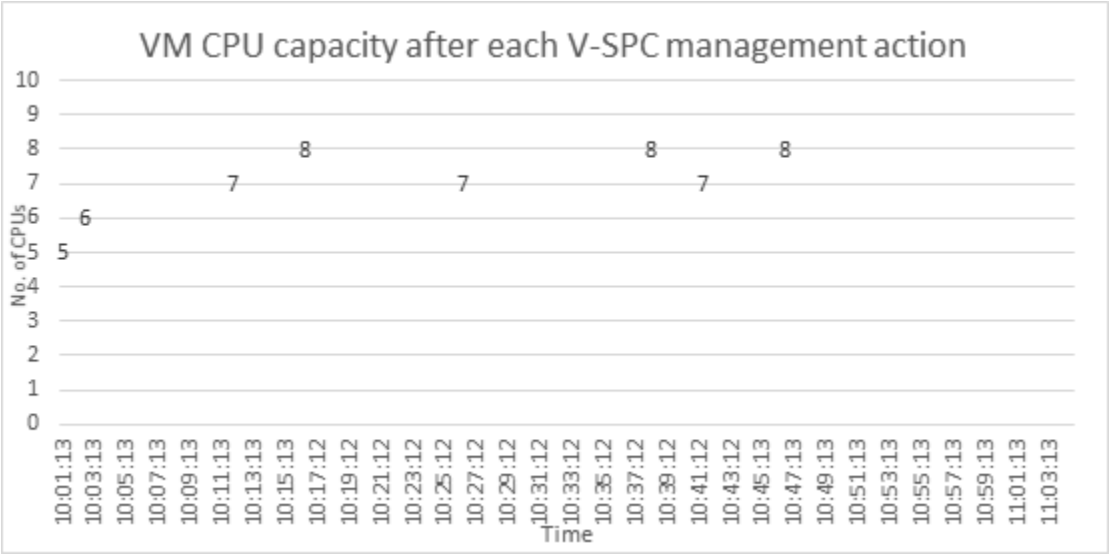


Figure 20. VM CPU capacity after each management action

4.5.3 Second test cycle (virtual router)

In this experiment we proceeded to further test the abstraction capabilities of the model, focusing on networking aspects. In addition, we provided another evaluation of the suitability of SPC as a valid resource management approach. For this cycle, each VM was assigned two p7 CPU, one GB of RAM memory and two virtual Ethernet adapters realized via the in-memory IEEE 802.1Q virtual switch. Each VM was configured on a different virtual LAN and assigned an IP address of its own address space. A third VM was introduced acting as a target web server. The first VM assumed the role of a stress test source system, running custom code in order to hammer the web server running on the third VM. Network-wise the stress station and the web server VMs use the second VM, which assumes the role of a network gateway. This second VM runs a copy of the "gated" routing daemon and serves as a virtual router for the 1.1.1.0/24 and 2.1.1.0/24 IP networks (for the first and third VM, respectively).

The first VM serves an additional role as an aggregation system: it collects performance information from the virtual router and the web server, gathers utilization patterns and produces consolidated statistics of the core resources provisioned on each VM. Furthermore, this VM collects hypervisor statistics for performance data reconciliation produced at the operating system level, against data provided by the physical system's hypervisor. The proposed model's features adequately describe the aforementioned test architecture and host-based virtual networks via the KF {Node, Hypervisor, HypervisorVirtualSwitch, VM and Network} classes. Virtualized hardware characteristics, such as provisioned CPU cores, memory and virtual Ethernet adapters' capacity are contained in the respective array of class attributes (e.g., AllocatedVirtualProcessors, AllocatedEthernetBandwidth).

In a similar fashion, resource allocation is gathered via AIX monitoring tools (`sar` and `vmstat` commands) and SSH DLPAR operation commands to the p740 system's Hardware Management Console (HMC). The resulting information is handled via the attributes VirtualProcessorsUsed, MemoryUsed and

AggregatedEthernetIO. Inheritance from the CIM Network class allows for network attributes description such as VLAN ids, IP addressing, subnet masks, etc.

During this experiment, the virtual router and the web server VMs were load-stressed using the Apache Foundation "jmeter"[5] and the IBM "nstress"[6] software packages. Dynamic VM CPU reconfiguration was attempted on the virtual router VM via the implemented SPC methods. Prior to applying the methods, five test runs were executed for which detailed performance data was collected (cf. *Appendix E – Table 7*). This data was analyzed and the required statistical attributes were calculated. Next, a new three-hour test run was, executed with the virtual router VM's CPU capacity initially decreased to one core so as to enforce and simulate a lack-of-resources condition. The imposed workload was increased by 100% every 30 minutes, for 5 consecutive cycles, before being restored to the initial stress level.

The resource controller successfully modified the allocated CPU resources in real time, covering the increase in processor power demand (cf. Figure 21. Virtual router VM CPU utilization and CPU capacity over time). The illustration shows the five distinct increases in the system's CPU utilization and the respective allocated CPU cores at any point in time. Once the high load had ceased, the allocated resources were released back to the hypervisor CPU pool, remaining available for any other CPU provision request. The resource controller successfully managed the VM, dynamically modifying its CPU capacity as per the observed CPU utilization changes, adapting to varying resource demand. The VM remained in controlled operation, avoiding poor service provisioning due to resource starvation.

[5] jmeter.apache.org
[6] www.ibm.com/developerworks/community/wikis/home/wiki/Power+Systems/page/nstress

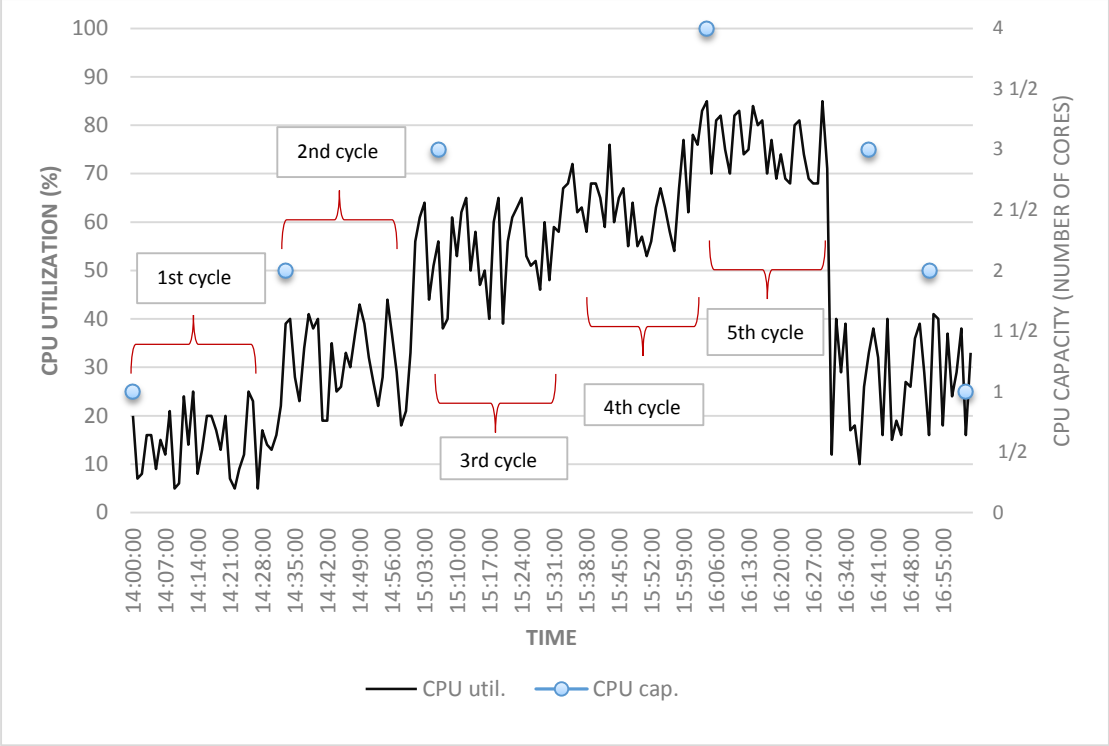


Figure 21. Virtual router VM CPU utilization and CPU capacity over time

4.6 Discussion

The experimental implementation provided a positive proof of concept of the suitability of the proposed model for describing the managed environment; and of Statistical Process Control as the basis for making online performance adjustments on a running VM.

4.6.1 Modeling the environment

The proposed information model provided a conceptual view of the managed environment, that attempts to unify (not replace) and extend existing instrumentation and management standards (SNMP, DMI, CMIP, etc.) using object-oriented constructs and design. Following the specifications of the parent CIM model, our proposal does not require any particular instrumentation or repository format: it allows unifying the data, using an object-oriented format, made available from any number of sources. This very object-oriented nature provides support for specific capabilities:

Abstraction and classification – To reduce the complexity of the problem domain, high level and fundamental concepts (the "objects" of the management domain) are defined. These objects are then grouped into types ("classes") by identifying common characteristics and features (properties), relationships (associations) and behavior (methods).

Object inheritance – By subclassing from the high level and fundamental objects, additional detail can be provided. A subclass "inherits" all the information (properties, methods and associations) defined for its higher level objects. Subclasses are created to put the right detail and complexity level at the right level in the model. This can be visualized as a triangle – where the top of the triangle is a "fundamental" object, and more detail and classes are defined as we move closer to the base.

Ability to depict dependencies, component and connection associations – Relationships between objects are extremely powerful concepts. The object paradigm offers an elegant approach in that relationships and associations are directly modeled. In addition, the way that relationships are named and defined describe the semantics of the object associations. Further semantics and information can be provided in association properties (specifying common characteristics and features). To illustrate this, consider a troubleshooting scenario where the infrastructure administrator discovers that one virtual network card in a virtual machine is at a high, bottleneck utilization. Investigating further (by traversing the proposed model's associations and exploring additional subclasses and data objects), the administrator finds that the hypervisor-provisioned port of that card has a very high traffic rate, and its traffic is related to a particular application. Again, following associations, the administrator can determine the "owner" of the System currently attached to the network port. If the owner cannot be reached, the administrator can use a standard method (instrumented in the KF provider) to "*add another network port*" to the VM and balance the traffic, thereby returning the hub to normal operating levels and restoring proper service operation.

Standard, inheritable methods – The ability to define standard object behavior (methods) is another form of abstraction. Bundling standard methods with an object's data is encapsulation. From a management perspective, it is crucial that an

infrastructure administrator can issue a command on a hardware component (e.g., a "reset" command), without the requirement to know the specifics of the component.

4.6.2 Managing the environment

The experimental implementation provided a positive proof of concept of the suitability of Statistical Process Control as the basis for taking local management decisions and making online performance adjustments on a running VM. During the managed operation for the first test experiment, the VM's CPU allocation was modified seven times as a result of the KF controller's action based on the interpretation of the live feed of performance data. During the second experiment, six respective resource modifications took place. SPC control charting is depended on the handling of the control points produced during process monitoring. There are various reasons that lead to control points that fall outside the calculated control limits – not all cases, however, indicate a problem in the monitored process. What is of importance is the sustained appearance of such points. This is reflected by the use of a decision threshold which determines, for the workload and infrastructure used, which control point repetition frequency is deemed as problematic.

The ability to dynamically allocate hardware resources is entirely platform-depended and relies on the characteristics of the hypervisor used. Whichever the virtualization platform, a certain amount hypervisor inertia is always present, affecting the end-to-end time needed for hardware reconfigurations to take effect [142][41]. Usually, memory operations take longer time than the respective ones on the platforms' CPU. To illustrate this problem, referring to the first experiment, had we initiated such action for every performance point that observed outside the UCL and LCL control limits during the test run, eighty-nine actions would have been needed. This would have resulted in a hypothetical need of a VM of sixty-two CPU (not to mention the duration overhead due to the enormous number of reconfigurations) [142]. Clearly, this would be an exaggeration and a failure of the proposed method, given that the acceptable process output is obtained with only ten CPUs. Managing the VM with our controller did produce a related, unanticipated outcome: that of bringing the system's performance within acceptable limits with a CPU capacity lower than that of the initial VM setup. It became apparent that the initial configuration was greater than

that required for servicing the workload within the requested time frame. Therefore, SPC can also be potentially used for capacity planning and profiling for any given workload.

For our approach to have an effective end result, as with other dynamic resource reconfiguration methods, it is necessary that any application running on the managed system be able to handle hardware changes. Such an application must be able to adapt to configuration changes; otherwise it will either remain unaffected or suffer performance degradation, failing to notice resources added or removed and scale accordingly. The platform's operating system and hypervisor play a crucial role in this direction by properly managing the allocated resources. We used a UNIX infrastructure in our experimental implementation [36] along with a well-parallelized multi-threaded application [37], where the dispatching of active processes was distributed relatively uniformly amongst all configured processors.

Despite the fact that our architecture focused on the CPU as the target core resource, it is possible for the KF controller to manage any resource provisioned by the hypervisor, as long as dynamic operations are supported on the resource. In modern hypervisors [41], memory and networking attributes can be managed dynamically. Given communications and computing convergence to a common operational entity, the core networking support in server virtualization environments is based on the IEEE 802.1Q VLAN implementation, where virtual network segments are established on top of physical switches. The latter are provided by the server's hardware features (the hypervisor works as a virtual Ethernet switch and supports queues for each VLAN in the system's memory). In this way, it is possible to establish network communication across different virtual servers, implementing virtual Ethernet adapters without routing network traffic outside the physical system that hosts them, performing the virtualization [41]. Depending on the resource managed and the platform used, a different cumulative threshold might be necessary for indicating sustained bad operation (as opposed to the three consecutive readings used in our CPU-focused implementation). Identifying the optimal value is an inherent step in SPC analysis of the process, so that proper control and operational parameters are determined.

During test runs VM CPU utilization was monitored four times every minute and performance data were transmitted to the resource controller. There is no ubiquitous rule on the sampling frequency. What is of importance is to ensure that the sampling used does capture any particular characteristics of the workload running on the VM. If, for example, there exist special processing stages that overcommit CPU or influence process output in any way, these must be included in the data collected. Hence, the sampling frequency is workload-dependent and should be determined in such way so as not to introduce any bias [181]. Generally, during the initial stages of process control, sampling occurs more often, decreasing once process stability has been ensured.

4.6.3 Metrics

For defining the auto-scaling triggering we concentrated on hardware level metrics, in particular, on system CPU utilization as this is one of the key metrics involved in systems performance investigation as well as in related cost/benefit decisions in modern datacenters [142][65][144]. In the experimental platform, CPU utilization was manifested as an aggregated percentage figure for the total online virtual processors allocated to the logical partitions by the IBM Power hypervisor. SPC processing of aggregated CPU utilization determines the thresholds (upper and lower control limits) that dictate when a resource allocation must be considered. Finer control of the environment or specialized case handling can be achieved by monitoring specific hardware utilization factors, such as the number of processor context switches, the number of threads awaiting on the processor's run queue, the memory or network interface usage and other fine-grain details [174].

The use of the metric was combined with a decision threshold variable indicating the number of consecutive occurrences at which the upper or lower control limits were exceeded. Determining a suitable decision threshold involves examining the workload and the virtualization platform characteristics. The ability to dynamically allocate hardware resources is entirely platform-depended, in particular, on the hypervisor used. Whichever the virtualization platform, a certain amount of hypervisor inertia is always present, affecting the end-to-end time needed for hardware reconfigurations to

take effect [142][41]. This delay may, also, include execution time consumed by intermediate systems that enable the hypervisor operations, such as the Hardware Management Console system in the technologies used in the experimental implementation. We refer to the time taken for a hardware reconfiguration task to complete as the adaptation time. Furthermore, each reconfiguration operation *itself* needs CPU time and the higher the *overall* load imposed on the CPU, the longer it takes for reconfigurations to complete. An optimum threshold must, therefore, be determined in order to avoid a negative effect due to overcommitting reconfiguration actions. In our infrastructure, the adaptation time for each CPU reconfiguration action was approximately 10 seconds. During the experiments, the decision to initiate a CPU reconfiguration action after three consecutive utilization readings (decision threshold) was proved to be the optimum for the infrastructure and workload used in our implementation.

4.6.4 Limitations of the proposed approach

SPC methods require the monitored data to be identically distributed. The standard Shewhart analysis of individual measurements produces control charts based on the Central Limit Theorem, which presupposes that the sampling distribution follows the normal distribution. This is workload-dependent and can be applied to VM utilizations with symmetric population distribution. Therefore, in the approach presented in this section, the VM's performance is assumed to exhibit a constant mean and the workload used in the experimental implementation did produce such a performance pattern. CPU (or other resource) utilization that does not present these characteristics may not be able to be handled by the standard Shewhart approach. In these cases, other types of SPC tools could be applicable, such as the IX-MR [177], MCUSUM, MEWMA and multivariate Shewhart control charts [183]. Control charts, regardless of their type, can very quickly indicate when the quality of the measured process has changed. They do not, however, indicate the direction and the magnitude of required corrective measures. Hence, additional effort needs to be put on determining the nature of the latter and these measures are workload and infrastructure dependent. Finally, control charts are not entirely error-free and may, in cases, indicate that the process is out of control as a result of a special cause variation,

whereas no issue actually exists. It is possible, therefore, for false alarms to be issued resulting in unnecessary management actions and possible unwanted overhead.

The characteristics of the virtualization platform form another factor that affects the applicability of our approach. The CPU allocations (cf. *Figure 20. VM CPU capacity after each management action,* and *Figure 21. Virtual router VM CPU utilization and CPU capacity over time*) were the results of management actions following the primary metric analysis. One issue that needs to be taken into account is the time taken for the hypervisor to complete a reconfiguration request. This depends on a number of factors that pertain to the hardware characteristics of the virtualization platform. On the deployed platform the dispatch latency of a virtual processor depends on the partition CPU entitlement and the number of virtual processors that are online in the virtual partition. The capacity entitlement is equally divided among these online virtual processors. Consequently, the number of online virtual processors impacts the length of each virtual processor's dispatch [68]. For the hypervisor used, the worst dispatch latency is 18 milliseconds, since the minimum dispatch cycle supported at the virtual processor level is one millisecond. This latency is based on the minimum partition entitlement of 1/10 of a physical processor and the 10 millisecond rotation period of the hypervisor's dispatch wheel [184][41]. Similar logic applies to any other hypervisor and CPU offered by other vendors and, depending on the workload or Cloud service characteristics, this overall CPU allocation hypervisor overhead has to be taken into account. Delays in this reconfiguration may lead to instability in the infrastructure's operation as, by the time the CPU schema has been reconfigured, the workload may have changed demanding different handling. Furthermore, as discussed, this hypervisor inertia requires an initial learning effort and overhead before the method is tuned for application on the platform of choice.

4.7 Chapter Summary

In this chapter, we demonstrated the verification approach followed for asserting the correctness of the proposed information model. We drew on one of the key issues of interest in the Cloud computing paradigm – that of resource allocation – and applied our model on an typical architecture as found in Cloud datacenters. This verification

is vital as, apart from assuring the feasibility of the proposal, it is performed on actual, real-life, infrastructure and configuration. Thus, the results yielded are not only of theoretical nature, but of practical importance. The resource management method proposed and applied is based on Statistics and, to the best of our knowledge, there is no equivalent that is applied to Cloud systems. The efficiency of our technique was evaluated via a number of simulations. The results showed that our approach is feasible for virtualized infrastructures, embodying system and networking components. We conclude that our proposed technique can be applied as the foundation of a management method for such infrastructures. This renders the proposed information model an efficient approach to supporting end-to-end management of hypervisor-based virtual network environments, prominent nowadays in Cloud datacenters.

