Monitoring a virtual network infrastructure

An IaaS perspective

Augusto Ciuffoletti *
Department of Computer Science — University of Pisa — Italy
augusto@di.unipi.it

ABSTRACT

Infrastructure as a Service (IaaS) providers keep extending with new features the computing infrastructures they offer on a *pay per use* basis. In this paper we explore reasons and opportunities to include networking within such features, meeting the demand of users that need composite computing architectures similar to Grids.

The introduction of networking capabilities within IaaSs would further increase the potential of this technology, and also foster an evolution of Grids towards a confluence, thus incorporating the experiences matured in this environment.

Network monitoring emerges as a relevant feature of such virtual architectures, which must exhibit the distinguishing properties of the IaaS paradigm: scalability, dynamic configuration, accounting. Monitoring tools developed with the same purpose in Grids provide useful insights on problems and solutions.

Categories and Subject Descriptors

C.2.1 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Architecture and Design; C.2.3 [**COMPUTER-COMMUNICATION NETWORKS**]: Network operations—*network monitoring*

General Terms

Design

Keywords

GRID applications, network-enabled IaaS

1. INTRODUCTION

The *Cloud Computing* paradigm is characterized by resource virtualization, a feature that is appreciated both by the infrastructure manager and by the end user. In fact, the *provider* of virtual resources easily optimizes their management, while the *user* invests on resources exactly tailored on its needs. These facts are firmly established for computing and storage resources, so the number of providers that make available such resources is growing.

Conversely, on the side of networking resources the user is given little control over both the interconnection topology, and the parameters of networking resources. Yet, as we will show in this paper, technology is ready to provide flexible networking infrastructures on demand. So we envision an evolution of the providers of IaaS towards an offer that integrates a composite networking: the trigger for this evolution is the emergence of users interested in architectures more complex than those required by a simple two tiers web service.

A possible source for this trigger is a migration from computing Grids towards more cost effective computing models. The emergence of Grids has been fostered by the need of sharing resources in the domain of scientific investigation. This push has not exhausted its energy, but Grids are hindered by a functional model that features them as computing islands dedicated to support projects authorized by distinct scientific institutions: in such a framework, interoperability, accounting and security may even be considered as marginal, while they are fundamental in an industrial framework.

On the contrary Cloud computing, although originated from similar resource sharing purposes, gives precedence to features appreciated in the industrial milieu: accounting, security, and interoperability considered as marketing options. But Cloud products lack the flexibility of the Grid, since Cloud developers targeted the broadest demand, web servers and search engines, providing a straightforward, yet rigid, interface for their implementation. In present IaaS infrastructures there is little or no way to keep under control data acquisition and displacement, distributed operation and many other features that are typical of Grids.

In this paper we conjecture that a possible meeting point of Grid and Cloud technologies, which are presently evolving independently despite a lot of efforts to merge them, may depend on the introduction of an highly configurable network in the infrastructure provided as a pay-per-use service.

AWS Amazon Web Services
API Application Programming Interface
ESB Enterprise Service Bus
IaaS Infrastructure as a Service
MVRP Multiple VLAN Registration Protocol
NIC Network Interface Card
QoS Quality of Service
RTP Real Time Protocol
SIP Session Initiation Protocol
SNMP Simple Network Management Protocol
SQL Structured Query Language
VoIP Voice over IP
VLAN Virtual Local Area Network
VM Virtual Machine

Table 1: Acronyms used in the paper

*Augusto Ciuffoletti is member of the ERCIM CoreGRID Working Group

Our conjecture is based on the observation that the complex workflows that characterize scientific applications are incompatible with the rigid IaaS networking. Enabling a flexible networking within an IaaS would probably motivate a migration of scientific applications, and Grids would promptly turn into into IaaS hosting infrastructures taking advantage of the accumulated know-how. It is worth noting that the technology that enables such step is already available, as discussed in the next section: so the process is just waiting for a trigger.

In this new framework the ability to control network operation is vital: the statement extends from the configuration of the virtual network, to the administrative control over its operation, and to the feedback provided to network-aware applications. In the paper, we compare network monitoring designs for real and virtual networks, with an eye to Grid legacy.