Chapter 5 – Conclusions

In this dissertation, we have investigated the use of information models in the context of hypervisor-based virtual network environments, such as in the Cloud computing paradigm. In particular, introductory information and related work are presented in Chapters 1- 2. Chapter 3 presented our proposed model, and Chapter 4 was concerned with the verification of the appropriateness and applicability of the model in actual infrastructures. In this last chapter, we summarize the contribution of this dissertation and we outline certain topics left for future work.

5.1 Summary of the contributions

Virtual networks have recently emerged to increasingly rely on computing servers and their hypervisors to allow for new network services without worrying about the specific details of the underlying (physical) infrastructure. Through system virtualization, multiple virtual networks can flexibly operate over the same physical infrastructure. In the context of virtual networking, a hypervisor provides system resources and monitors the operational virtual networks, reallocating resources accordingly as per the demands of the workload. The efficient and reliable operation of these virtual networks requires effective management of the environment. This involves accurate information modeling, able to abstract the specifics of the virtualized system and networking infrastructure, along with sophisticated resource allocation algorithms for assigning the physical system resources to the virtual networks. The contributions of this dissertation are multi-fold:

A new information model for managing host-based virtual networks. Recent technological advancements, where virtualized computing plays an integral part in computer networks, complicate the network's architecture, operation and management, introducing new aspects that need to be considered for proper end-to-end service delivery. We proposed the KF model, a Common Information Model based approach, showing that standardization of virtual network representation where computing servers are involved is, indeed, feasible. The model allows for the conceptual representation of involved components and for the introduction of targeted actions against them. The proposed model builds upon the common ground of one of

the main information model standards, to characterize both logical artifacts and the runtime environment managed. Focusing on networking resources implemented at the hypervisor layer, the model abstracts the hypervisor virtual Ethernet switch (consistent with IEEE 802.1 Q standard): the objects and attributes that enable the operation of virtual LAN (VLAN) bridges that permit the definition, operation, and administration of virtual LAN topologies within a bridged LAN infrastructure. The model not only allows the representation of various environment configurations but, additionally, allows validation of the managed resources state. Contrary to the capabilities of other proposals, the KF model efficiently represents hypervisor-based infrastructures and the hypervisor per-se, distinctively and accurately. This is not possible via the use of other available information models and related methods.

A verification of the model. To verify the applicability of our proposal we applied the model on an actual environment; more specifically, on a typical virtual network architecture, as found in Cloud datacenters, running real workloads. We demonstrated the extensibility characteristics and the applicability of the model by modifying its provider (controller) so as to implement a statistical approach to resource management, allowing for dynamic modification of a virtual machine's resource allocation, following continuous statistical evaluation of the observed performance. This approach allows real-time management of the VNE's computing server allocated resources. This is an essential feature, especially in complex datacenter implementations.

A new approach to dynamic resource management. It is widely accepted in the IT industry that effective infrastructure resource management is of vital importance in successfully delivering services to users and other interested parties alike. Infrastructure resources need to be efficiently allocated in order to support varying application workload demands. Failing to do so may result in erratic infrastructure operation, hence improper service or excessive financial costs. This is more prominent in modern datacenters where the architecture complexity increases with the use of server virtualization to support building and operating computer networks. In this dissertation, as part of the model verification, we introduced a new approach to dynamic resource management, based on a set of statistical concepts and tools, namely Statistical Process Control. Our approach allows for the online, dynamic

management of the CPU capacity of a virtual machine, taking into account the particular characteristics of the application workload running on it. We demonstrated the method on an experimental setup using real instead of synthetic data, so as to provide a proof of concept of the method for use in actual datacenter environments. Our proposal efficiently manages the CPU resources of a virtual machine, adapting to changes in the workload patterns and providing for capacity planning and optimization of the initial resource allocation. The method can be extended to manage other type of resources such as hypervisor-provisioned system memory, virtual switch network bandwidth, etc.

5.2 Future Work

VNE holistic model. Advances in VNE related technology and research, as well as trends in business use and application, form a dynamic environment which can provide room for various areas of further research. It is evident that a VNE *holistic* model will need to incorporate elements from the networking, virtualization and business worlds. A virtual network environment, as discussed in previous Chapters of this dissertation, can be a very large and complex entity. In addition to the purely technical elements, non-technical aspects (such as operations and monitoring, personnel roles and skills) come into play in the virtual network picture and need to be considered and accounted for in the model. Specific business contexts and their parameters need to be addressed within the intended use and scope of a VNE model, such as:

- The different type of businesses and service models involved in VNE application (ISPs, backbone providers, cloud computing providers, hardware and software provisioning, etc.).
- The probability of financial gain by the existence of a common model and, respectively, financial loss by not applying a common standard.
- The very need for a common model in a VNE business context.

Overall, it is possible that a single, extensible, *holistic* model can neither be constructed nor that the total investment required is justified by the expected advantages and outcomes of having such a model. Hence, *the scope and creation feasibility* of a holistic model need to be determined. These represent another

important research question which should be addressed, either before, or in parallel with, the model creation effort. A relevant topic that needs further research is the determination of the commonality level in VNE involved components, thus making the creation of a common model possible.

Modeling of the virtualization layer. Aspects of systems virtualization and physical/logical server's resources and characteristics, as discussed, cannot be ignored as they influence network operation and, ultimately, the quality of the provided service [38]. Hypervisors [65] do impose a new level of virtualization by creating a secondary address space, intervening between network applications and physical infrastructure layers [41][10]. The modeling side of the hypervisor layer needs to be further investigated. Additionally, complete virtualization of the network address space is still a relatively new field, influenced by the use of hypervisors in the network infrastructure stack and by the IPv6 evolution.

Statistical Process Control-based dynamic resource management. The method introduced and illustrated in this dissertation was verified as appropriate for typical workloads, namely those that exhibit symmetric utilization patterns. Our model and provider, in its current form, cannot cope with asymmetric infrastructure utilization. Erratic patterns of resource consumption needs can be addressed by approaches to control charting, other than the standard Shewhart Charts. Incorporating such approaches will enhance the provider and enable the creation of a basis for the end-to-end management of a datacenter computing resource, adhering to actual SLA-specified constraints.

5.3 Closing remarks

This work has appeared, or is currently under review, in several journals and conference papers (cf. Appendix A). In short, our proposed model is described in [19], and its usage as a datacenter management option has been introduced in [185]. Our proposed model's positioning in the information modeling realm of Computing and Networking is illustrated in [186]. The verification methodology is described in [187], and [188]. Lastly, our approach's implementation in the datacenter context is further explored in [189].

Appendix A – Publications

In Press

Refereed papers in international journals

i. D. Kontoudis and P. Fouliras,
"**A survey of models for computer networks management**",
International Journal of Computer Networks & Communications (IJCNC),
Vol.6, No.3, pp. 157-176, 2014, ISSN: 0974-9322 (online), 0975-2293 (print),
DOI: 10.5121/ijcnc.2014.6313.

Abstract: *The virtualization concept along with its underlying technologies has been warmly adopted in many fields of computer science. In this direction, network virtualization research has presented considerable results. In a parallel development, the convergence of two distinct worlds, communications and computing, has increased the use of computing server resources (virtual machines and hypervisors acting as active network elements) in network implementations. As a result, the level of detail and complexity in such architectures has increased and new challenges need to be taken into account for effective network management. Information and data models facilitate infrastructure representation and management and have been used extensively in that direction. In this paper we survey available modeling approaches and discuss how these can be used in the virtual machine (host) based computer network landscape; we present a qualitative analysis of the current state-of-the-art and offer a set of recommendations on adopting any particular method.*

ii. D. Kontoudis and P. Fouliras,
"**Host-based Virtual Networks Management in Cloud Datacenters**",
accepted for publication in Computing and Informatics, to appear in 2017.
Impact Factor: 0.504, ISSN: 1335-9150, http://www.cai.sk/ojs/index.php/cai/index

Abstract: *Infrastructure management is of key importance in a wide array of computer and network environments. In Cloud datacenters, the use of virtualization has driven the convergence of communications and computing to a common operational entity. Failure to effectively manage the involved infrastructure results in impediments in successful service provisioning. Information models facilitate infrastructure management and current solutions can effectively be applied in most datacenter scenarios except in cases where the networking architecture heavily relies on systems virtualization. In this paper we propose an information model for managing virtual network architectures where hypervisors and computing server resources are deployed as the basis of the networking layer. We provide a successful proof of concept by managing, using statistical methods, a virtual machine-based network infrastructure acting as an IP routing platform. Our proposal enables the dynamic reconfiguration of allocated infrastructure resources adapting, in real-time, to variations in the imposed workload.*

Refereed papers in proceedings of international conferences

iii. D. Kontoudis and P. Fouliras, "**A CIM-based approach for managing computing servers and hypervisors acting as active network elements**", In Proceedings of the International Federation for Information Processing (IFIP) 7th HET-NETs conference NET-PEN 2013 Networks and Performance Engineering, Bradford, United Kingdom, Mon 11th-Wed 13th Nov. 2013, River Publishers, ISBN: 978-87-93102-32-3.

Abstract: *Communication network implementations present an ever increasing use of computing servers and hypervisors assuming the role of active network elements. This fact, along with the virtualization concept which has, also, been adopted in the computer networks field, introduces new challenges that need to be accounted for when managing such networks. In this paper we propose a new modular and extensible information model which can be applied to most scenarios of network architectures where hypervisors are involved, facilitating the efficient representation and management of the provisioned computing server resources.*

iv. D. Kontoudis and P. Fouliras, "**Modeling and managing virtual network environments**", In Proceedings of the 17th Panhellenic Conference on Informatics, PCI'13, 19-21 September 2013, pp. 39-46, ACM, New York, USA, 2013, ISBN: 978-1-4503-1969-0, DOI: 10.1145/2491845.2491846.

Abstract: *While the virtualization concept and its underlying technologies have been adopted in the computer networks field, as in many other Information Technology areas, a generic model of network virtualization environments (NVEs) has not, yet, been proposed. This lack of conceptual impression imposes limitations on the representation and management of these environments. In this paper we propose a new modular and extensible conceptual model, based on Distributed Management Task Force's Common Information Model (DMTF CIM), which can be applied to most scenarios of NVE architectures. We, also, present a test case where the model is extended to include Statistical Process Control methods for guaranteed network performance delivery. In this example, an NVE spans different virtual server nodes, which provide its required resources, and this architecture is modeled and managed resulting in a mechanism to assess and control the overall resources consumption.*

v. D. Kontoudis, P. Fouliras and C. Lambrinoudakis, "**Statistics-driven Datacenter Resources Provisioning**", In Proceedings of the 19th Panhellenic Conference on Informatics, PCI'15, 1-3 October 2015, ACM, New York, USA, pp. 185-190, ISBN: 978-1-4503-3551-5, DOI=10.1145/2801948.2801971.

Abstract: *The virtualization concept along with its underlying technologies has been warmly adopted in many fields of computer science. In modern datacenters, the convergence of communications and computing to a common design and operational entity has proved an inevitable reality, introducing virtual servers as active network elements thus increasing the infrastructure complexity. Effective resource management in such architectures is crucial, impacting both service delivery and the resulting infrastructure operating costs. We propose a novel approach, based on Statistical Process Control, for the dynamic resource provisioning of datacenter virtualized resources. Our work provides an integrated, platform-independent and technology-neutral, framework consisting of a CIM-based resource controller which allows for the on-line, adaptive, management of a virtual machine's CPU allocation. The controller can be extended to manage other types of hypervisor-provisioned resources. We provide a successful proof-of-concept of our work, deploying the controller on an IBM pSeries UNIX system running a core banking environment software with actual business data. Given the platform agnostic description of the controller and the managed environment it is, thus, possible to easily provide different operating system and hardware platform ports.*

In review

Refereed papers in international journals

vi. D. Kontoudis and P. Fouliras,
"**A Statistical Approach to Virtualized Server Resource Management**",
under review in *Concurrency and Computation, Practice and Experience*,
Impact Factor: 0.997, Wiley, http://onlinelibrary.wiley.com/journal/10.1002/(ISSN)1532-0634

Abstract: *Resource management is of key importance in a wide array of computer and network environments. Failure to effectively allocate the resources necessary for smooth infrastructure operation may result in impediments in successful service provisioning. The mathematical science of Statistics offers an extensive theoretical and practical toolbox that can be of potential use in tackling such problems in the Information Technology domain. We propose a novel statistical approach for resource management of virtualized data center components. We have worked towards creating a mechanism that allows for dynamic allocation of the involved resources, constantly adapting to their changing operation patterns. Our system incorporates a resource controller, based on Statistical Process Control, which permits the online management of a virtual machine's CPU allocation through the real-time analysis of its observed performance. We successfully demonstrate our approach, with real and not synthetic data, on an architecture running the core banking environment of a financial institution. The controller successfully manages the CPU resource allocation of the involved virtual machine and stabilizes its performance while actual business transactions are processed. Our approach can be extended to realize different workload models and manage other types of hypervisor-provisioned resources*

Appendix B – UML Schemas

1. KF VNE information model

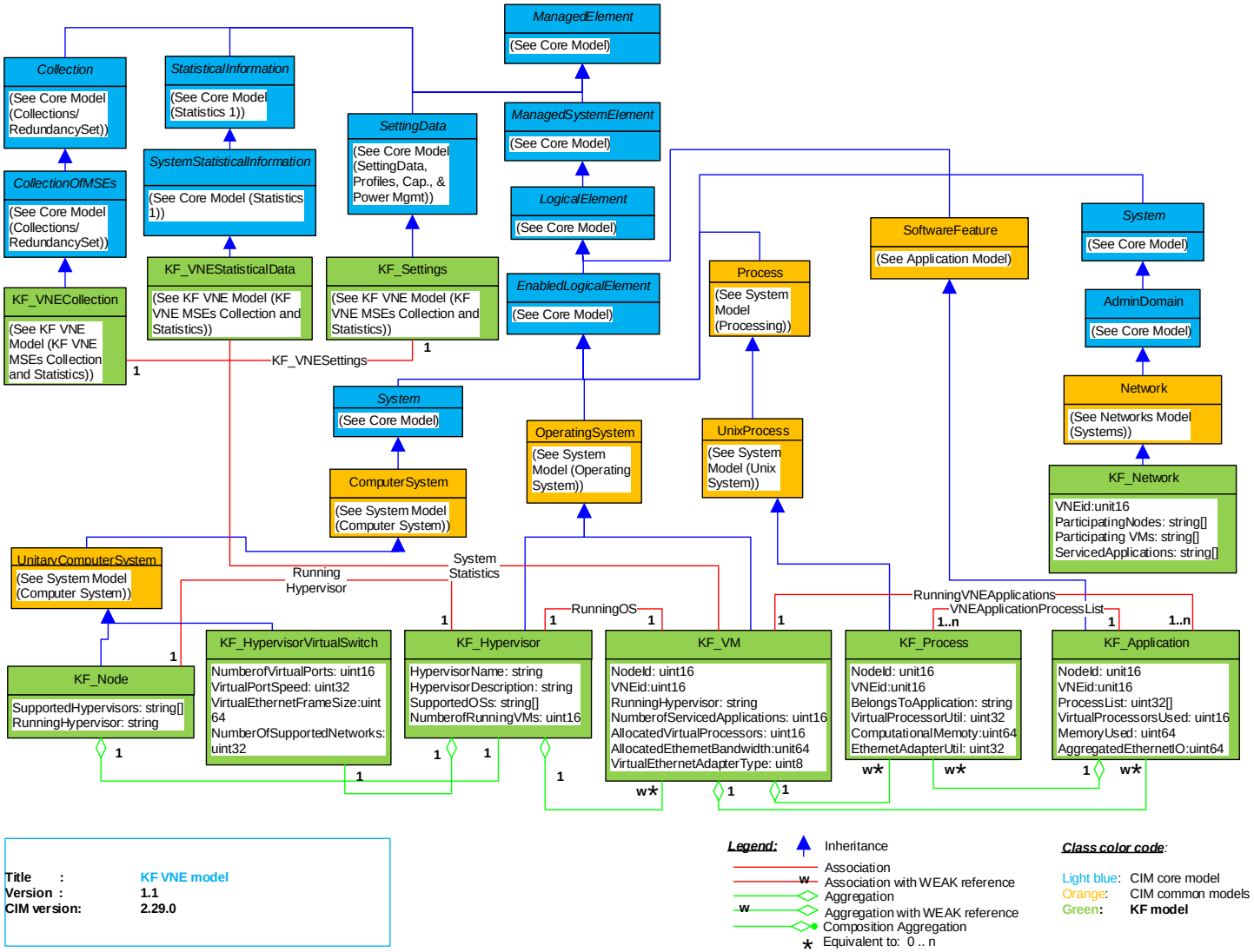


Figure 22. KF VNE information model

2. KF VNE MSEs collection and statistics

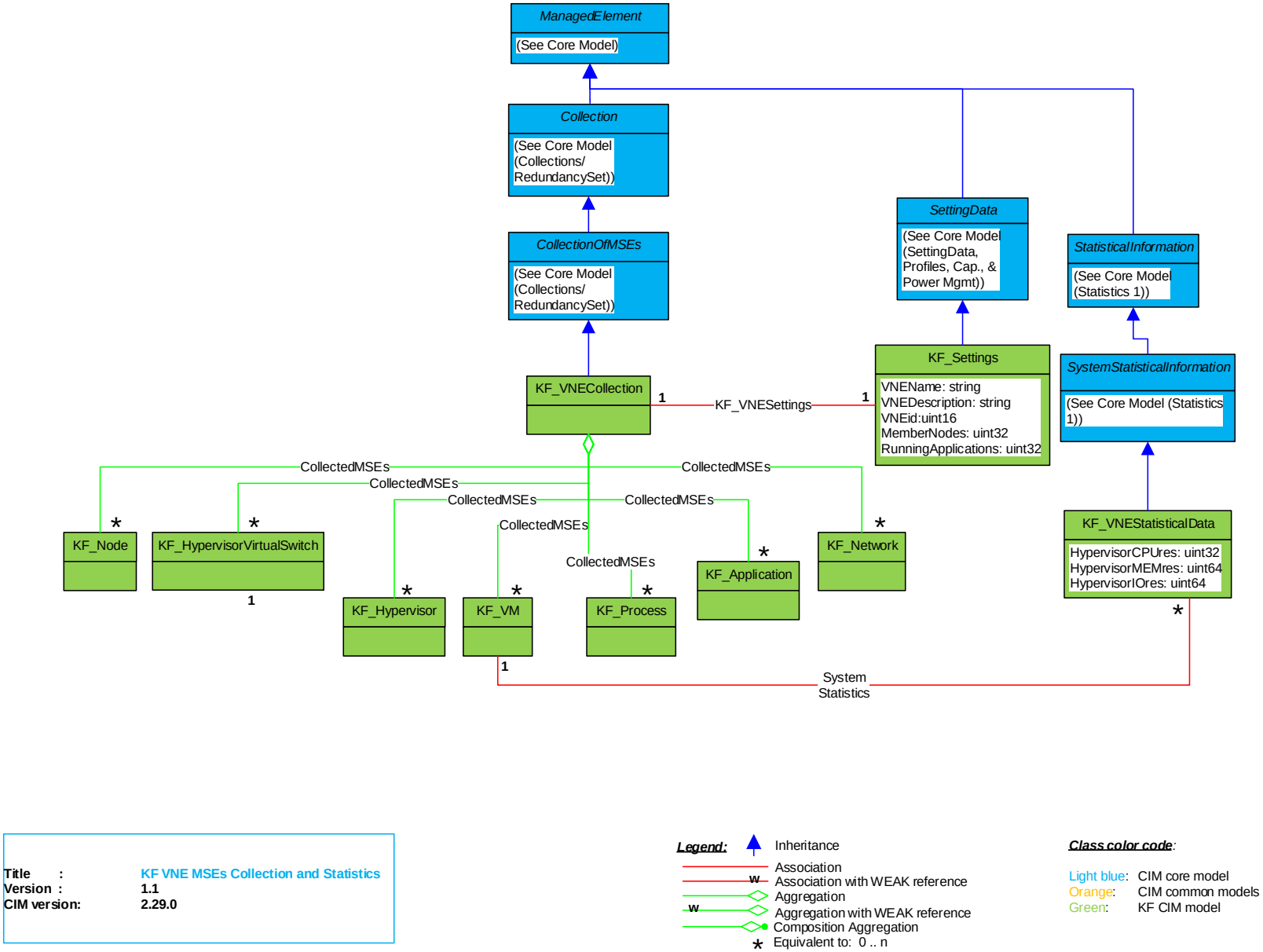
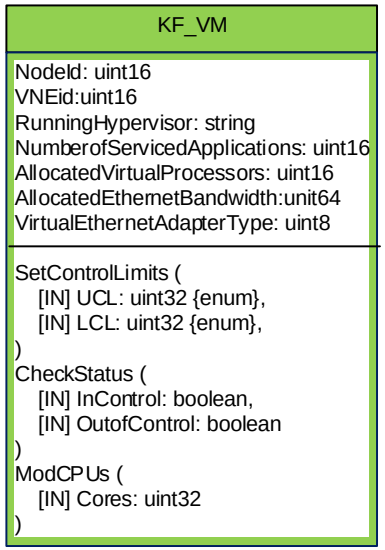
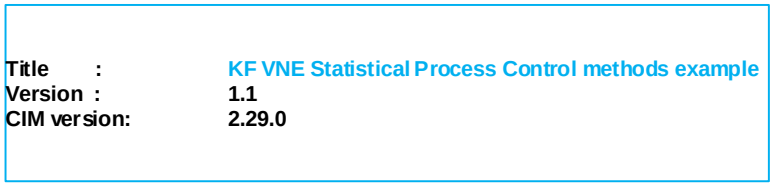
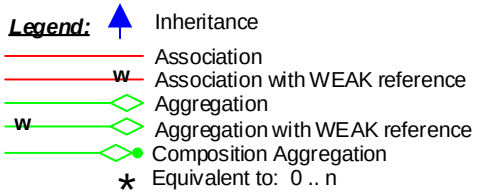


Figure 23. KF VNE MSEs collection and statistics







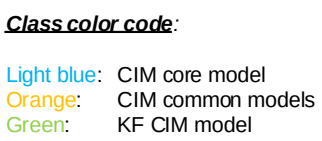


Figure 24. KF VNE Statistical Process Control method

Appendix C – MOF descriptions

1. KFcore.mof

```
// KFcore.mof, v1.1
// CIM version 2.29.0
//
// Copyright (C) 2016 Dimitris Kontoudis, Panayiotis Fouliras.
//
// ===========================================================
//  KF VNE Model
// ===========================================================
//
// KF model core classes

[Version ( "1.0" ),
    Description (
      "The KF_Node class extends UnitaryComputerSystem to abstract an VNE node,"
      "i.e. a physical system consisting of hardware and low-level (bios) software" )]
class KF_Node : CIM_UnitaryComputerSystem {
    [Description (
        "An array of free-form strings containing the supported hypervisors on this particular hardware node"),
            ValueMap { "IBM Power Hypervisor", "Xen", "Citrix XenServer", "Oracle VM", "KVM", "Sun Logical Domains",
                                        "VMware ESX Server", "Microsoft Hyperv-V", "L4 Microkernels", "Integrity Padded Cell", "Trango",
                                        "VirtualLogix VLX", "Hitachi Virtage", "Vmware Server", "QEMU", "Sun VirtualBox", "Other" },
            ArrayType ( "Indexed" ) ]
  string SupportedHypervisors[];
    [Description (
        "A string that describes the active hypervisor running on the physical node"),
            ValueMap { "IBM Power Hypervisor", "Xen", "Citrix XenServer", "Oracle VM", "KVM", "Sun Logical Domains",
                                        "VMware ESX Server", "Microsoft Hyperv-V", "L4 Microkernels", "Integrity Padded Cell", "Trango",
                                        "VirtualLogix VLX", "Hitachi Virtage", "Vmware Server", "QEMU", "Sun VirtualBox", "Other" }]
```

```
  string RunningHypervisor;
};

[Version ( "1.0" ),
    Description (
      "The KF_HypervisorVirtualSwitch class extends UnitaryComputerSystem to abstract,"
      "the IEEE 802.1Q VLAN compatible virtual switch implemented by the node's hypervisor" )]
class KF_HypervisorVirtualSwitch : CIM_UnitaryComputerSystem {
    [Description (
        "The maximum number of virtual ethernet ports supported by the virtual switch" )]
  uint16 NumberofVirtualPorts;
    [Description (
        "The supported port speed in Gbps" )]
  uint32 VirtualPortSpeed;
    [Description (
        "The maximum legal Ethernet frame size, in bytes, for tagged frames" )]
  uint16 VirtualEthernetFrameSize;
     [Description (
        "The maximum number of networks that can be instantiated on the virtual switch" )]
  uint32 NumberofSupportednetworks;
};

[Version ( "1.0" ),
    Description (
      "The KF_Hypervisor class extends OperatingSystem to abstract,"
      "a special purpose, low level (firmware/bios), software code "
              "that manages the physical hardware and allows for the operation"
              "of different operating system instances on that hardware" )]
class KF_Hypervisor : CIM_OperatingSystem {
    [Description (
        "The name of the running hypervisor" )]
  string HypervisorName;
    [Description (
        "A more detailed description of the running hypervisor, including"
```

```
                    "its version number and other operational characteristics" )]
  string HypervisorDescription;
   [Description (
      "An array of free-form strings containing the Operating Systems that can be run (are supported) on this hypervisor"),
           ValueMap { "IBM AIX", "Oracle SUN Solaris", "Linux", "Microsoft Windows" },
           ArrayType ( "Indexed" ) ]
  string SupportedOSs[];
    [Description (
      "The number of virtual machines (i.e. operating system instances) that are active on this hypervisor" )]
  uint16 NumberofRunningVMs;
};

[Version ( "1.0" ),
    Description (
     "The KF_VM class extends OperatingSystem to abstract a particular virtual"
     "machine (i.e. operating system) instance " )]
class KF_VM : CIM_OperatingSystem {
    [Description (
      "A unique number identifying this particular virtual machine" )]
  uint16 NodeId;
   [Description (
      "A unique number identifying the Network Virtualization Environment serviced by this virtual machine" )]
  uint16 NVEid;
   [Description (
      "The hypervisor hosting the virtual machine" )]
  string RunningHypervisor;
   [Description (
      "The number of applications (i.e. software code) active on this virtual machine" )]
  uint16 NumberofServicedApplications;
   [Description (
      "The number of processor cores (virtual, since the environment is run off a hypervisor)"
      "to this virtual machine" )]
  uint16 AllocatedVirtualProcessors;
   [Description (
```

```
    "The aggregated Ethernet bandwidth, in Gbps, of the virtual ethernet adapater allocated to"
    "this virtual machine" )]
uint64 AllocatedEthernetBandwidth;
  [Description (
    "The virtual ethernet adapter type. (0) for hypervisor virtual switch type, (1) for"
                  "hardware network cards allocated to the physical node and shared to the virtual machines",
        ValueMap { "0", "1" },
        Values { "HYP-based", "HW_Nic-based" } )]
uint8 VirtualEthernetAdapterType;
  [Description (
    "This method set the Upper and Lower Control Limits, as per the Statistical Process Control"
    "context, in order to establish the accepted boundaries of proper VM running performance. The"
                  "boundaries refer to CPU utilization." )]
uint32 KF_SPC_SetControlLimits(
                  [IN, Description (
      "The desired CPU utilization Upper Control Limit" )]
  uint32 KF_SPC_UCL,
    [IN, Description (
      "The desired CPU utilization Lower Control Limit" )]
  uint32 KF_SPC_LCL
        );
  [Description (
    "This method checks whether the VM operates within the specified CPU performance parameters,"
     "i.e. UCL and LCL. Two boolean variables report respective true/false status." )]
uint32 KF_SPC_CheckStatus(
    [IN (false), OUT, Description (
      "Boolean indicating if VM is running In Control, i.e. within control limits." )]
  boolean KF_SPC_InControl,
    [IN (false), OUT, Description (
      "Boolean indicating if VM is running Out of Control, i.e. outside control limits." )]
  boolean KF_SPC_OutofControl
        );
  [Description (
    "This method modifies the VMs CPU running capacity (i.e. active cores) so as to keep the performance"
```

```
        "withing acceptable control limits (i.e. in control)." )]
  uint32 KF_ModCPU(
                          [IN, Description (
          "The desired active CPU cores" )]
    uint16 KF_SPC_Cores
              );
};

[Version ( "1.0" ),
    Description (
      "The KF_Process class extends UnixProcess to abstract a process/thread (i.e. program) running on a Unix virtual"
      "machine instance. This process is bound to a specific VM and is NOT parallelized across different nodes."
              "Utilization metrics reflect quantities since the beginning of any monitoring process" )]
class KF_Process : CIM_UnixProcess {
    [Description (
        "A unique number identifying this particular virtual machine" )]
  uint16 NodeId;
    [Description (
        "A unique number identifying the Network Virtualization Environment serviced by this virtual machine" )]
  uint16 NVEid;
    [Description (
        "The networking, or other, application that uses this particular process" )]
  string BelongsToApplication;
    [Description (
        "The virtual machine processor utilization (in percentage) for this process" )]
  uint32 VirtualProcessorUtil;
    [Description (
        "The virtual machine real memory portion (in bytes) occupied by the process" )]
  uint64 ComputationalMemory;
    [Description (
        "The aggregated Ethernet adapter utilization (in percentage) consumed by the process" )]
  uint32 EthernetAdapterUtil;
};
```

```
[Version ( "1.0" ),
    Description (
      "The KF_Application class extends SoftwareFeature to abstract a networking or other application "
              "(can consist of a single process/programm or a collective of them) running on a Unix virtual"
      "machine instance." )]
class KF_Application : CIM_SoftwareFeature {
    [Description (
        "A unique number identifying this particular virtual machine" )]
  uint16 NodeId;
    [Description (
        "A unique number identifying the Network Virtualization Environment serviced by this virtual machine" )]
  uint16 NVEid;
    [Description (
        "An array of free-form numbers containing all Unix processes running on a virtual machine, comprising"
                        "the particular application. The value map contains the Unix process id of each process.",
                        ArrayType ( "Indexed" ) )]
  uint32 ProcessList[];
    [Description (
        "The total sum of virtual processors used for this application" )]
  uint16 VirtualProcessorsUsed;
    [Description (
        "The total sum of computational memory used for this application" )]
  uint64 MemoryUsed;
    [Description (
        "The aggregated Ethernet adapter utilization (in percentage) consumed by the process" )]
  uint64 AggregatedEthernetIO;
};

[Version ( "1.0" ),
    Description (
      "The KF_Network class extends Network to abstract a virtual network spanning different "
              "server nodes which runn a flavor of the Unix operating system" )]
class KF_Network : CIM_Network {
    [Description (
```

```
      "A unique number identifying this particular virtual machine" )]
  uint16 NodeId;
   [Description (
      "A unique number identifying the Network Virtualization Environment serviced by this virtual machine" )]
  uint16 NVEid;
   [Description (
      "An array of free-form strings containing the descriptions of the physical nodes on which this virtual network runs" ),
                  ArrayType ( "Indexed" ) ]
  string ParticipatingNodes[];
   [Description (
      "An array of free-form strings containing the of virtual machines nodes on which this virtual network runs" ),
                  ArrayType ( "Indexed" ) ]
  string ParticipatingVMs[];
   [Description (
      "An array of free-form strings containing the applications running on this virtual network" ),
                  ArrayType ( "Indexed" )]
  string ServicedApplications[];
};
```

2. KFassociations.mof

```
// KFassociations.mof, v1.1
// CIM version 2.29.0
//
// Copyright (C) 2016 Dimitris Kontoudis, Panayiotis Fouliras.
//
// ===================================================================
//  KF VNE Model Associations
// ===================================================================
//
// KF model association classes

[Association, Version("1.0"),
 Description(
   "KF_RunningHypervisor associates a hypevisor with the physical node (KF_Node) on which it runs" )]
class KF_RunningHypervisor : CIM_Dependency {
     [Override ( "Antecedent" ),
      Description ( "The Hypervisor" )]
  KF_Hypervisor REF Antecedent;
     [Override ( "Dependent" ),
      Description ( "The physical node that needs to be associated with the Hypervisor" )]
  KF_Node REF Dependent;
};

[Association, Version("1.0"),
 Description(
   "KF_RunningOS associates the operating system runnning on a virtual machine"
            "on top of a specific hypervisor" )]
class KF_RunningOS : CIM_Dependency {
     [Override ( "Antecedent" ),
      Description ( "The Operating System" )]
  KF_VM REF Antecedent;
```

```
    [Override ( "Dependent" ),
     Description ( "The Hypervisor that needs to be associated with the Operating System (VM)" )]
  KF_Hypervisor REF Dependent;
};

[Association, Version("1.0"),
 Description(
   "KF_RunningNVEApplications associates the network virtualization environment applications"
            "operating on a specific VM" )]
class KF_RunningNVEApplications : CIM_Dependency {
    [Override ( "Antecedent" ),
     Description ( "The application list" )]
  KF_Application REF Antecedent;
    [Override ( "Dependent" ),
     Description ( "The operating system (VM) that needs to be associated with the applications" )]
  KF_VM REF Dependent;
};

[Association, Version("1.0"),
 Description(
   "KF_NVEApplicationsProccessList associates a network virtualization environment's application"
            "operating on a specific VM to the specific process IDs of the processes comprising the application,"
            "as presented on the operating system's process table" )]
class KF_NVEApplicationsProccessList : CIM_Dependency {
    [Override ( "Antecedent" ),
     Description ( "The application" )]
  KF_Application REF Antecedent;
    [Override ( "Dependent" ),
     Description ( "The operating system (VM) processes that need to be associated with the application" )]
  KF_Process REF Dependent;
};

[Association, Version("1.0"),
 Description(
```

```
   "KF_SystemStatistics associates a VM's core operational statistics to the overall VNE"
   "statistical data class" )]
class KF_SystemStatistics : CIM_Dependency {
     [Override ( "Antecedent" ),
      Description ( "The VM's statistics" )]
  KF_VM REF Antecedent;
     [Override ( "Dependent" ),
      Description ( "The overall statistical data that need to be associated with the VM's statistics" )]
  KF_NVEStatisticalData REF Dependent;
};

[Association, Version("1.0"),
 Description(
   "KF_NVESettings associates a particular configuration set (KF_Settings) to a collection of"
            "managed system elements, i.e. to all parts comprising the VNE and modeled via KF (KF_NVECollection)."
            "This settings are, then, available and propagated to all VNE constituent parts" )]
class KF_NVESettings : CIM_Dependency {
     [Override ( "Antecedent" ),
      Description ( "The configuration set" )]
  KF_Settings REF Antecedent;
     [Override ( "Dependent" ),
      Description ( "The group of VNE members that need to be associated with the configuration set" )]
  KF_NVECollection REF Dependent;
};
```

3. KF_collections_and_statistics.mof

```
// KF_collection_and_statistics.mof, v1.1
// CIM version 2.29.0
//
// Copyright (C) 2016 Dimitris Kontoudis, Panayiotis Fouliras.
//
// ===========================================================
//  KF VNE Model MSEs Collection and Statistics
// ===========================================================
//
// KF model Collection and Statistics classes

[Version("1.0"),
 Description(
   "The KF_NVECollection is a named collection and logically groups the models core "
            "elements, i.e. the consituents of the VNE entity. In such way the VNE can be handled"
            "as a single entity and global elements, such as settings, can be propagated" )]
class KF_NVECollection : CIM_CollectionOfMSEs {
    [Key, Override ( "CollectionID" )]
  string CollectionID;
};

[Version("1.0"),
 Description(
   "The KF_NVEStatisticalData abstracts an instantiated VNE's core statistical information,"
            "i.e. since it is running on virtual machines, these statistics represent the aggregated"
            "quantities of core resources used (CPU, memory, I/O) )]
class KF_NVEStatisticalData : CIM_SystemStatisticalInformation {
   [Description(
      "The max number of CPU cores allocated for use by all VNE's VMs" )]
   uint32 HypervisorCPUres;
   [Description(
```

```
      "The max number of memory, in bytes, allocated for use on all VNE's VMs )]
   uint64 HypervisorMEMres;
            [Description(
      "The total I/O traffic, in bytes, of all VM's virtual ethernet adapters" )]
   uint64 HypervisorIOres;
};
```

4. KF_MSEs_aggregation_associations.mof

```
// KF_MSEs_aggregation_associations.mof, v1.1
// CIM version 2.29.0
//
// Copyright (C) 2016 Dimitris Kontoudis, Panayiotis Fouliras.
//
// ============================================================
//  KF VNE Model MSEs Aggregation Association classes
// ============================================================
//
// The following aggregations establishe the constituent parts of a VNE.
// Based on the current version of the KF model seven members are needed
// in order to semantically form a fully functional VNE.

  [Association, Aggregation, Version ( "2.6.0" ),
   Description (
      "The physical node." )]
class KF_NodeInNVE : CIM_CollectedMSEs {
     [Aggregate, Override ( "Collection" ),
      Max ( 1 ),
      Description (
         "The VNE collection that aggregates the members participating in it." )]
  KF_NVECollection REF Collection;
     [Override ( "Member" ),
      Min ( 1 ),
      Description ( "The physical nodes that participate in the VNE." )]
  KF_Node REF Member;
};

  [Association, Aggregation, Version ( "2.6.0" ),
   Description (
      "The hypervisor's virtual switch." )]
```

```
class KF_HypVSInNVE : CIM_CollectedMSEs {
    [Aggregate, Override ( "Collection" ),
     Max ( 1 ),
     Description (
       "The VNE collection that aggregates the members participating in it." )]
  KF_NVECollection REF Collection;
    [Override ( "Member" ),
     Min ( 1 ),
     Description ( "The hypervisor virtual switch of each physical node that forms the VNE." )]
  KF_HypervisorVirtualSwitch REF Member;
};

  [Association, Aggregation, Version ( "2.6.0" ),
   Description (
     "The physical node's hypervisor." )]
class KF_HypInNVE : CIM_CollectedMSEs {
    [Aggregate, Override ( "Collection" ),
     Max ( 1 ),
     Description (
       "The VNE collection that aggregates the members participating in it." )]
  KF_NVECollection REF Collection;
    [Override ( "Member" ),
     Min ( 1 ),
     Description ( "The hypervisor of each physical node that forms the VNE." )]
  KF_Hypervisor REF Member;
};

  [Association, Aggregation, Version ( "2.6.0" ),
   Description (
     "The virtual machines running on each physical node's hypervisor." )]
class KF_VMInNVE : CIM_CollectedMSEs {
    [Aggregate, Override ( "Collection" ),
     Max ( 1 ),
     Description (
```

```
      "The VNE collection that aggregates the members participating in it." )]
  KF_NVECollection REF Collection;
    [Override ( "Member" ),
     Min ( 1 ),
     Description ( "The virtual machines that run on each physical node's hypervisor." )]
  KF_VM REF Member;
};

  [Association, Aggregation, Version ( "2.6.0" ),
   Description (
     "The UNIX processes running on the virtual machines." )]
class KF_ProcessInNVE : CIM_CollectedMSEs {
    [Aggregate, Override ( "Collection" ),
     Max ( 1 ),
     Description (
       "The VNE collection that aggregates the members participating in it." )]
  KF_NVECollection REF Collection;
    [Override ( "Member" ),
     Min ( 1 ),
     Description ( "The UNIX processes running on virtual machines and are part of specific VNE applications." )]
  KF_Process REF Member;
};

  [Association, Aggregation, Version ( "2.6.0" ),
   Description (
     "The active VNE applications." )]
class KF_ApplicationInNVE : CIM_CollectedMSEs {
    [Aggregate, Override ( "Collection" ),
     Max ( 1 ),
     Description (
       "The VNE collection that aggregates the members participating in it." )]
  KF_NVECollection REF Collection;
    [Override ( "Member" ),
     Min ( 1 ),
```

```
    Description ( "The active VNE applications." )]
  KF_Application REF Member;
};

  [Association, Aggregation, Version ( "2.6.0" ),
   Description (
     "The virtual networks instantiated in the VNE." )]
class KF_NetworkInNVE : CIM_CollectedMSEs {
    [Aggregate, Override ( "Collection" ),
     Max ( 1 ),
     Description (
       "The VNE collection that aggregates the members participating in it." )]
  KF_NVECollection REF Collection;
    [Override ( "Member" ),
     Min ( 1 ),
     Description ( "The virtual networks instantiated in the VNE." )]
  KF_Network REF Member;
};
```