The next section concentrates on the ready technologies that enable a transition towards networking enabled IaaS. In section 3 we study the problems related with the control of virtual networking provided as a service. In section 3.1 we wrap up some conclusions about relevant design issues concerning network monitoring.

2. ENABLING TECHNOLOGIES

Technology is ready to offer virtualization also for networking resources: just like the *hypervisor* implements virtual workstations over powerful computing architectures, a Virtual Local Area Network (VLAN) synthesizes plain Ethernet networks over complex infrastructures.

IEEE802.1Q [8] is the standard that addresses the management of VLANs, by regulating the operation of the device that enables their implementation over a switched network. The history of IEEE802.1Q begins in 1998, and a revision was released in 2005: there is an intense activity around this standard, and several substandards are being developed.

We observe that the IEEE802.1Q protocol splits the Data Link layer into two tiers: the lower layer (that we call *infrastructure management*) implements flat broadcast networks that provide connectivity to the upper *user applications* layer. The relevant effect is that infrastructure administration concerns — including logistic issues and load balancing — are transparent to the end user, which appreciates a flat network abstraction.

The infrastructure management layer makes use of traffic engineering techniques to interconnect diverse networking technologies — from long haul links to LANs. The devices interconnecting network elements are called VLAN-aware bridges: they route MAC frames using a VLAN identifier included in each frame. Each network element is associated with one or more VLANs, and the circulation of frames is confined within network elements that participate in the indicated VLAN. Routing decisions that implement certain aspects of traffic engineering move from layer 3 (network) to level 2 (data link), with improved performance.

In figure 1 we show an example of a network split into three segments, each hosting three distinct Virtual LANs: we may imagine this configuration motivated by office logistics.

Each VLAN is separately manageable, and VLAN-aware bridges are informed about the quality of service associated with a given VLAN: they can be instructed to differentiate expedited traffic, like Voice over IP (VoIP), from other classes. The team working around IEEE802.1 is deeply concerned with audio/video streaming over bridged architectures, and such interest materializes in a specific task group, whose activity also covers the transport of timing information [11], and end-to-end resource reservation.

If we aim to overlay a virtual network requested by a user over the real networking fabric [3], we need to define a suitable mapping from network elements to VLANs. This data structure is used by bridges for frame level routing, and is defined as *Filtering Database* in IEEE802.1Q terms.

In principle the Filtering Database is compiled by network administrators; in practice this task is so delicate and difficult that a specific protocol is designed to collect user requests, and to diffuse appropriate Filtering Database updates to the concerned VLAN switches.

The existence of this protocol is central in our discussion, since it makes practically feasible the *on demand* management of a VLAN-based network, that we consider as a basic building block for a networking enabled IaaS.

Another relevant fact refers to the physical interface between the servers and the networking fabric. According with IEEE802.1Q the same physical interface device may be shared among different VLANs, thus implementing several distinct virtual interfaces: although they share the same hardware, they run in total isolation among each other, in order to ensure privacy and security.

So we finally come to the abstract concept of a virtual host attached through a virtual interface to a virtual LAN, which is configured on demand: the basic building block of a networking enabled IaaS.

3. NETWORK MONITORING IN THE AGE OF THE CLOUD

Currently, the limited role of network monitoring in IaaS offers is justified by the simplicity of the provided infrastructure. However, if we envision the evolution of IaaS towards the provision of complex networks, like those used in scientific experiments, we need to consider the presence of an adequate monitoring of the networking infrastructure. In fact, network monitoring is not an option for a production infrastructure: its administration must keep under control the utilization and the performance of the network, and applications take advantage of input from traffic sensors. One of the basic features of network monitoring in a networking-enabled IaaS is that the output should be adherent to the VLAN abstraction, and that the interface should allow access from the network administrator as well as from network-aware applications.

To investigate the issues that are found in the design of a network monitoring infrastructure offered as part of an IaaS, we need to identify the activities that take advantage of its outcomes, starting from the most popular: *load balancing*.