5. KF_settings.mof

```
// KFsettings.mof, v1.1
// CIM version 2.29.0
//
// Copyright (C) 2016 Dimitris Kontoudis, Panayiotis Fouliras.
//
// ===========================================================
//  KF VNE Model Global Settings
// ===========================================================
//
// KF model settings class

[Version ( "1.0" ),
    Description (
      "The KF_Settings class extends SettingData to provide a central point for all VNE settings" )]
class KF_Settings : CIM_SettingData {
    [Description (
        "A unique name identifying this particular Virtual Network Environment (VNE)" )]
  string NVEName;
    [Description (
        "A unique sting containing a detailed description about this VNE" )]
  string NVEDescription;
    [Description (
        "A unique number identifying this VNE" )]
  uint16 NVEid;
    [Description (
        "The number of physical nodes on which this VNE runs" )]
  uint32 MemberNodes;
    [Description (
        "The number of applications running on this VNE" )]
  uint32 MemberNodes;
};
```

6. KFVM.mof

```
// KFVM.mof, v1.1
// CIM version 2.29.0
//
// Copyright (C) 2016 Dimitris Kontoudis, Panayiotis Fouliras.
//
// ===========================================================
//  KF_VM Model
// ===========================================================
//

[Version ( "1.0" ),
    Description (
      "The KF_VM class extends OperatingSystem to abstract a particular virtual"
      "machine (i.e. operating system) instance " )]
class KF_VM : CIM_OperatingSystem {
    [Key, Description ( "A unique number identifying this particular virtual machine" )],
  uint16 NodeId;
    [Description ( "A unique number identifying the Network Virtualization Environment serviced by this virtual machine" )]
  uint16 NVEid;
    [Description ( "The hypervisor hosting the virtual machine" )]
  string RunningHypervisor;
    [Description ( "The number of applications (i.e. software code) active on this virtual machine" )]
  uint16 NumberofServicedApplications;
    [Description (
        "The number of processor cores (virtual, since the environment is run off a hypervisor)"
        "to this virtual machine" )]
  uint16 AllocatedVirtualProcessors;
    [Description (
        "The aggregated Ethernet bandwidth, in Gbps, of the virtual ethernet adapater allocated to"
        "this virtual machine" )]
  uint64 AllocatedEthernetBandwidth;
```

```
    [Description (
       "The virtual ethernet adapter type. (0) for hypervisor virtual switch type, (1) for"
                   "hardware network cards allocated to the physical node and shared to the virtual machines",
            ValueMap { "0", "1" },
            Values { "HYP-based", "HW_Nic-based" } )]
  uint8 VirtualEthernetAdapterType;
    [Description (
       "This method set the Upper and Lower Control Limits, as per the Statistical Process Control"
       "context, in order to establish the accepted boundaries of proper VM running performance. The"
                   "boundaries refer to CPU utilization." )]
  uint32 KF_SPC_SetControlLimits(
                   [IN, Description ( "The desired CPU utilization Upper Control Limit" )]
    uint32 KF_SPC_UCL,
      [IN, Description ( "The desired CPU utilization Lower Control Limit" )]
    uint32 KF_SPC_LCL
            );
    [Description (
       "This method checks whether the VM operates within the specified CPU performance parameters,"
        "i.e. UCL and LCL. Two boolean variables report respective true/false status." )]
  uint32 KF_SPC_CheckStatus(
      [IN (false), OUT, Description ( "Boolean indicating if VM is running In Control, i.e. within control limits." )]
    boolean KF_SPC_InControl,
      [IN (false), OUT, Description ( "Boolean indicating if VM is running Out of Control, i.e. outside control limits." )]
    boolean KF_SPC_OutofControl
            );
    [Description (
       "This method modifies the VMs CPU running capacity (i.e. active cores) so as to keep the performance"
       "withing acceptable control limits (i.e. in control)." )]
  uint32 KF_ModCPU(
                   [IN, Description ( "The desired active CPU cores" )]
    uint16 KF_SPC_Cores
            );
};
```

Appendix D – KF controller code

1. Disclaimer notice

2. KF controller

```
#!/bin/ksh
# NB: establish SSH key exchange between the system running the controller
#     and the HMC managing the target VM/LPAR prior to controller execution

# Trap HUP, INT, QUIT and TERM signals and exit the program
trap "abend \" Abnormally exiting at user request. Received HUP, INT, QUIT or TERM signal\" 3" 1 2 3 15

# Setup operating parameters
function setparms {
        NOW=`date +%C%y_%m_%d_%H:%M:%S`       # Obtain general execution timestamp
        FRAME_PRD=Server2-SN06C0111           # Physical frame hosting managed system
        NODE_TO_ADD=vm1                       # Managed system name
        HMC_NAME=hmckm1                       # HMC name
        HMC_IP=10.32.106.96                   # HMC IP address
        PROC_TO_ADD=1                         # Managed VM CPU expansion block
        CPU_CNT=5                             # Managed VM CPU initial capacity
        VPROC_TO_ADD=16                       # Minimum processing units required for each virtual processor : 0.10
        PERIOD=3705                           # Management time period (in seconds)
        PSEXC=$PERIOD                         # Management time period (used for final statistical report)
        UCL=71,09                             # SPC Parameters - Upper Control Limit
        LCL=57,95                             # SPC Parameters - Lower Control Limit
        OCC_HIGH=0                            # SPC Parameters - Counter for perf. point exceeding UCL in loop
        OCC_HIGH_TOTAL=0                      # SPC Parameters - Total no. of points exceeded UCL
        OCC_LOW=0                             # SPC Parameters - Counter for perf. point below LCL in loop
        OCC_LOW_TOTAL=0                       # SPC Parameters - Total no. of points below LCL
        CPU_MGMT=0                            # Total no. of CPU reconfigurations performed on the managed system
        OCC_HIGH_TRG=0                        # Trigger of initiating action of perf. readings exceeding UCL
        OCC_LOW_TRG=0                         # Trigger of initiating action of perf. readings below LCL
        HIGH_TRG=0                            # Total no. of triggers for perf. readings exceeding UCL
        LOW_TRG=0                             # Total no. of triggers for perf. readings below LCL
        TRG=3                                 # No. of consecutive occurences before any management action is taken
```

```
        SAMPLING_STEP=15                # Performance reading time step (in seconds)
        RUN=1                           # General counter, measuring no. of iterations
        BASE=/usr/local/dev             # KF Controller storage base directory
        LOGDIR=$BASE/log                # Log directory
        LOG=$LOGDIR/$NOW/vspc.log.txt   # Logfile

        if [ ! -d $LOGDIR/$NOW ]; then
                mkdir -p $LOGDIR/$NOW
                mkdir $LOGDIR/$NOW/perf
                touch $LOG
        fi
}

# Display the disclaimer text
function disp_disclaimer {
cat << EOF | tee -a $LOG
            ***  INSERT HERE COMPLETE TEXT APPEARING IN APPENDIX D.1 (Disclaimer text) ***


EOF
}

# Confirmation function
function confirm {
echo
  printf "$1 `tput bold`[yes/no]`tput rmso`? " | tee -a $LOG
  read resp
  echo
  if [ "$resp" = "yes" -o "$resp" = "y" ]; then
    return 0
  else
    return 1
  fi
}
```

```
# Display a message to user console and log the message to the active log
function log {
      echo $1 | tee -a $LOG
}

# Display an elevated priority message to user console and log the message to the active log
function report {
      echo "*** `tput smso`$1`tput rmso`" | tee -a $LOG
}

# Check to see if the HMC management system is alive and accessible
function check_hmc {
ping -c 2 $HMC_NAME > /dev/null 2>&1
if [ $? -eq 0 ]
then
      log "control system $HMC_NAME is alive - continuing KF operation"
else
      log "control system $HMC_NAME is not alive, KF operation can't be performed - controller exiting"
      abend " STOPPING KF OPERATION - Abnormal exit" 2
fi
}

# Get system's overall CPU utilization
function get_util {
            CPU_UTIL=`sar 1|tail -1|awk '{print 100-$5}'`
            return $CPU_UTIL
}

# Perform the SPC related operation
function vspc_mgmt {
log "entering management loop for a $PERIOD seconds time period"
VSPC_ST=`date +%H:%M:%S`
while [[ $PERIOD > 0 ]]
```

```
do
while read CPU_UTIL
do
                        echo
                        log "> Run: $RUN"
                        # get_util
                        log "> VM CPU capacity at `lparstat 1 1 | tail -1| awk '{print  $5}'` virtual processor cores, with utilization: $CPU_UTIL %"
# lsdev -Ccprocessor|grep Available|wc -l
        if [[ $CPU_UTIL > $UCL ]]; then
            ((OCC_HIGH_TOTAL = OCC_HIGH_TOTAL + 1))
            ((HIGH_TRG = HIGH_TRG + 1))
            log "+ $CPU_UTIL > $UCL detected: total high points $OCC_HIGH_TOTAL"
        else
            HIGH_TRG=0
        fi
        if [[ $CPU_UTIL < $LCL ]]; then
            ((OCC_LOW_TOTAL = OCC_LOW_TOTAL + 1))
            ((LOW_TRG = LOW_TRG + 1))
            log "- $CPU_UTIL < $LCL detected: total low points $OCC_LOW_TOTAL"
        else
            LOW_TRG=0
        fi
        if [[ $HIGH_TRG = $TRG || $LOW_TRG = $TRG ]]; then
            if [[ $HIGH_TRG = $TRG ]]; then
                report "== ACTION TRIGGERED - ADDING 1 CPU CORE"
                ((OCC_HIGH_TRG = OCC_HIGH_TRG + 1))
                ((CPU_MGMT = CPU_MGMT + 1))
                ((CPU_CNT = CPU_CNT + 1))
                HIGH_TRG=0
                LOW_TRG=0
            else
                report "== ACTION TRIGGERED - REMOVING 1 CPU CORE"
                ((OCC_LOW_TRG = OCC_LOW_TRG + 1))
                ((CPU_MGMT = CPU_MGMT + 1))
```

```
                        ((CPU_CNT = CPU_CNT - 1))
                        HIGH_TRG=0
                        LOW_TRG=0
                fi
        fi
        ((PERIOD = PERIOD - $SAMPLING_STEP))
        ((RUN = RUN + 1))
        log "> seconds left: $PERIOD"
#        sleep $SAMPLING_STEP
done < $BASE/simulation/run.data
done
VSPC_FN=`date +%H:%M:%S`
}

# Calculate time difference between two operation intervals
function vspc_ctrl_time {
        stime=$1
        etime=$2
        echo $stime|tr ":" " "|read shh smm sss
        echo $etime|tr ":" " "|read ehh emm ess
        ((shhs=$shh*3600))
        ((ehhs=$ehh*3600))
        ((smms=$smm*60))
        ((emms=$emm*60))
        ((ssecs=$shhs+$smms+$sss))
        ((esecs=$ehhs+$emms+$ess))
        ((delta=$esecs-$ssecs))
        if [ $delta -lt 0 ]
        then
                ((delta=$delta+86400))
        fi
        if [ "$3" = "hh:mm:ss" ]
        then
                typeset -Z2 dhh dmm dss
```

```
                ((dhh=$delta/3600))
                ((delta=$delta-(3600*$dhh)))
                ((dmm=$delta/60))
                ((dss=$delta-(60*$dmm)))
                echo "$dhh:$dmm:$dss"
        else
                echo $delta
        fi
}

# Convert seconds to hhmmss
function conv_s_to_hr {
        hour=$(($1/3600))
        minute=$(($1/60-60*$hour))
        sec=$(($1-3600*$hour-60*$minute))
        echo $hour'h '$minute'm '$sec's'
}

# Display statistics
function display_stats {
        report " End of Management Period"
        log "--     Statistics:"
        log "--     Total no. of perf. readings acquired from $NODE_TO_ADD managed system: `expr $PSEXC / $SAMPLING_STEP`"
        log "--         no. of readings exceeding UCL: $OCC_HIGH_TOTAL"
        log "--         no. of readings below LCL: $OCC_LOW_TOTAL"
        log "--         no. of CPU reconfigurations: $CPU_MGMT"
        log "--         no. of CPU additions: $OCC_HIGH_TRG"
        log "--         no. of CPU reductions: $OCC_LOW_TRG"
        log "--     Resulting VM CPU capacity: $CPU_CNT cores"
        log "--     Requested management duration: `conv_s_to_hr $PSEXC`"
        TTM=`vspc_ctrl_time $VSPC_ST $VSPC_FN`; log "--      KF controller executed for: `conv_s_to_hr $TTM`"
        report " STOPPING KF OPERATION"
}
```

```
# Gracefull finish function
function do_finish {
      exit 0
}
# Abnormal finish function
function abend {
            echo
      echo "*** `tput smso`$1`tput rmso`" | tee -a $LOG
            echo "*** Exit code $2" | tee -a $LOG
      exit $2
}
####### Main program body
# Initialize operating parameters
setparms

# Display disclaimer and request user acknowledgement before proceeding with controller operation
disp_disclaimer
confirm "If you agree with the above COPYRIGHT NOTICE AND DISCLAIMER OF WARRANTY, type 'yes' or 'y' to proceed."
[[ $? = 1 ]] && abend " COPYRIGHT NOTICE AND DISCLAIMER OF WARRANTY not accepted - STOPPING KF OPERATION" 1

# Initialize logfile and console output
log ""
report " $NOW - STARTING KF OPERATION"

# Check that HMC is alive and responding to requests
check_hmc
# Initiate primary control loop
vspc_mgmt
# Display statistical details of controller operation
display_stats
# Graceful exit
do_finish
```

3. KF controller performance data collection and preprocessing

```
#!/bin/ksh
#
# KF performance data collection and preprocessing script
#
#

# Trap HUP, INT, QUIT and TERM signals and exit the program
trap "abend \" Abnormally exiting at user request. Received HUP, INT, QUIT or TERM signal\" 3" 1 2 3 15

# Setup operating parameters
function setparms {
        NOW=`date +%C%y_%m_%d_%H:%M:%S`       # Obtain general execution timestamp
        FRAME_PRD=Server2-SN06C0111           # Physical frame hosting managed system
        NODE_TO_ADD=vm1                       # Managed system name
        HMC_NAME=hmckm1                       # HMC name
        HMC_IP=10.32.106.96                   # HMC IP address
        SAMPLING_STEP=1                       # Performance reading time step (in seconds)
        PERIOD=600                            # Performance period duration (in seconds)
        RUN=5                                 # General counter, measuring no. of iterations
        BASE=/usr/local/dev                   # KF Controller storage base directory
        PERFDIR=$BASE/perfdata                # Performance collection directory
        LOGDIR=$PERFDIR/$NOW                  # Log directory
        LOG=$LOGDIR/vspc_data_col.log.txt     # Logfile

    if [ ! -d $PERFDIR ]; then
        mkdir -p $PERFDIR
    fi

        if [ ! -d $LOGDIR ]; then
                mkdir -p $LOGDIR
                touch $LOG
```

```
        fi
}

# Display a message to user console and log the message to the active log
function log {
    echo $1 | tee -a $LOG
}

# Display an elevated priority message to user console and log the message to the active log
function report {
    echo "*** `tput smso`$1`tput rmso`" | tee -a $LOG
}

# Data collection function
function data_col {
    log "> starting data collection"
        COUNT=0
        while [[ $COUNT < $RUN ]]
        do
        log "> run: $COUNT"
        lparstat  $SAMPLING_STEP $PERIOD > $LOGDIR/lparstat.$COUNT.txt
                # lsdev -Ccprocessor|grep Available|wc -l
                log "> `date +%C%y_%m_%d-%H:%M:%S`"
                log "> completed"
        ((COUNT = COUNT + 1))
        done
}

# Calculate total CPU utilization for each dataset
# as {100-%idle} or {100-(%user+%sys+%wait)}
function calc_CPU_util {
    log "> calculating total CPU utilization"
        for datafile in `ls $LOGDIR | grep -v log`; do
                tail +6 $LOGDIR/$datafile > $LOGDIR/$datafile.CPU.noheader.txt
```

```
                while read line; do
                        echo $line | awk '{print 100-$4}' >> $LOGDIR/$datafile.CPU.total.txt
                done < $LOGDIR/$datafile.CPU.noheader.txt
        done
        /bin/rm $LOGDIR/*noheader*
        /usr/bin/paste $LOGDIR/*total* > $LOGDIR/aggregated_data.txt
    log "> cummulative data file created"
    log "> temporary files deleted"
        /bin/rm $LOGDIR/*total*
}

function calc_avg {
    log "> calculating averages"
        while read line; do
                echo $line | awk '{ for(i=1; i<=NF;i++) j+=$i; print j/(i-1); j=0 }' >> $LOGDIR/averages.txt
        done < $LOGDIR/aggregated_data.txt
    log "> averages data file created"
}

# Gracefull finish function
function do_finish {
        report "*** KF preprocessing completed successfully"
    exit 0
}

# Abnormal finish function
function abend {
        echo
    echo "*** `tput smso`$1`tput rmso`" | tee -a $LOG
        echo "*** Exit code $2" | tee -a $LOG
    exit $2
}

####### Main program body
```

```
# Initialize operating parameters
setparms

# Initialize log and screen output
report "KF preprocessing: executing $RUN times, for $PERIOD seconds each run, with sample step $SAMPLING_STEP second(s)"
report "Working directory: $LOGDIR"
log "Starting on $NOW"

# Collect performance data
data_col

## Preprocessing

# Calculate total utilization for each dataset
calc_CPU_util

# Calculate Averages
calc_avg

# Graceful exit
do_finish
```

Appendix E – Statistical Data

Table 8. First test cycle (core banking software) statistical data

TIME	RUN 1	RUN 2	RUN 3	RUN 4	RUN 5	Average		TIME	RUN 1	RUN 2	RUN 3	RUN 4	RUN 5	Average
10:00:00	0.357	0.763	0.353	0.453	0.453	0.56		10:30:42	7.451	4.997	6.211	6.54	6.196	6.37
10:00:05	0.634	0.453	0.531	0.342	0.742	0.62		10:30:57	5.336	5.787	5.476	6.442	8.197	6.34
10:00:10	0.765	0.478	0.356	0.736	0.776	0.71		10:31:12	5.36	6.465	7.048	6.245	7.92	6.70
10:00:13	4.568	5.6	5.129	7.084	5.887	5.74		10:31:27	7.646	6.508	7.927	6.369	5.679	6.91
10:00:28	6.052	6.22	5.508	7.714	4.547	6.09		10:31:42	6.911	6.447	8.753	5.853	5.847	6.85
10:00:43	4.73	6.837	5.49	8.341	4.95	6.15		10:31:57	5.957	6.818	7.798	5.751	5.062	6.36
10:00:58	5.82	6.263	4.301	8.314	6.911	6.41		10:32:12	7.683	5.353	7.923	8.696	6.099	7.24
10:01:13	4.719	7.251	4.799	7.405	6.836	6.29		10:32:27	8.235	6.664	5.618	8.086	7.51	7.31
10:01:28	4.758	6.911	5.154	7.587	5.175	6.00		10:32:42	6.488	5.254	3.306	8.159	6.777	6.08
10:01:43	5.817	5.302	4.775	7.55	4.528	5.68		10:32:57	7.856	5.294	3.629	8.367	6.442	6.41
10:01:58	6.816	4.756	4.407	6.896	4.905	5.64		10:33:12	9.074	5.671	5.733	7.317	7.318	7.11
10:02:13	6.47	7.039	7.499	8.027	5.517	6.99		10:33:27	8.482	7.804	6.131	6.47	4.391	6.74
10:02:28	7.074	7.337	8.268	8.382	6.276	7.55		10:33:42	7.859	7.508	7.051	6.625	5.186	6.93
10:02:43	7.07	7.198	6.96	6.005	7.391	7.01		10:33:57	6.731	5.786	7.989	6.288	6.106	6.67
10:02:58	5.176	5.217	4.88	6.196	6.899	5.76		10:34:12	7.63	3.671	6.49	6.481	7.771	6.50
10:03:13	7.503	4.967	4.491	7.32	7.265	6.39		10:34:27	6.742	3.685	5.822	6.631	8.794	6.42
10:03:28	6.934	5.093	3.771	5.703	8.382	6.06		10:34:42	6.03	6.709	5.919	5.95	6.391	6.29
10:03:43	8.348	3.985	4.444	5.598	5.909	5.74		10:34:57	6.356	7.973	5.942	6.111	5.063	6.38
10:03:58	5.112	4.707	4.524	7.523	7.076	5.87		10:35:12	5.128	7.948	7.433	7.58	6.393	6.98
10:04:13	6.377	5.649	4.447	7.565	6.74	6.24		10:35:27	4.885	7.222	7.519	7.087	7.154	6.86
10:04:28	5.142	5.988	5.556	6.073	6.722	5.98		10:35:42	6.41	6.867	7.467	7.366	6.09	6.93
10:04:43	3.764	6.188	4.397	4.941	8.284	5.60		10:35:57	6.805	8.116	7.02	7.362	4.611	6.87

10:04:58	3.855	6.111	5.178	6.601	8.409	6.11		10:36:12	5.386	7.484	5.506	5.848	5.927	6.12
10:05:13	4.383	5.375	6.116	6.245	5.246	5.56		10:36:27	6.901	8.16	5.045	6.335	5.615	6.50
10:05:28	4.727	5.327	6.153	5.386	5.003	5.40		10:36:42	6.164	6.44	4.353	5.066	4.707	5.43
10:05:43	6.951	6.841	8.342	6.191	4.135	6.58		10:36:57	8.522	8.051	5.275	5.602	5.906	6.76
10:05:58	5.846	6.458	6.776	5.524	4.932	5.99		10:37:12	7.936	8.275	5.679	6.054	4.859	6.65
10:06:13	5.316	5.352	6.858	4.739	5.771	5.69		10:37:27	8.284	7.088	5.724	6.299	5.592	6.69
10:06:28	5.67	7	7.817	5.482	4.263	6.13		10:37:42	8.027	7.542	6.647	6.973	8.438	7.61
10:06:43	8.212	7.544	6.024	5.789	4.439	6.49		10:37:57	7.581	6.52	6.93	6.974	7.909	7.27
10:06:58	9.279	8.282	5.33	8.018	4.756	7.22		10:38:12	7.836	7.049	7.758	6.574	6.854	7.30
10:07:13	8.963	4.87	4.177	7.4	4.046	5.98		10:38:27	4.992	8.581	7.362	6.794	5.892	6.81
10:07:28	7.459	6.321	4.135	8.176	4.396	6.18		10:38:42	5.884	8.125	4.727	6.648	5.422	6.25
10:07:43	5.68	7.469	3.077	7.847	4.424	5.78		10:38:57	5.035	7.601	4.487	5.788	6.072	5.89
10:07:58	7.025	7.723	3.606	7.1	6.351	6.45		10:39:12	4.675	7.517	6.142	7.08	6.858	6.54
10:08:13	7.349	7.329	3.882	4.281	7.336	6.12		10:39:27	6.165	7.098	7.785	7.364	5.674	6.91
10:08:28	7.556	7.071	7.857	2.879	6.386	6.43		10:39:42	4.637	6.427	7.424	6.396	5.815	6.23
10:08:43	6.607	6.556	7.229	2.776	6.566	6.03		10:39:57	6.561	5.999	5.694	8.233	5.681	6.52
10:08:58	3.897	7.655	5.284	5.123	5.611	5.60		10:40:12	6.178	7.395	5.501	7.185	4.841	6.31
10:09:13	5.794	6.868	5.122	4.315	5.851	5.67		10:40:27	5.566	6.212	7.879	6.312	5.113	6.31
10:09:28	5.141	7.558	4.4	4.745	4.163	5.29		10:40:42	6.193	4.529	8.18	5.799	5.458	6.12
10:09:43	5.762	8.208	5.598	5.517	6.109	6.32		10:40:57	5.012	5.651	7.766	4.66	3.098	5.33
10:09:58	4.811	8.669	6.193	7.008	3.931	6.21		10:41:12	6.176	6.581	8.641	5.512	5.731	6.62
10:10:13	4.78	7.693	7.403	7.217	5.582	6.62		10:41:27	6.445	7.228	8.088	6.136	7.254	7.12
10:10:28	5.881	6.239	6.715	8.214	4.931	6.48		10:41:42	6.822	8.157	7.904	7.172	5.98	7.30
10:10:43	7.378	4.22	7.325	8.182	4.423	6.39		10:41:57	7.874	8.447	5.972	7.096	6.185	7.20
10:10:58	7.838	5.646	5.712	8.827	6.379	6.97		10:42:12	6.311	8.295	6.111	7.72	5.066	6.79
10:11:13	7.614	5.917	6.151	8.83	3.961	6.58		10:42:27	5.917	8.433	6.78	6.673	5.758	6.80

10:11:28	6.783	7.049	7.592	7.976	4.649	6.89		10:42:42	4.944	5.878	5.669	6.868	6.983	6.16
10:11:43	5.85	7.147	7.787	7.693	6.125	7.01		10:42:57	7.005	5.196	5.698	6.563	6.002	6.18
10:11:58	5.97	8.015	7.414	7.921	7.231	7.40		10:43:12	6.994	6.065	5.84	6.872	6.116	6.47
10:12:13	7.436	6.673	7.07	6.691	7.891	7.24		10:43:28	6.935	7.162	5.232	7.679	4.407	6.37
10:12:28	7.222	6.99	7.272	7.106	7.292	7.26		10:43:43	6.044	7.619	6.222	7.913	5.787	6.81
10:12:43	7.784	5.573	7.128	7.398	6.155	6.89		10:43:58	7.228	9.322	6.791	7.94	6.127	7.57
10:12:58	7.093	5.482	4.737	6.473	6.521	6.15		10:44:13	8.003	8.647	7.218	6.196	5.743	7.25
10:13:13	7.493	5.783	4.872	7.471	7.912	6.79		10:44:28	8.17	6.603	7.203	5.802	5.713	6.79
10:13:28	6.722	7.254	5.973	7.113	7.21	6.94		10:44:43	6.644	7.596	6.926	4.148	5.152	6.18
10:13:43	7.191	8.099	5.522	7.045	8.74	7.40		10:44:58	6.026	7.372	5.165	6.231	5.083	6.06
10:13:58	7.311	7.852	6.522	5.91	8.781	7.36		10:45:13	7.177	7.071	6.786	5.74	5.794	6.60
10:14:13	7.557	7.295	5.28	4.694	7.302	6.51		10:45:28	6.913	6.149	7.229	5.603	6.733	6.62
10:14:28	6.381	6.15	6.726	4.976	7.319	6.40		10:45:43	7.198	6.724	7.491	5.27	6.259	6.68
10:14:43	5.48	4.59	5.997	5.545	5.955	5.60		10:45:58	6.931	7.584	7.811	4.952	7.324	7.01
10:14:58	5.002	6.323	6.059	7.428	6.473	6.34		10:46:13	7.702	6.773	6.507	7.468	8.573	7.49
10:15:13	6.214	6.045	7.212	8.301	6.889	7.02		10:46:28	9.194	7.852	6.202	8.482	8.721	8.18
10:15:28	7.592	5.874	6.64	8.507	7.137	7.24		10:46:43	7.099	8.161	5.305	7.873	6.774	7.13
10:15:43	6.028	5.749	6.499	6.343	5.718	6.15		10:46:58	5.819	7.33	5.614	7.856	5.883	6.59
10:15:58	5.842	6.937	7.998	6.898	6.939	7.01		10:47:13	6.433	7.166	5.946	5.728	6.078	6.36
10:16:13	6.621	7.278	7.408	6.599	7.359	7.14		10:47:28	6.976	6.406	4.549	7.158	6.053	6.32
10:16:28	6.947	6.991	7.599	6.901	5.331	6.84		10:47:43	6.919	7.712	5.372	6.976	5.821	6.65
10:16:43	6.843	8.108	7.324	7.733	5.288	7.14		10:47:58	6.581	6.774	7.398	6.726	8.383	7.26
10:16:58	6.39	7.294	6.557	7.097	7.726	7.10		10:48:13	5.705	5.201	8.323	6.632	7.433	6.75
10:17:12	5.562	8.215	5.392	6.554	8.613	6.95		10:48:28	6.034	5.424	8.901	6.833	7.164	6.96
10:17:27	7.457	7.897	6.695	7.847	8.682	7.80		10:48:43	4.861	6.469	7.873	6.33	6.439	6.48
10:17:42	7.973	7.112	5.295	7.5	6.583	6.98		10:48:58	7.964	5.022	7.819	6.381	5.143	6.56

10:17:57	8.115	5.818	5.476	8.899	7.468	7.24		10:49:13	4.56	5.286	6.388	6.596	7.075	6.07
10:18:12	7.18	5.076	6.511	8.397	6.103	6.74		10:49:28	4.881	8.554	7.181	7.176	6.559	6.96
10:18:27	7.002	5.671	7.398	8.743	6.98	7.24		10:49:43	6.269	8.57	7.981	6.909	5.814	7.20
10:18:42	5.706	7.036	6.223	7.317	6.202	6.58		10:49:58	5.034	8.323	7.329	7.202	5.914	6.85
10:18:57	5.968	5.43	6.815	6.388	4.537	5.91		10:50:13	4.615	7.505	7.632	8.407	7.731	7.27
10:19:12	5.739	6.698	6.909	7.002	5.643	6.48		10:50:28	7.709	7.556	7.973	6.952	6.204	7.37
10:19:27	6.944	6.426	6.896	6.146	6.235	6.62		10:50:43	7.689	6.808	7.136	6.853	4.287	6.64
10:19:42	6.656	5.15	6.958	6.711	6.645	6.51		10:50:58	9.125	6.598	8.301	6.35	4.498	7.06
10:19:57	6.855	5.827	7.842	6.275	5.71	6.59		10:51:13	9.383	8.156	8.201	6.896	3.867	7.39
10:20:12	6.891	6.07	8.843	6.904	7.14	7.26		10:51:28	7.051	6.75	8.476	7.974	5.667	7.27
10:20:27	6.691	8.141	7.869	5.631	5.418	6.84		10:51:43	6.81	6.031	7.33	7.25	8.194	7.21
10:20:42	5.164	8.154	6.678	6.743	5.567	6.55		10:51:58	6.417	5.469	7.777	7.206	7.56	6.98
10:20:57	4.815	7.724	5.786	5.645	6.391	6.16		10:52:13	5.125	6.204	6.253	7.683	7.673	6.68
10:21:12	6.27	7.03	7.194	4.718	6.689	6.47		10:52:28	6.252	4.915	7.042	6.426	5.903	6.20
10:21:27	7.219	6.283	6.507	4.402	6.61	6.29		10:52:43	6.258	3.767	6.284	6.789	6.42	5.99
10:21:42	5.449	7.411	6.616	7.547	6.973	6.89		10:52:58	6.976	4.603	7.179	7.405	6.754	6.67
10:21:57	6.31	8.246	6.246	6.819	6.283	6.87		10:53:13	4.794	7.41	7.674	7.349	6.306	6.80
10:22:12	6.81	7.759	5.49	6.378	5.952	6.56		10:53:28	7.062	6.987	8.143	6.788	7.184	7.32
10:22:27	5.024	8.043	5.107	5.29	7.937	6.37		10:53:43	4.088	7.172	6.819	6.895	6.655	6.42
10:22:42	4.492	7.097	4.012	6.467	7.97	6.09		10:53:58	4.425	6.434	8.216	7.502	4.147	6.24
10:22:57	5.019	7.169	3.93	6.357	7.145	6.01		10:54:13	6.927	5.849	6.369	6.735	3.672	6.00
10:23:12	5.765	5.869	5.508	7.711	7.748	6.61		10:54:28	6.167	5.944	6.596	6.443	3.429	5.81
10:23:27	4.412	5.128	7.449	7.911	6.219	6.31		10:54:43	6.483	6.366	7.649	7.513	3.15	6.32
10:23:42	5.306	4.331	8.234	7.73	6.413	6.49		10:54:58	5.616	6.262	7.748	7.078	3.756	6.18
10:23:57	4.534	4.85	6.324	6.483	5.623	5.65		10:55:13	5.833	7.623	7.411	7.385	6.242	6.99
10:24:12	6.586	6.739	5.35	8.13	5.681	6.58		10:55:28	5.271	7.87	7.108	6.599	6.248	6.71

10:24:27	6.287	6.847	4.684	5.169	6.245	5.93		10:55:43	6.383	7.489	7.26	6.547	7.493	7.13
10:24:42	4.504	8.581	4.513	4.869	5.927	5.77		10:55:58	5.873	6.194	6.524	6.207	5.49	6.15
10:24:57	6.559	7.459	4.976	6.73	4.579	6.15		10:56:13	6.43	5.327	5.139	5.624	6.208	5.84
10:25:12	7.085	8.054	4.772	7.201	5.891	6.69		10:56:28	7.198	4.218	5.862	7.432	5.399	6.11
10:25:27	7.213	7.532	6.347	7.022	5.204	6.75		10:56:43	6.158	5.32	4.653	7.925	4.315	5.77
10:25:42	7.149	5.843	5.855	6.154	3.637	5.81		10:56:58	5.414	5.066	6.139	8.397	3.757	5.85
10:25:57	6.035	6.699	4.499	5.543	4.085	5.46		10:57:13	5.918	7.372	5.843	8.588	5.462	6.73
10:26:12	5.787	8.129	4.525	7.018	2.855	5.75		10:57:28	7.379	6.689	6.639	6.939	5.757	6.77
10:26:27	4.97	6.972	2.453	6.194	5.857	5.38		10:57:43	7.133	7.411	7.253	7.417	5.999	7.13
10:26:42	6.731	7.616	4.175	6.458	8.024	6.69		10:57:58	5.806	7.821	8.625	6.544	7.463	7.34
10:26:57	5.853	7.609	5.704	6.525	5.274	6.28		10:58:13	5.294	7.329	7.732	6.954	7.417	7.04
10:27:12	6.607	7.364	7.826	5.948	5.069	6.65		10:58:28	4.317	5.731	8.521	7.191	7.758	6.80
10:27:27	6.129	6.316	6.224	6.978	4.916	6.20		10:58:43	5.923	5.513	7.93	5.667	4.733	6.04
10:27:42	6.148	7.07	4.713	5.952	5.245	5.91		10:58:58	3.765	6.103	8.009	6.202	3.415	5.59
10:27:57	5.189	5.056	4.548	7.345	5.773	5.67		10:59:13	4.529	5.555	6.832	7.319	2.818	5.50
10:28:12	6.091	7.356	4.239	7.678	4.564	6.07		10:59:28	5.603	7.646	7.69	8.642	2.527	6.51
10:28:27	6.883	8.286	3.573	7.656	4.579	6.28		10:59:43	6.931	7.676	7.16	7.727	4.066	6.80
10:28:42	5.663	8.16	5.387	8.012	4.974	6.53		10:59:58	4.182	8.329	7.029	7.962	6.323	6.86
10:28:57	5.228	7.13	6.401	8.658	5.801	6.73		11:00:13	5.577	4.299	5.728	7.327	4.596	5.60
10:29:12	5.603	7.281	6.233	8.109	4.551	6.44		11:00:28	5.336	5.105	5.956	8.169	3.42	5.69
10:29:27	5.659	5.86	8.31	7.63	5.485	6.68		11:00:35	0.671	0.828	0.529	0.151	0.243	0.58
10:29:42	6.217	4.9	6.446	6.767	3.975	5.75		11:00:40	0.48	0.251	0.317	0.269	0.307	0.42
10:29:57	8.044	8.02	5.92	7.653	6.369	7.29		11:00:45	0.194	0.253	0.309	0.191	0.179	0.32
10:30:12	6.683	7.445	6.076	8.251	4.943	6.77		11:00:50	0.051	0.458	0.605	0.188	0.365	0.43
10:30:27	8.129	4.939	7.15	6.822	5.774	6.65		11:00:55	0.056	0.541	0.429	0.238	0.147	0.37
								11:01:00	0.141	0.597	0.229	0.228	0.134	0.36