IaaS providers offer load balancing as a part of the service (a recent survey is in [1], and Amazon Web Services (AWS) solution is in [2]), but such preconfigured service is adapted to a limited number of use cases, and opaque to the user. Instead an effective load balancing strategy needs to keep under control the performance of all involved resources, included the network, in an application dependent way. However the basic architecture currently offered to the user — whose simplicity on the other hand is a reason of success of the *IaaS* concept — is opaque to the user application (as in the case of Amazon EC2 Web Service). It consists of a *two-tiers Web Service* (as shown in figure 2): an array of virtual *web servers* offers to the clients outside the cloud a service based on a *repository* which is possibly hosted in a virtual storage. The primary interest of the user application is that all web requests are serviced with a uniform bandwidth, and that traffic latencies from web servers to internal storage servers are also predictable: however this kind of feedback is not available.

We observe that the maintenance of uniform network performances would also improve the effectiveness of load balancing techniques. In fact, when network performance is not homoge-

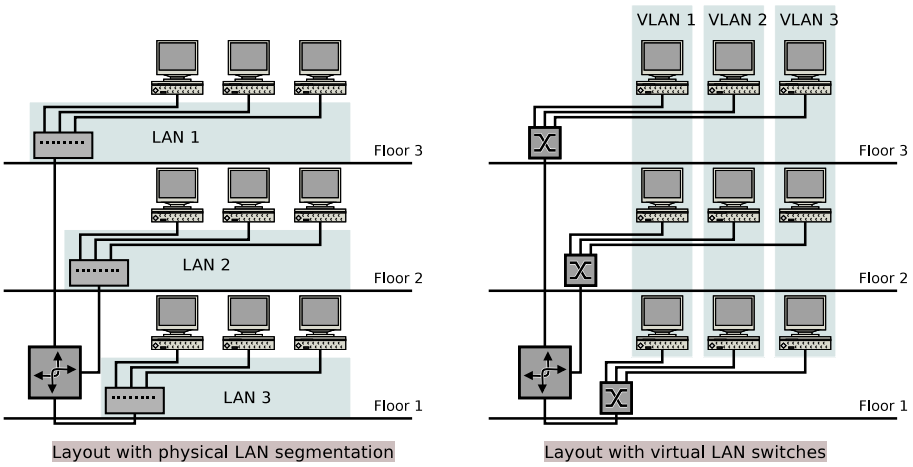

Figure 1: Two tiers view with and without LAN virtualization

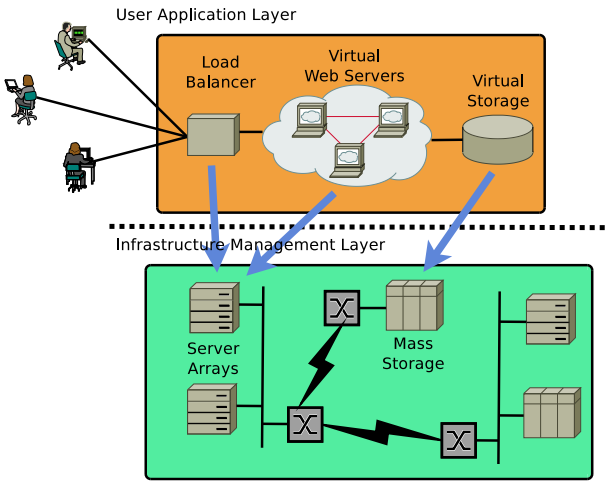

Figure 2: Infrastructure and user application views of a basic *elastic web service* infrastructure

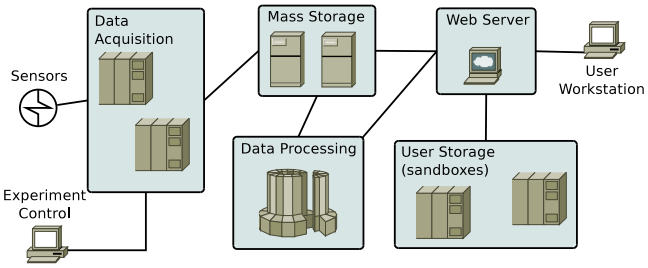

Figure 3: User application view of the computational infrastructure for a scientific experiment

neous, load balancing turns out to be ineffective, independently from the available processing power: a query may be penalized by the fact that the user — which sits outside the cloud — is poorly connected with the virtual web server, or that this latter experiences a high latency with the storage service.