Table 9. Second test cycle (virtual router) statistical data

TIME	RUN 1	RUN 2	RUN 3	RUN 4	RUN 5	Average		TIME	RUN 1	RUN 2	RUN 3	RUN 4	RUN 5	Average
10:00:00	0.357	0.763	0.353	0.453	0.453	0.56		10:30:42	7.451	4.997	6.211	6.54	6.196	6.37
10:00:05	0.634	0.453	0.531	0.342	0.742	0.62		10:30:57	5.336	5.787	5.476	6.442	8.197	6.34
10:00:10	0.765	0.478	0.356	0.736	0.776	0.71		10:31:12	5.36	6.465	7.048	6.245	7.92	6.70
10:00:13	4.568	5.6	5.129	7.084	5.887	5.74		10:31:27	7.646	6.508	7.927	6.369	5.679	6.91
10:00:28	6.052	6.22	5.508	7.714	4.547	6.09		10:31:42	6.911	6.447	8.753	5.853	5.847	6.85
10:00:43	4.73	6.837	5.49	8.341	4.95	6.15		10:31:57	5.957	6.818	7.798	5.751	5.062	6.36
10:00:58	5.82	6.263	4.301	8.314	6.911	6.41		10:32:12	7.683	5.353	7.923	8.696	6.099	7.24
10:01:13	4.719	7.251	4.799	7.405	6.836	6.29		10:32:27	8.235	6.664	5.618	8.086	7.51	7.31
10:01:28	4.758	6.911	5.154	7.587	5.175	6.00		10:32:42	6.488	5.254	3.306	8.159	6.777	6.08
10:01:43	5.817	5.302	4.775	7.55	4.528	5.68		10:32:57	7.856	5.294	3.629	8.367	6.442	6.41
10:01:58	6.816	4.756	4.407	6.896	4.905	5.64		10:33:12	9.074	5.671	5.733	7.317	7.318	7.11
10:02:13	6.47	7.039	7.499	8.027	5.517	6.99		10:33:27	8.482	7.804	6.131	6.47	4.391	6.74
10:02:28	7.074	7.337	8.268	8.382	6.276	7.55		10:33:42	7.859	7.508	7.051	6.625	5.186	6.93
10:02:43	7.07	7.198	6.96	6.005	7.391	7.01		10:33:57	6.731	5.786	7.989	6.288	6.106	6.67
10:02:58	5.176	5.217	4.88	6.196	6.899	5.76		10:34:12	7.63	3.671	6.49	6.481	7.771	6.50
10:03:13	7.503	4.967	4.491	7.32	7.265	6.39		10:34:27	6.742	3.685	5.822	6.631	8.794	6.42
10:03:28	6.934	5.093	3.771	5.703	8.382	6.06		10:34:42	6.03	6.709	5.919	5.95	6.391	6.29
10:03:43	8.348	3.985	4.444	5.598	5.909	5.74		10:34:57	6.356	7.973	5.942	6.111	5.063	6.38
10:03:58	5.112	4.707	4.524	7.523	7.076	5.87		10:35:12	5.128	7.948	7.433	7.58	6.393	6.98
10:04:13	6.377	5.649	4.447	7.565	6.74	6.24		10:35:27	4.885	7.222	7.519	7.087	7.154	6.86
10:04:28	5.142	5.988	5.556	6.073	6.722	5.98		10:35:42	6.41	6.867	7.467	7.366	6.09	6.93
10:04:43	3.764	6.188	4.397	4.941	8.284	5.60		10:35:57	6.805	8.116	7.02	7.362	4.611	6.87
10:04:58	3.855	6.111	5.178	6.601	8.409	6.11		10:36:12	5.386	7.484	5.506	5.848	5.927	6.12
10:05:13	4.383	5.375	6.116	6.245	5.246	5.56		10:36:27	6.901	8.16	5.045	6.335	5.615	6.50
10:05:28	4.727	5.327	6.153	5.386	5.003	5.40		10:36:42	6.164	6.44	4.353	5.066	4.707	5.43

10:05:43	6.951	6.841	8.342	6.191	4.135	6.58		10:36:57	8.522	8.051	5.275	5.602	5.906	6.76
10:05:58	5.846	6.458	6.776	5.524	4.932	5.99		10:37:12	7.936	8.275	5.679	6.054	4.859	6.65
10:06:13	5.316	5.352	6.858	4.739	5.771	5.69		10:37:27	8.284	7.088	5.724	6.299	5.592	6.69
10:06:28	5.67	7	7.817	5.482	4.263	6.13		10:37:42	8.027	7.542	6.647	6.973	8.438	7.61
10:06:43	8.212	7.544	6.024	5.789	4.439	6.49		10:37:57	7.581	6.52	6.93	6.974	7.909	7.27
10:06:58	9.279	8.282	5.33	8.018	4.756	7.22		10:38:12	7.836	7.049	7.758	6.574	6.854	7.30
10:07:13	8.963	4.87	4.177	7.4	4.046	5.98		10:38:27	4.992	8.581	7.362	6.794	5.892	6.81
10:07:28	7.459	6.321	4.135	8.176	4.396	6.18		10:38:42	5.884	8.125	4.727	6.648	5.422	6.25
10:07:43	5.68	7.469	3.077	7.847	4.424	5.78		10:38:57	5.035	7.601	4.487	5.788	6.072	5.89
10:07:58	7.025	7.723	3.606	7.1	6.351	6.45		10:39:12	4.675	7.517	6.142	7.08	6.858	6.54
10:08:13	7.349	7.329	3.882	4.281	7.336	6.12		10:39:27	6.165	7.098	7.785	7.364	5.674	6.91
10:08:28	7.556	7.071	7.857	2.879	6.386	6.43		10:39:42	4.637	6.427	7.424	6.396	5.815	6.23
10:08:43	6.607	6.556	7.229	2.776	6.566	6.03		10:39:57	6.561	5.999	5.694	8.233	5.681	6.52
10:08:58	3.897	7.655	5.284	5.123	5.611	5.60		10:40:12	6.178	7.395	5.501	7.185	4.841	6.31
10:09:13	5.794	6.868	5.122	4.315	5.851	5.67		10:40:27	5.566	6.212	7.879	6.312	5.113	6.31
10:09:28	5.141	7.558	4.4	4.745	4.163	5.29		10:40:42	6.193	4.529	8.18	5.799	5.458	6.12
10:09:43	5.762	8.208	5.598	5.517	6.109	6.32		10:40:57	5.012	5.651	7.766	4.66	3.098	5.33
10:09:58	4.811	8.669	6.193	7.008	3.931	6.21		10:41:12	6.176	6.581	8.641	5.512	5.731	6.62
10:10:13	4.78	7.693	7.403	7.217	5.582	6.62		10:41:27	6.445	7.228	8.088	6.136	7.254	7.12
10:10:28	5.881	6.239	6.715	8.214	4.931	6.48		10:41:42	6.822	8.157	7.904	7.172	5.98	7.30
10:10:43	7.378	4.22	7.325	8.182	4.423	6.39		10:41:57	7.874	8.447	5.972	7.096	6.185	7.20
10:10:58	7.838	5.646	5.712	8.827	6.379	6.97		10:42:12	6.311	8.295	6.111	7.72	5.066	6.79
10:11:13	7.614	5.917	6.151	8.83	3.961	6.58		10:42:27	5.917	8.433	6.78	6.673	5.758	6.80
10:11:28	6.783	7.049	7.592	7.976	4.649	6.89		10:42:42	4.944	5.878	5.669	6.868	6.983	6.16
10:11:43	5.85	7.147	7.787	7.693	6.125	7.01		10:42:57	7.005	5.196	5.698	6.563	6.002	6.18
10:11:58	5.97	8.015	7.414	7.921	7.231	7.40		10:43:12	6.994	6.065	5.84	6.872	6.116	6.47
10:12:13	7.436	6.673	7.07	6.691	7.891	7.24		10:43:28	6.935	7.162	5.232	7.679	4.407	6.37

10:12:28	7.222	6.99	7.272	7.106	7.292	7.26		10:43:43	6.044	7.619	6.222	7.913	5.787	6.81
10:12:43	7.784	5.573	7.128	7.398	6.155	6.89		10:43:58	7.228	9.322	6.791	7.94	6.127	7.57
10:12:58	7.093	5.482	4.737	6.473	6.521	6.15		10:44:13	8.003	8.647	7.218	6.196	5.743	7.25
10:13:13	7.493	5.783	4.872	7.471	7.912	6.79		10:44:28	8.17	6.603	7.203	5.802	5.713	6.79
10:13:28	6.722	7.254	5.973	7.113	7.21	6.94		10:44:43	6.644	7.596	6.926	4.148	5.152	6.18
10:13:43	7.191	8.099	5.522	7.045	8.74	7.40		10:44:58	6.026	7.372	5.165	6.231	5.083	6.06
10:13:58	7.311	7.852	6.522	5.91	8.781	7.36		10:45:13	7.177	7.071	6.786	5.74	5.794	6.60
10:14:13	7.557	7.295	5.28	4.694	7.302	6.51		10:45:28	6.913	6.149	7.229	5.603	6.733	6.62
10:14:28	6.381	6.15	6.726	4.976	7.319	6.40		10:45:43	7.198	6.724	7.491	5.27	6.259	6.68
10:14:43	5.48	4.59	5.997	5.545	5.955	5.60		10:45:58	6.931	7.584	7.811	4.952	7.324	7.01
10:14:58	5.002	6.323	6.059	7.428	6.473	6.34		10:46:13	7.702	6.773	6.507	7.468	8.573	7.49
10:15:13	6.214	6.045	7.212	8.301	6.889	7.02		10:46:28	9.194	7.852	6.202	8.482	8.721	8.18
10:15:28	7.592	5.874	6.64	8.507	7.137	7.24		10:46:43	7.099	8.161	5.305	7.873	6.774	7.13
10:15:43	6.028	5.749	6.499	6.343	5.718	6.15		10:46:58	5.819	7.33	5.614	7.856	5.883	6.59
10:15:58	5.842	6.937	7.998	6.898	6.939	7.01		10:47:13	6.433	7.166	5.946	5.728	6.078	6.36
10:16:13	6.621	7.278	7.408	6.599	7.359	7.14		10:47:28	6.976	6.406	4.549	7.158	6.053	6.32
10:16:28	6.947	6.991	7.599	6.901	5.331	6.84		10:47:43	6.919	7.712	5.372	6.976	5.821	6.65
10:16:43	6.843	8.108	7.324	7.733	5.288	7.14		10:47:58	6.581	6.774	7.398	6.726	8.383	7.26
10:16:58	6.39	7.294	6.557	7.097	7.726	7.10		10:48:13	5.705	5.201	8.323	6.632	7.433	6.75
10:17:12	5.562	8.215	5.392	6.554	8.613	6.95		10:48:28	6.034	5.424	8.901	6.833	7.164	6.96
10:17:27	7.457	7.897	6.695	7.847	8.682	7.80		10:48:43	4.861	6.469	7.873	6.33	6.439	6.48
10:17:42	7.973	7.112	5.295	7.5	6.583	6.98		10:48:58	7.964	5.022	7.819	6.381	5.143	6.56
10:17:57	8.115	5.818	5.476	8.899	7.468	7.24		10:49:13	4.56	5.286	6.388	6.596	7.075	6.07
10:18:12	7.18	5.076	6.511	8.397	6.103	6.74		10:49:28	4.881	8.554	7.181	7.176	6.559	6.96
10:18:27	7.002	5.671	7.398	8.743	6.98	7.24		10:49:43	6.269	8.57	7.981	6.909	5.814	7.20
10:18:42	5.706	7.036	6.223	7.317	6.202	6.58		10:49:58	5.034	8.323	7.329	7.202	5.914	6.85
10:18:57	5.968	5.43	6.815	6.388	4.537	5.91		10:50:13	4.615	7.505	7.632	8.407	7.731	7.27

10:19:12	5.739	6.698	6.909	7.002	5.643	6.48		10:50:28	7.709	7.556	7.973	6.952	6.204	7.37
10:19:27	6.944	6.426	6.896	6.146	6.235	6.62		10:50:43	7.689	6.808	7.136	6.853	4.287	6.64
10:19:42	6.656	5.15	6.958	6.711	6.645	6.51		10:50:58	9.125	6.598	8.301	6.35	4.498	7.06
10:19:57	6.855	5.827	7.842	6.275	5.71	6.59		10:51:13	9.383	8.156	8.201	6.896	3.867	7.39
10:20:12	6.891	6.07	8.843	6.904	7.14	7.26		10:51:28	7.051	6.75	8.476	7.974	5.667	7.27
10:20:27	6.691	8.141	7.869	5.631	5.418	6.84		10:51:43	6.81	6.031	7.33	7.25	8.194	7.21
10:20:42	5.164	8.154	6.678	6.743	5.567	6.55		10:51:58	6.417	5.469	7.777	7.206	7.56	6.98
10:20:57	4.815	7.724	5.786	5.645	6.391	6.16		10:52:13	5.125	6.204	6.253	7.683	7.673	6.68
10:21:12	6.27	7.03	7.194	4.718	6.689	6.47		10:52:28	6.252	4.915	7.042	6.426	5.903	6.20
10:21:27	7.219	6.283	6.507	4.402	6.61	6.29		10:52:43	6.258	3.767	6.284	6.789	6.42	5.99
10:21:42	5.449	7.411	6.616	7.547	6.973	6.89		10:52:58	6.976	4.603	7.179	7.405	6.754	6.67
10:21:57	6.31	8.246	6.246	6.819	6.283	6.87		10:53:13	4.794	7.41	7.674	7.349	6.306	6.80
10:22:12	6.81	7.759	5.49	6.378	5.952	6.56		10:53:28	7.062	6.987	8.143	6.788	7.184	7.32
10:22:27	5.024	8.043	5.107	5.29	7.937	6.37		10:53:43	4.088	7.172	6.819	6.895	6.655	6.42
10:22:42	4.492	7.097	4.012	6.467	7.97	6.09		10:53:58	4.425	6.434	8.216	7.502	4.147	6.24
10:22:57	5.019	7.169	3.93	6.357	7.145	6.01		10:54:13	6.927	5.849	6.369	6.735	3.672	6.00
10:23:12	5.765	5.869	5.508	7.711	7.748	6.61		10:54:28	6.167	5.944	6.596	6.443	3.429	5.81
10:23:27	4.412	5.128	7.449	7.911	6.219	6.31		10:54:43	6.483	6.366	7.649	7.513	3.15	6.32
10:23:42	5.306	4.331	8.234	7.73	6.413	6.49		10:54:58	5.616	6.262	7.748	7.078	3.756	6.18
10:23:57	4.534	4.85	6.324	6.483	5.623	5.65		10:55:13	5.833	7.623	7.411	7.385	6.242	6.99
10:24:12	6.586	6.739	5.35	8.13	5.681	6.58		10:55:28	5.271	7.87	7.108	6.599	6.248	6.71
10:24:27	6.287	6.847	4.684	5.169	6.245	5.93		10:55:43	6.383	7.489	7.26	6.547	7.493	7.13
10:24:42	4.504	8.581	4.513	4.869	5.927	5.77		10:55:58	5.873	6.194	6.524	6.207	5.49	6.15
10:24:57	6.559	7.459	4.976	6.73	4.579	6.15		10:56:13	6.43	5.327	5.139	5.624	6.208	5.84
10:25:12	7.085	8.054	4.772	7.201	5.891	6.69		10:56:28	7.198	4.218	5.862	7.432	5.399	6.11
10:25:27	7.213	7.532	6.347	7.022	5.204	6.75		10:56:43	6.158	5.32	4.653	7.925	4.315	5.77
10:25:42	7.149	5.843	5.855	6.154	3.637	5.81		10:56:58	5.414	5.066	6.139	8.397	3.757	5.85

10:25:57	6.035	6.699	4.499	5.543	4.085	5.46		10:57:13	5.918	7.372	5.843	8.588	5.462	6.73
10:26:12	5.787	8.129	4.525	7.018	2.855	5.75		10:57:28	7.379	6.689	6.639	6.939	5.757	6.77
10:26:27	4.97	6.972	2.453	6.194	5.857	5.38		10:57:43	7.133	7.411	7.253	7.417	5.999	7.13
10:26:42	6.731	7.616	4.175	6.458	8.024	6.69		10:57:58	5.806	7.821	8.625	6.544	7.463	7.34
10:26:57	5.853	7.609	5.704	6.525	5.274	6.28		10:58:13	5.294	7.329	7.732	6.954	7.417	7.04
10:27:12	6.607	7.364	7.826	5.948	5.069	6.65		10:58:28	4.317	5.731	8.521	7.191	7.758	6.80
10:27:27	6.129	6.316	6.224	6.978	4.916	6.20		10:58:43	5.923	5.513	7.93	5.667	4.733	6.04
10:27:42	6.148	7.07	4.713	5.952	5.245	5.91		10:58:58	3.765	6.103	8.009	6.202	3.415	5.59
10:27:57	5.189	5.056	4.548	7.345	5.773	5.67		10:59:13	4.529	5.555	6.832	7.319	2.818	5.50
10:28:12	6.091	7.356	4.239	7.678	4.564	6.07		10:59:28	5.603	7.646	7.69	8.642	2.527	6.51
10:28:27	6.883	8.286	3.573	7.656	4.579	6.28		10:59:43	6.931	7.676	7.16	7.727	4.066	6.80
10:28:42	5.663	8.16	5.387	8.012	4.974	6.53		10:59:58	4.182	8.329	7.029	7.962	6.323	6.86
10:28:57	5.228	7.13	6.401	8.658	5.801	6.73		11:00:13	5.577	4.299	5.728	7.327	4.596	5.60
10:29:12	5.603	7.281	6.233	8.109	4.551	6.44		11:00:28	5.336	5.105	5.956	8.169	3.42	5.69
10:29:27	5.659	5.86	8.31	7.63	5.485	6.68		11:00:35	0.671	0.828	0.529	0.151	0.243	0.58
10:29:42	6.217	4.9	6.446	6.767	3.975	5.75		11:00:40	0.48	0.251	0.317	0.269	0.307	0.42
10:29:57	8.044	8.02	5.92	7.653	6.369	7.29		11:00:45	0.194	0.253	0.309	0.191	0.179	0.32
10:30:12	6.683	7.445	6.076	8.251	4.943	6.77		11:00:50	0.051	0.458	0.605	0.188	0.365	0.43
10:30:27	8.129	4.939	7.15	6.822	5.774	6.65		11:00:55	0.056	0.541	0.429	0.238	0.147	0.37

References

[1] T.-R. Banniza, D. Boettle, R. Klotsche, P. Schefczik, M. Soellner, and K. Wuenstel, "A European Approach to a Clean Slate Design for the Future Internet," *Bell Labs Tech. J.*, vol. 14, no. 2, pp. 5–22, 2009.

[2] J. Carapinha and J. Jimenez, "Network Virtualization - a View from the Bottom," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, Barcelona, Spain, 2009, pp. 73–80.

[3] N. Niebert, I. Khayat, S. Baucke, R. Keller, R. Rembarz, and J. Sachs, "Network Virtualization: A viable path towards the future Internet," *Wirel. Pers. Commun.*, vol. 45, no. 4, pp. 511–520, 2008.

[4] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through network virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, 2005.

[5] R. Canonico, D. Emma, and G. Ventre, "An XML Description Language for Web-Based Network Simulation," in *Proceedings of the 7th IEEE International Symposium on Distributed Simulation and Real-Time Applications*, Delft, The Netherlands, 2008, pp. 76–81.

[6] F. Galan, D. Fernandez, W. Fuertes, M. Gomez, and J. Vergara, "Scenario-based virtual infrastructure management in research and educational testbeds with VNUML," *Ann. Telecommun.*, vol. 64, no. 5–6, pp. 305–323, 2010.

[7] K. Tutschku, P. Tran-Gia, and F. U. Andersen, "Trends in network and service operation for the emerging future Internet," *Int. J. Electron. Commun.*, vol. 62, no. 9, pp. 705–714, 2008.

[8] "GENI," *Exploring networks of the future*, 2013. [Online]. Available: http://www.geni.net/. [Accessed: 21-Apr-2013].

[9] M. Creeger, "Moving to the edge: a CTO roundtable on network virtualization," *Commun. ACM*, vol. 53, no. 8, pp. 55–62, 2010.

[10] IBM, "Virtual I/O Server," *Virtual I/O Server overview*, 2011. [Online]. Available: http://pic.dhe.ibm.com/infocenter/powersys/v3r1m5/index.jsp?topic=/iphb1/iphb1_vios_virtualioserveroverview.htm. [Accessed: 28-Jul-2013].

[11] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," University of California, Berkeley, Dept. Electrical Eng. and Comput. Sciences, UCB/EECS 28, 2009.

[12] D. Evans, "Beyond Things: The Internet of Everything Takes Connections to the Power of Four," *Cisco Systems*. [Online]. Available: http://blogs.cisco.com/ioe/beyond-things-the-internet-of-everything-takes-connections-to-the-power-of-four. [Accessed: 25-Nov-2015].

[13] "Network-agile multi-provisioned infrastructure for GENI and ExoGENI," *Network-agile multi-provisioned infrastructure for GENI and ExoGENI*, 2012. [Online]. Available: http://groups.geni.net/geni/wiki/EXOGENI. [Accessed: 10-Jan-2013].

[14] Distributed Management Task Force, "Common Information Model," Distributed Management Task Force, DMTF Specifications DSP0004, 2011. Available: http://www.dmtf.org/standards/cim/ [Accessed: 20-Jan-2016].

[15] Distributed Management Task Force, "DMTF," *DMTF Home*, 2013. [Online]. Available: http://www.dmtf.org/. [Accessed: 17-Sep-2013].

[16] TMForum, “Information Framework (SID),” *SID TM FORUM*, 2013. [Online]. Available: http://tmforum.org/InformationFramework/1684/home.html. [Accessed: 07-Apr-2013].
[17] J. Strassner, “DEN-ng: achieving business-driven network management,” in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Colorado Springs, USA, 2002, pp. 753–766.
[18] “The Autonomic Communications Forum,” 2012. [Online]. Available: http://www.autonomic-communication.org/. [Accessed: 10-Jan-2013].
[19] D. Kontoudis and P. Fouliras, “Modeling and managing virtual network environments,” in *Proceedings of the 17th PanHellenic Conference on Informatics (PCI ‘13)*, Thessaloniki, Greece, 2013, pp. 39–46.
[20] W. Fuertes, J. E. V. Lopez, F. Meneses, and F. Galan, “A generic model for the management of virtual network environments,” in *Proceedings of the IEEE (NOMS) Network Operations and Management Symposium*, Osaka, Japan, 2010, pp. 813–816.
[21] N. Agoulmine, *Autonomic Network Management Principles*. Elsevier Academic Press, 2010.
[22] P. Grosso, J. Van der Ham, J. Steger, A. Fekete, L. Lymberopoulos, C. Papagianni, Y. Kryftis, A. Chuang, and K. Wierenga, “Information models for virtualized architectures - NOVI,” EU 7th FP, EU Deliverable NOVI-D2.1-11.02.28-v3.1, Feb. 2011. Available: http://www.fp7-novi.eu/deliverables/doc_download/25-d21
[23] J. Van der Ham, P. Chrysa, S. Jozsef, M. Peter, K. Yiannos, P. Grosso, and L. Lymberopoulos, “Challenges of an information model for federating virtualized infrastructures,” in *5th International DMTF Academic Alliance Workshop on Systems and Virtualization Management: Standards and the Cloud*, Paris, France, 2011, pp. 1–6.
[24] ERCIM, “Network description tools and standards,” *ERCIM News*, vol. 77, no. 1, pp. 33–34, 2009.
[25] B. Fenner, “MIB index,” *MIB Index*, 2012. [Online]. Available: http://www.icir.org/fenner/mibs/mib-index.html. [Accessed: 25-Jan-2013].
[26] J. A. Garcia-Espin, J. F. Riera, E. Lopez, S. Figuerola, P. Donadio, G. Buffa, E. Escalona, S. Peng, S. Soudan, F. Anhalt, P. Robinson, A. F. Antonescu, A. Tzanakaki, M. Anastasopoulos, A. Tovar, J. Jimnez, X. Chen, C. Ngo, M. Chijsen, Y. Demchemko, G. Landi, L. Lopatowski, J. Gutkowski, and B. Belter, “Functional Description of the Logical Infrastructure Composition Layer (LICL) - GEYSERS deliverable D3.1,” 2010. Available: http://www.geysers.eu/images/stories/deliverables/geysers-deliverable_3.1.pdf
[27] Open Networking Foundation, “Software-Defined Networking: The New Norm for Networks,” White Paper, Apr. 2012.Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf
[28] K. Hyojoon and N. Feamster, “Improving network management with software defined networking,” *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114–119, 2013.
[29] A. Devlic, W. John, and P. Skoldstrom, “A Use-Case Based Analysis of Network Management Functions in the ONF SDN Model,” presented at the 2012 European Workshop on Software Defined Networking (EWSDN), Darmstadt, Germany, 2012, pp. 85–90.

[30] G. M. Giaglis, “A taxonomy of business process modeling and information systems modeling techniques,” *Int. J. Flex. Manuf.*, vol. 13, no. 2, pp. 209–228, 2001.
[31] J. Mylopoulos, “Information Modeling in the Time of the Revolution,” *Inf. Syst.*, vol. 23, no. 3–4, pp. 127–155, 1998.
[32] J. Strassner, *Policy-based Network Management: Solutions for the Next Generation*. Morgan Kaufman, 2003.
[33] J. Brendan and R. Stadler, “Resource management in clouds: Survey and research challenges,” *J. Netw. Syst. Manag.*, pp. 1–53, 2014.
[34] S. S. Manvi and G. K. Shyam, “Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey,” *J. Netw. Comput. Appl.*, vol. 41, pp. 424–440, 2014.
[35] S. Patil and D. Lilja, “Statistical methods for computer performance evaluation,” *WIREs Comput. Stat.*, vol. 4, no. 1, pp. 98–106, 2012.
[36] IBM, “IBM AIX operating system,” *IBM AIX, UNIX software for Power Systems*, 2013. [Online]. Available: http://www-03.ibm.com/systems/power/software/aix/index.html. [Accessed: 04-Oct-2013].
[37] “Temenos T24 Core Banking Software,” 2014. [Online]. Available: http://www.temenos.com/en/products-and-services/front-and-middle-office/t24-core-banking/ [Accessed: 29-May-2015]
[38] J. Lu, L. Makhlis, and J. Chen, “Measuring and Modeling the Performance of the Xen VMM,” presented at the Int. CMG Conference, Osaka, Japan, 2006, pp. 621–628.
[39] S. Jyotiprakash, S. Mohapatra, and R. Lath, “Virtualization: A survey on concepts, taxonomy and associated security issues,” in *Proceedings of the 2nd International Conference on Computer and Network Technology ICCNT*, Bangkok, Thailand, 2010, pp. 222–226.
[40] G. Shields, *The Shortcut Guide to Selecting the Right Virtualization Solution*. Realtimepublishers, 2007.
[41] W. Armstrong, R. Arndt, D. Boutcher, R. Kovacs, D. Larson, K. Lucke, N. Nayar, and R. Swanberg, “Advanced virtualization capabilities of POWER5 systems,” *IBM J. Res. Dev.*, vol. 49, no. 4, pp. 523–532, 2005.
[42] J. Szefer, E. Keller, R. B. Lee, and J. Rexford, “Eliminating the hypervisor attack surface for a more secure cloud,” in *Proceedings of the 18th ACM conference on Computer and communications security*, Chicago, Illnois, USA, 2011, pp. 401–412.
[43] M. McFaden, J. Schoenwaelder, T. Tsou, and C. Zhou, “Definition of Managed Objects for Virtual Machines Controlled by a Hypervisor,” IETF Draft draft-schoenw-opsawg-vm-mib-01, Jul. 2012. Available: http://tools.ietf.org/html/draft-schoenw-opsawg-vm-mib-01
[44] R. Chandramouli, “Security Recommendations for Hypervisor Deployment,” National Institute of Standards and Technology, Draft Special Publication 800-125 A, 2014. Available: http://csrc.nist.gov/publications/drafts/800-125a/sp800-125a_draft.pdf
[45] D. W. Anderson, F. J. Sparacio, and R. M. Tomasulo, “The IBM System/360 model 91: Machine philosophy and instruction-handling,” *IBM J. Res. Dev.*, vol. 11, no. 1, pp. 8–24, 1967.
[46] M. F. Mergen, V. Uhlig, O. Krieger, and J. Xenidis, “Virtualization for high-performance computing,” *ACM SIGOPS Oper. Syst. Rev.*, vol. 40, no. 2, pp. 8–11, 2006.

[47] R. Ameen and A. Hamo, “Survey of Server Virtualization,” *Int. J. Comput. Sci. Inf. Secur.*, vol. 11, no. 3, 2013.
[48] N. M. M. K. Chowdhury and R. Boutaba, “A survey of network virtualization,” *Comput. Netw.*, vol. 54, no. 5, pp. 862–876, 2010.
[49] N. M. M. K. Chowdhury and R. Boutaba, “Network Virtualization: State of the Art and Research Challenges,” *IEEE Commun. Mag.*, vol. 47, no. 7, pp. 20–26, 2009.
[50] D. L. Tennenhouse and D. J. Wetherall, “Towards an active network architecture,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 5, pp. 81–94, 2007.
[51] A. T. Campbell, H. G. D. Meer, M. E. Kouvaris, K. Miki, J. B. Vicente, and D. Villela, “A Survey of Programmable Networks,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 2, pp. 7–23, 1999.
[52] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, “Resilient Overlay Networks,” *ACM SIGOPS Comput. Commun. Rev.*, vol. 32, no. 1, pp. 66–66, 2001.
[53] J. Touch and S. Hotz, “The X-Bone,” in *Proceedings of the IEEE 3rd Global Internet Mini-Conference in conjunction with GLOBECOMM98*, Sydney, Australia, 1998.
[54] N. M. M. K. Chowdhury, F. Zaheer, and R. Boutaba, “iMark: An Identity Management Framework for Network Virtualization Environment,” in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*, Long Island, New York, USA, 2009, pp. 335–342.
[55] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy, “Network Virtualization Architecture: Proposal and Initial Prototype,” in *Proceedings of the 1st ACM workshop on Virtualized Infrastructure Systems and Architectures*, Barcelona, Spain, 2009, pp. 63–72.
[56] M. Boucadair, P. Georgatsos, N. Wang, D. Griffin, G. Pavlou, M. Howarth, and A. Elizondo, “The AGAVE approach for network virtualization: differentiated services delivery,” *Ann. Telecommun.*, vol. 64, no. 5–6, pp. 277–288, 2009.
[57] N. Feamster, L. Gao, and J. Rexford, “How to lease the Internet in your spare time,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 61–64, 2007.
[58] K. Oberle, M. Kessler, M. Stein, T. Voith, D. Lamp, and S. Berger, “Network virtualization: the missing piece,” in *Proceedings of the IEEE 13th International Conference on Intelligence in Next Generation Networks*, Bordeaux, France, 2009, pp. 1–6.
[59] “IRMOS Project,” *Interactive Realtime Multimedia Applications on Service Oriented Infrastructures*, 2011. [Online]. Available: http://www.irmosproject.eu/. [Accessed: 29-May-2013].
[60] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, “Survey on Network Virtualization Hypervisors for Software-Defined Networking,” arXiv preprint arXiv:1506.07275, 2015.
[61] R. Jain and S. Paul, “Network virtualization and software defined networking for cloud computing: a survey,” *IEEE Commun. Mag.*, vol. 51, no. 11, pp. 24–31, 2013.
[62] M. F. Bari, R. Boutaba, R. Esteves, and L. Z. Granville, “Data Center Network Virtualization: A Survey,” *IEEE Commun. Surv. Tutor.*, vol. 15, no. 2, pp. 909–928, 2013.

[63] J. Martins, M. Ahmed, C. Raiciu, V. Oltenau, M. Honda, R. Bifulco, and F. Huici, “ClickOS and the Art of Network Function Virtualization,” in *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, Seattle, WA, USA, 2014, pp. 459–473.
[64] The Linux Foundation, “XEN Hypervisor,” 2013. [Online]. Available: http://www.xenproject.org/. [Accessed: 10-Oct-2013].
[65] M. Fenn, M. Murphy, J. Martin, and S. Goasguen, “An evaluation of KVM for use in cloud computing,” in *Proceedings of the 2nd International Conference on the Virtual Computing Initiative (ICVCI ’08)*, Durham, NC, 2008.
[66] ETSI.org, “Leading operators create ETSI standards group for network functions virtualization,” Sep-2013. [Online]. Available: http://www.etsi.org/ index.php/news-events/news/ 644-2013-01-isg-nfv-created.
[67] VMware, “VMware vSphere Hypervisor (ESXi),” 2013. [Online]. Available: http://www.vmware.com/products/vsphere-hypervisor. [Accessed: 30-May-2013].
[68] IBM, “Server virtualization with IBM PowerVM,” *Server virtualization with IBM PowerVM*, 2013. [Online]. Available: http://www-03.ibm.com/systems/power/software/virtualization/. [Accessed: 03-Oct-2013].
[69] Oracle, “Oracle VM Server for x86,” White Paper, Aug. 2011. Available: http://www.oracle.com/us/technologies/virtualization/oraclevm/overview/index .html
[70] Q. Duan, Y. Yan, and A. Vasilakos, “A Survey on Service-Oriented Network Virtualization Toward Convergence of Networking and Cloud Computing,” *IEEE Trans. Netw. Serv. Manag.*, vol. 9, no. 4, pp. 373–392, 2012.
[71] C. E. Perkins, “Mobile IP,” *IEEE Commun. Mag.*, vol. 35, no. 5, pp. 84–89, 2002.
[72] K. Kroeker, “The evolution of virtualization,” *Commun. ACM*, vol. 52, no. 3, pp. 18–20, 2009.
[73] W. Haerick, T. Wauters, C. Develder, F. Turck, and B. Dhoedt, “Transparent resource sharing framework for internet services on handheld devices,” *Ann. Telecommun.*, vol. 65, no. 7–8, pp. 419–432, 2010.
[74] Z. Qi, C. Lu, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, 2010.
[75] V. A. Danciou, “Bottom-up harmonisation of management attributes describing hypervisors and virtual machines,” in *Proceedings of the 5th International DMTF Alliance Workshop*, Paris, France, 2011, pp. 1–10.
[76] Microsoft Corporation, “Hyper-V WMI Provider,” Oct. 2013. Available: http://msdn.microsoft.com/en-us/library/cc136992%28v=vs.85%29.aspx
[77] VMware, “VMware CIM SMASH/Server Management API for ESXi 5.0.” 2012. Available: http://pubs.vmware.com/vsphere-50/index.jsp
[78] LibVirt Community, “Libvirt-CIM,” *LibVirt CIM provider*, 2013. [Online]. Available: http://libvirt.org/CIM/intro.html. [Accessed: 01-Apr-2013].
[79] J. Jiyong, H. Saeyoung, K. Jinseok, P. Sungyong, B. Seungjo, and C. W. Young, “A Performance Evaluation Methodology in Virtual Environments,” presented at the 7th IEEE International Conference on Computer and Information Technology (CIT 2007), Aizu-Wakamatsu, Fukushima, 2007, pp. 351–358.
[80] Open Management Group, “Unified Modeling Language (UML),” 2013. [Online]. Available: http://www.uml.org/. [Accessed: 29-Apr-2013].

[81] Distributed Management Task Force, “DMTF CIM Network Model whitepaper,” Distributed Management Task Force, White Paper DSP 0152, Dec. 2003. Available: http://www.dmtf.org/sites/default/files/standards/documents/DSP0152.pdf
[82] DMTF, “CIM System Virtualization Profile,” DMTF, Informational DSP2013, Nov. 2007. Available: http://www.dmtf.org/standards/published_documents/DSP2013_1.0.0.pdf
[83] DMTF, “CIM System Virtualization Model,” DMTF, Informational DSP2013, Nov. 2007. Available: http://www.dmtf.org/sites/default/files/standards/documents/DSP2013_1.0.0.pdf
[84] TMForum, “TM Forum Frameworx,” 2012. [Online]. Available: http://www.tmforum.org/TMForumFrameworx/1911/home.html. [Accessed: 10-Jun-2013].
[85] J. Strassner, “FOCALE – A Novel Autonomic Computing Architecture,” presented at the Latin American Autonomic Computing Symposium (LAACS), Campo Grande, MS, Brazil, 2006.
[86] D. Raymer, J. Strassner, E. Lehtihet, and S. Van der Meer, “End-to-End Model Driven Policy Based Network Management,” presented at the 7th IEEE International Workshop on Policies for Distributed Systems and Networks, London, Ont., 2006, pp. 4–70.
[87] J. Strassner, S. Van der Meer, D. O’Sullivan, and S. Dobson, “The Use of Context-Aware Policies and Ontologies to Facilitate Business-Aware Network Management,” *J. Netw. Syst. Manag.*, vol. 17, no. 3, pp. 225–284, 2009.
[88] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen, “Policy core information model, version 1 Specification,” IETF Network Working Group, RFC RFC3060, 2001. Available: http://www.ietf.org/rfc/rfc3060.txt
[89] J. Strassner, Y. Liu, M. Jiang, J. Zhang, S. Van der Meer, M. O Foghlu, C. Fahy, and W. Donnelly, “Modelling Context for Autonomic Networking,” presented at the IEEE Network Operations and Management Symposium, Salvador da Bahia, 2008, pp. 299–308.
[90] EU 7th FP, “The AUTOI project,” 2012. [Online]. Available: http://ist-autoi.eu. [Accessed: 19-Jan-2013].
[91] A. Bassi, “European Union Autonomic Internet (AutoI) project factsheet,” Factsheet, Mar. 2008. Available: http://ist-autoi.eu/wp-content/uploads/2012/10/projects-autoi-factsheet_en.pdf
[92] Y. Choi, J. Li, Y. Han, J. Strassner, and J. W. K. Hong, “Towards a Context-Aware Information Model for Provisioning and Managing Virtual Resources and Services,” in *Modelling autonomic communication environments (MACE ’10)*, 2010, vol. 6473, pp. 100–112.
[93] K. McCloghrie, D. Perkins, and J. Schoenwaelder, “Structure of Management Information Version 2 (SMIv2),” IETF Network Working Group, RFC RFC2578. Available: http://tools.ietf.org/html/rfc2578
[94] “IETF MIBs,” *The SimpleWeb*, 2013. [Online]. Available: http://www.simpleweb.org/ietf/mibs/index.php?sel=IETF. [Accessed: 14-Sep-2013].
[95] A. Bierman and K. Jones, “Physical Topology MIB,” RFC RFC2922, 2000. Available: http://gamay.tools.ietf.org/html/rfc2922
[96] K. McCloghrie and A. Bierman, “Entity MIB (Version 2),” Cisco Systems, RFC RFC2737, 1999. Available: http://tools.ietf.org/html/rfc2737

[97] C. Srinivasan, L. P. Bloomberg, A. Viswanathan, and T. Nadeau, "Multiprotocol Label Switching Traffic Engineering Management Information Base," Cisco Systems, RFC RFC3812, Jun. 2004. Available: http://www.ietf.org/rfc/rfc3812.txt
[98] W. Ng, D. Jun, R. Chow, R. Boutaba, and A. Leon-Garcia, "MIBlets: a practical approach to virtual network management," in *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management*, Boston, MA, USA, 1999, pp. 201–215.
[99] E. Stelzer, S. Hancock, B. Schliesser, and J. Laria, "Virtual Router Management Information Base Using SMIv2," Corona Networks, IETF Draft draft-ietf-l3vpn-vr-mib-04, 2005. Available: http://tools.ietf.org/html/draft-ietf-l3vpn-vr-mib-04
[100] T. Bray, J. Paoli, C. M. Sperberg-McQueen, F. Yergeau, and J. Cowan, "Extensible Markup Language (XML)," W3C, W3C Technical Report REC-xml11-20060816, Aug. 2006. Available: http://www.w3pdf.com/W3cSpec/XML/2/REC-xml11-20060816.pdf
[101] S. and N. E. (SNE) research group University of Amsterdam, "Network Description Language," 2013. [Online]. Available: http://www.science.uva.nl/research/sne/ndl. [Accessed: 10-Jan-2013].
[102] W3C, "World Wide Web Consortium," 2013. [Online]. Available: http://www.w3c.org. [Accessed: 19-Mar-2013].
[103] "Global Lamda Integrated Facility," *GLIF: the Global Lambda Integrated Facility*, 2013. [Online]. Available: http://www.glif.is/. [Accessed: 05-Jun-2013].
[104] System and Network Engineering Research Group, "Device description using NDL," *Multi layer NDL configuration files*. [Online]. Available: http://ndl.uva.netherlight.nl/config.html. [Accessed: 24-May-2013].
[105] J. Van der Ham, "Map creation with NDL," *Interactive GLIF Map*, 2013. [Online]. Available: http://staff.science.uva.nl/~vdham/NDL/googlemap.html. [Accessed: 23-Mar-2013].
[106] F. Dijkstra, "Multi-Layer Path Finding with NDL," *Multi-Layer Path Finding with NDL*, 2013. [Online]. Available: http://ndl.uva.netherlight.nl/Glif-Demo-Sep07/. [Accessed: 10-Jul-2013].
[107] NML Working Group, "Network Markup Language," 2013. [Online]. Available: http://www.ogf.org/gf/group_info/areasgroups.php?area_id=1. [Accessed: 19-Jan-2013].
[108] Open Grid Forum, "Open Grid Forum," *Open Grid Forum*, 2010. [Online]. Available: http://www.ogf.org/. [Accessed: 05-Jul-2013].
[109] NML Working Group, *Network Markup Language Working Group*. 2010. Available: https://forge.gridforum.org/sf/projects/nml-wg
[110] J. Van der Ham, F. Dijkstra, R. Lapacz, and J. Zurawski, "Network Markup Language Base Schema version 1." Open Grid Forum, 2012. Available: http://www.ogf.org/gf/group_info/areasgroups.php?area_id=1
[111] Rice University, "The Virtual Grid Description Language: vgDL," 2009. [Online]. Available: http://vgrads.rice.edu/publications/vgdltr. [Accessed: 11-Mar-2013].
[112] L. Ramakrishnan, C. Koelbel, Y. S. Kee, R. Wolski, D. Nurmi, D. Gannon, G. Obertelli, A. YarKhan, A. Mandal, T. M. Huang, K. Thyagaraja, and D. Zagorodnov, "VGrADS: Enabling e-Science Workflows on Grids and Clouds with Fault Tolerance," presented at the SC09 High Performance Computing

Networking, Storage and Analysis Conference, New York, USA, 2009, pp. 1–12.
[113] Rice University, “vgDL Applications,” 2009. [Online]. Available: http://vgrads.rice.edu/research/applications/. [Accessed: 29-Jan-2013].
[114] G. P. Koslovski, P. V. B. Primet, and A. S. Charao, “VXDL: Virtual Resources and Interconnection Description Language,” *Netw. Grid Appl.*, vol. 2, no. 1, pp. 138–154, 2009.
[115] F. Anhalt, G. Koslovski, and P. V. B. Primet, “Specifying and provisioning virtual infrastructures with HIPerNET,” *Int. J. Netw. Manag.*, vol. 20, no. 3, pp. 129–148, 2010.
[116] M. Ghijsen, J. Van der Ham, P. Grosso, and C. De Laat, “Towards an Infrastructure Description Language for Modeling Computing Infrastructures,” presented at the 10th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA 2012), Leganés, Madrid, 2012, pp. 207–214.
[117] “EU Geysers Project.” [Online]. Available: http://www.geysers.eu/. [Accessed: 28-May-2013].
[118] L. Volker, D. Martin, I. Khayaty, C. Werle, and M. Zitterbart, “A Node Architecture for 1000 Future Networks,” in *Proceedings of the Intenational Conference on Communications Workshops*, Dresden, Germany, 2009, pp. 1–5.
[119] “4WARD,” *The FP7 4WARD Project*, 2011. [Online]. Available: http://www.4ward-project.eu/. [Accessed: 03-Jul-2013].
[120] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, “A survey of information-centric networking,” *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, 2012.
[121] “EU Scalable and Adaptive Internet Solutions (SAIL) Project,” *Scalable and Adaptive Internet Solutions (SAIL)*, 2012. [Online]. Available: http://www.sail-project.eu/. [Accessed: 10-Sep-2013].
[122] I. Fajjari, M. Ayari, and G. Pujolle, “VN-SLA: A Virtual Network Specification Schema for Virtual Network Provisioning,” in *Proceedings of the IEEE 9th International Conference on Networks*, Menuires, France, 2010, pp. 337–342.
[123] H. Okita, M. Yoshizawa, T. Suzuki, and T. Iijima, “Virtual Network Management Information Model (VNMI),” Hitachi Ltd., IETF Draft draft-okita-ops-vnetmodel-05, 2011. Available: http://tools.ietf.org/html/draft-okita-ops-vnetmodel-05
[124] R. Wang, D. Butnariou, and J. Rexford, “OpenFlow-based server load balancing gone wild,” in *Proceedings of the Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, Boston, MA, USA, 2011, pp. 12–12.
[125] M. Casado, T. Koponen, R. Ramathan, and S. Shenker, “Virtualizing the Network Forwarding Plane,” in *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*, Philadelphia, USA, 2010.
[126] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, “Onix: A Distributed Control Platform for Large-scale Production Networks,” in *Proceedings of the 9th Usenix Conference on Operating Systems Design and Implementation*, Vancouver, BC, Canada, 2010, vol. 10, pp. 1–6.

[127] C. Rotsos, R. Mortier, A. Madhavapeddy, B. Singh, and A. Moore, “Cost, performance & flexibility in OpenFlow: Pick three,” in *Proceedings of the SDN 2012 Workshop on Software Defined Networks*, Ottawa, Canada, 2012, pp. 6601–6605.
[128] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: Enabling Innovation in Campus Networks,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 67–94, 2008.
[129] R. Enns, “NETCONF Configuration Protocol,” Juniper Networks, NWG RFC RFC4741, 2006. Available: http://www.ietf.org/rfc/rfc4741.txt
[130] M. Bjorklund, “YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF),” Tail-f Systems, IETF RFC RFC6020, 2010. Available: http://tools.ietf.org/html/rfc6020
[131] Cisco Systems, “What is OverDrive Network Hypervisor?,” OL-24904-01, 2011. Available: http://www.cisco.com/en/US/docs/net_mgmt/datacenter_mgmt/OverDrive/4.0/Introducing/intro_overview.html
[132] Extreme Networks, “SDN: Software Defined Networking,” 2013. [Online]. Available: http://www.extremenetworks.com/solutions/datacenter_sdn.aspx. [Accessed: 12-Apr-2013].
[133] A. Gemino and Y. Wand, “A framework for empirical evaluation of conceptual modeling techniques,” *Requir. Eng.*, vol. 9, no. 4, pp. 248–260, 2004.
[134] “Advanced architecture for INTER-domain quality of service MONitoring and visualization,” *IST Intermon*, 2012. [Online]. Available: http://www.ist-intermon.org/. [Accessed: 03-Jun-2013].
[135] H. Asai, Y. Sekiya, K. Shima, and H. Esaki, “Management Information Base for the Virtual Machine Manager,” IETF Draft draft-asai-vmm-mib-00, 2013. Available: http://tools.ietf.org/html/draft-asai-vmm-mib-00
[136] “Network management RFCs sorted by SMI/MIB,” *IETF Network management modules*, 2013. [Online]. Available: http://www.simpleweb.org/ietf/rfcs/rfcbymodule.php. [Accessed: 28-Apr-2013].
[137] ITU-T, “Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation,” International Telecommunication Union, ITU-T Recommendation X.680, Jul. 2002. Available: http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf
[138] DMTF, “CIM Policy whitepaper,” DMTF, Informational DSP0108, Jun. 2003. Available: http://dmtf.org/documents/cim-policy-white-paper-270
[139] J. M. Tseng, H. L. Lee, J. W. Hu, T. L. Liu, J. G. Chang, and W. C. Huang, “Network Virtualization with Cloud Virtual Switch,” presented at the IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS), Tainan, 2011, pp. 998–1003.
[140] Distributed Management Task Force, “CIM Metrics Model whitepaper,” Distributed Management Task Force, White Paper DSP0141, Jun. 2003. Available: http://www.dmtf.org/sites/default/files/standards/documents/DSP0141.pdf
[141] Open Management Group, “Meta Object Facility (MOF),” *OMG’s MetaObject Facility*, 2012. [Online]. Available: http://www.omg.org/mof/. [Accessed: 22-Apr-2013].

[142] N. Huber, M. Von Quast, M. Hauck, and S. Kounev, “Evaluating and modeling virtualization performance overhead for cloud environments,” in *Proc. of the 1st International Conference on Cloud Computing and Services Science*, Noordwijkerhout, The Netherlands, 2011, pp. 563–573.
[143] A. Scordaki and S. Psarakis, “Statistical Process Control in Service Industry - an Application with Real Data in a Commercial Company,” in *Proc. 7th Hellenic European Conference on Computer Mathematics and Its Applications*, Athens, Greece, 2005.
[144] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, “Efficient Resource Provisioning in Compute Clouds via VM Multiplexing,” in *Proceedings of the 7th International Conference on Autonomic Computing*, 2010, pp. 11–20.
[145] Y. Diao, J. L. Hellerstein, S. Parekh, and R. Griffith, “A control theory foundation for self-managing computing systems,” *IEEE J. Sel. Areas Commun.*, vol. 23, no. 12, pp. 2213–2222, 2005.
[146] L. Chenyang, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son, “A feedback control approach for guaranteeing relative delays in web servers,” in *Proceedings of the 7th IEEE Real-Time Technology and Applications Symposium*, Taipei, 2001, pp. 51–62.
[147] T. F. Abdelzaher, K. G. Shin, and N. Bhatti, “Performance guarantees for web server end-systems: A control-theoretical approach,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 1, pp. 80–96, 2002.
[148] L. Ying and T. F. Abdelzaher, “Design, implementation, and evaluation of differentiated caching services,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 5, pp. 440–452, 2004.
[149] N. Gandhi and D. M. Tilbury, “MIMO control of an apache web server: Modeling and controller design,” in *Proceedings of the 2002 American Control Conference*, Anchorage, USA, 2002, vol. 6, pp. 4922 – 4927.
[150] Y. Zhang, A. Bestavros, M. Guirguis, I. Matta, and R. West, “Friendly virtual machines: leveraging a feedback-control model for application adaptation,” in *Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, Chicago, USA, 2005, pp. 2–12.
[151] P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Mechant, and K. Salem, “Adaptive control of virtualized resources in utility computing environments,” in *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems*, 2007, pp. 289–302.
[152] A. Mohit, P. Druschel, and W. Zwaenepoel, “Cluster reserves: a mechanism for resource management in cluster-based network servers,” *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 28, no. 1, pp. 90–101, 2000.
[153] M. Jones, D. Rosu, and M. C. Rosu, “CPU Reservations and Time Constraints: Efficient, Predictable Scheduling of Independent Activities,” *ACM SIGOPS Oper. Syst. Rev.*, vol. 31, no. 5, pp. 198–211, 1997.
[154] A. Chandra, W. Gong, and P. Shenoy, “Dynamic Resource Allocation for Shared Data Centers Using Online Measurements,” in *Proceedings of IWQoS 2003*, Monterey, CA, 2003, pp. 381–398.
[155] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle, “Managing energy and server resources in hosting centers,” *ACM SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 103–116, 2001.
[156] D. Breitgand and A. Epstein, “Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds,” in

Proceedings of the 2012 IEEE International Conference on Computer Communications, 2012, pp. 2861–2865.
[157] M. Bennani and D. Menasce, "Resource allocation for autonomic data centers using analytic performance models," in *2nd International Conference on Autonomic Computing*, 2005, pp. 229–240.
[158] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An analytical model for multi-tier internet services and its applications," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 291–302, 2005.
[159] A. Ali-Eldin, J. Tordsson, and E. Elmroth, "An adaptive hybrid elasticity controller for cloud infrastructures," in *13th IEEE Network Operations and Management Symposium*, 2012, pp. 204–2012.
[160] "AGILE: elastic distributed resource scaling for infrastructure-as-a-service," in *10th International Conference on Autonomic Computing*, San Jose, CA, 2013, pp. 69–82.
[161] R. S. Guh, J. D. T. Tannock, and C. O'Brien, "IntelliSPC: a hybrid intelligent tool for on-line economical statistical process control," *Expert Syst. Appl.*, vol. 17, no. 3, pp. 195–212, 1999.
[162] Z. Chen, S. Lu, and S. Lam, "A hybrid system for SPC concurrent pattern recognition," *Adv. Eng. Inform.*, vol. 21, no. 3, pp. 303–310, 2007.
[163] N. Boffoli, G. Bruno, D. Caivano, and G. Mastelloni, "Statistical process control for software: a systematic approach," in *Proc. of the 2nd ACM-IEEE international symposium on Empirical software engineering and measurement*, Kaiserslautern, Germany, 2008, pp. 327–329.
[164] T. H. Nguyen, B. Adams, Z. M. Jiang, A. E. Hassan, M. Nasser, and P. Flora, "Automated Detection of Performance Regressions Using Statistical Process Control Techniques," in *Proc. of the 3rd ACM/SPEC International Conference on Performance Engineering*, Boston, MA, USA, 2012, pp. 299–310.
[165] C. Lee, D. Lee, J. Koo, and J. Chung, "Proactive Fault Detection Schema for Enterprise Information System Using Statistical Process Control," in *Proc. of the Symposium on Human Interface 2009 on ConferenceUniversal Access in Human-Computer Interaction*, San Diego, Ca, USA, 2009, pp. 113–122.
[166] Y. Nong, S. M. Emran, L. Xiangyang, and C. Qiang, "Statistical process control for computer intrusion detection," in *Proc. of the DARPA Information Survivability Conference & Exposition*, Anaheim, CA, USA, 2001, vol. 1, pp. 3–14.
[167] Q. L. Zhang and J. Gao, "Applying SPC to autonomic computing," in *Proc. of the 2004 International Conference on Machine Learning and Cybernetics*, Shanghai, China, 2004, vol. 2, pp. 744–749.
[168] D. T. Pham and E. Oztemel, "XPC: an on-line expert system for statistical process control," *Int. J. Prod. Res.*, vol. 30, no. 12, pp. 2857–2872, 1992.
[169] R. R. Oliveira, A. A. Loureiro, and A. C. Frery, "A Multi-Scale Statistical Control Process for Mobility and Interference Identification in IEEE 802.11," *Mob. Netw. Appl.*, vol. 14, no. 6, pp. 725–743, 2009.
[170] R. R. Oliveira, R. R. Pereira, and A. A. Loureiro, "Adaptive configuration of wpans and wlans communications using multi-scale statistical process control," in *Proc. of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, Chania, Crete, Greece, 2007, pp. 138–142.
[171] N. M. Calcavecchia, B. A. Caprarescu, E. Di Nitto, D. J. Dubois, and D. Petcu, "DEPAS: A Decentralized Probabilistic Algorithm for Auto-Scaling," *Computing*, vol. 94, no. 8–10, pp. 701–730, 2012.

[172] F. Wuhib, R. Yanggratoke, and R. Stadler, “Allocating Compute and Network Resources under Management Objectives in Large-Scale Clouds,” *J. Netw. Syst. Manag.*, vol. 23, no. 1, pp. 111–136, 2015.
[173] T. Lorido-Botran, J. Miguel-Alonso, and J. Lozano A., “Auto-scaling Techniques for Elastic Applications in Cloud Environments,” University of the Basque Country, Technical Report EHU-KAT-IK-09-12, Sep. 2012.
[174] H. Ghanbari, B. Simmons, M. Litoiu, and G. Iszlai, “Exploring Alternative Approaches to Implement an Elasticity Policy,” in *Proceedings of the 2011 IEEE International Conference on Cloud Computing*, 2011, pp. 716–723.
[175] B. Wetherill and W. Brown, *Statistical Process Control: Theory and Practice*. Chapman and Hall, 1991.
[176] W. H. Woodall, “Controversies and Contradictions in Statistical Process Control,” *J. Qual. Technol.*, vol. 32, no. 4, pp. 341–350, 2000.
[177] D. J. Wheeler and D. S. Chambers, *Understanding Statistical Process Control*. Knoxville, TN.: SPC Press, 2010.
[178] B. Zhang, X. Wang, R. Lai, L. Yang, Y. Luo, and X. Li, “Evaluating and optimizing I/O virtualization in kernel-based virtual machine (KVM),” in *Network and Parallel Computing*, vol. 6289, Springer Berlin Heidelberg, 2010, pp. 220–231.
[179] Y. Dong, Y. Xiaowei, L. Jianhui, L. Guangdeng, T. Kun, and G. Haibing, “High performance network virtualization with SR-IOV,” *J. Parallel Distrib. Comput.*, vol. 72, no. 11, pp. 1471–1480, 2012.
[180] S. Bain, A. Bradfield, J. Guizan, J. Read, R. Rogers, J. Shedlersky, J. Thomas, and B. Willner, “A Benchmark Study on Virtualization Platforms for Private Clouds,” IBM SWG Competitive Project Office, ZSW03125-USEN-01, 2009.
[181] J. S. Oakland, *Statistical Process Control*. Butterworth-Heinemann, 2003.
[182] W. H. Woodall and D. C. Montgomery, “Research Issues and Ideas in Statistical Process Control,” *J. Qual. Technol.*, vol. 31, no. 4, pp. 376–387, 1999.
[183] S. Bersimis, S. Psarakis, and J. Panaretos, “Multivariate statistical process control charts,” *Qual. Reliab. Eng. Int.*, vol. 23, pp. 517–543, 2007.
[184] S. Vetter, M. Cordero, L. Correia, H. Lin, V. Thatikonda, and R. Xavier, *IBM PowerVM Virtualization Introduction and Configuration*. IBM, 2015.
[185] D. Kontoudis and P. Fouliras, “A CIM-based approach for managing computing servers and hypervisors acting as active network elements,” in *Proceedings of the International Federation for Information Processing (IFIP) 7th HET-NETs conference*, Bradford, UK, 2013. ISBN: 978-87-93102-32-3, Available: http://www.riverpublishers.com/pdf/ebook/RP_978-87-93102-32-3.pdf
[186] D. Kontoudis and P. Fouliras, “A survey of models for computer networks management,” *Int. J. Comput. Netw. Commun.*, vol. 6, no. 3, pp. 157–176, 2014.
[187] D. Kontoudis and P. Fouliras, “A Statistical Approach to Virtualized Server Resource Management” Under review in *Concurr. Comput. Pract. Exp.*, Wiley, ISSN 1532-0634
[188] D. Kontoudis, P. Fouliras, and C. Lambrinoudakis, “Statistics-driven Datacenter Resources Provisioning,” in *Proceedings of the 19th Panhellenic Conference on Informatics, PCI’15*, Athens, Greece, 2015, pp. 185–190.
[189] D. Kontoudis and P. Fouliras, “Host-based Virtual Networks Management in Cloud Datacenters,”, Accepted for publ. in *Comput. Inform.*, ISSN 1335-9150