The survey in [1] corroborates this conclusion with field experiments, and the benefits coming from the availability of network monitoring results at user application level become more evident when the use case is more demanding than the simple one described above.

For instance, consider the typical grid-like computational infrastructure for a scientific experiment (see figure 3). During its lifetime it will go through a sequence of phases: starting with the collection of experimental data, next preserving and making accessible these data, and occasionally supporting their processing.

Such a *life cycle* exhibits peaks of activity that are favorably managed using a cloud infrastructure. During the acquisition period, the storage infrastructure is stressed with a continuous stream of data: to contain buffer size, the network is required to introduce a sufficiently small jitter. Afterward, the data remains safely stored for a long period of time, available for successive processing sessions in order to extract significant scientific results. The latter activity is computationally intensive, and may use sophisticated parallelization techniques. The secure and reliable transport of data is persistently a key feature, since collected data, as well as the extracted figures, are considered a sensitive resource.

So, each of these phases shows distinct requirements from the point of view of the networking infrastructure. With a short digression, we note that the VLAN technology fits this framework, since it allows the flexible networking of a complex virtual infrastructure, preserving the differentiation of requests during the life cycle of a long lived project.

In complex cloud architectures like the one described above (see also a detailed description and discussion of a use case in [12]) it is reasonable that the IaaS provider offers a pre-configured monitoring service for better exploiting the provided service. But the collection of monitoring data that falls outside the direct control of the cloud provider, for instance for accounting third party users of the virtual infrastructure, can be effectively implemented only inside user applications.

A similar case exists for media streaming where, as a general rule, the application itself collects statistics for buffer optimization, or for other reasons related to the real time nature of the stream. In this case the IaaS provider cannot anticipate what kind of data are going to be helpful to the user.

So we identify many network sensitive activities: not only load balancing, but also data streaming, accounting, and control of parallel computations. All these activities are mostly under control of

the user, so we claim that, in addition to the results, also the *configuration* of network monitoring should be available to the user, and not restricted to a few Quality of Service (QoS) options offered by the IaaS provider.

3.1 Virtual devices for network monitoring

A number of Network Monitoring tools, for instance those that use `libpcap` [6], demonstrate that network performance can be effectively measured across network interface devices. In the IaaS approach these interfaces are bundled with Virtual Machine (VM) images: an apparently viable solution is to appoint the user with the task of configuring them, so to extract traffic patterns and characteristics. However this solution is of limited validity since, in order to monitor the virtual network, packet filtering rules have to contain data that are not available to the user. For instance this is the case of the VLAN identifier, which should not be manipulated by the user application for security reasons.

Monitoring the traffic across intermediate devices — like virtual switches in VMware [13] — is hardly applicable, since these devices are out of user's scope: the user that wants to inspect their operation and configuration, for instance using Simple Network Management Protocol (SNMP), has to rely on features implemented and made accessible by the IaaS provider. But the provider is unwilling to give user access to inter-networking devices, primarily for security reasons.

In this perspective, it is conceivable that the infrastructure management cooperates with the user in the configuration of virtual network interfaces, for instance providing network monitoring tools as configurable plug-ins of the virtual host. In the case of a simple web service, one such plug-in might measure the data transfer rate between the data servers and the web interface, the results being used to account the web service user for the quantity of data transferred.

At the infrastructure management level, such plug-ins are implemented using end-to-end monitoring between the physical network interfaces of the servers where user virtual machines are running: the data needed to implement packet filter rules are now accessed by the infrastructure management, but following user requests.

The option of monitoring user traffic in generic intermediate points has limits, since traffic among physical servers is trunked, and link aggregation is used. To avoid such drawbacks, network monitoring should be operated near or within the end-points, mainly using passive techniques embedded in virtual interfaces or switches.

In a nutshell, one building block of a networking-enabled IaaS is a virtual network device that provides traffic measurement on demand: for instance, in the data server example above, when one virtual server is added or removed to respond to load changes, the network monitoring activity is upgraded accordingly. Which is also consistent with cloud computing philosophy.

A suitable Application Programming Interface (API) has to be provided to allow the user application to have access to network monitoring. Through this gate the user interacts with network monitoring plug-ins that are dynamically loaded in VM images.

The interface should be as much transparent as possible to events that interfere with network monitoring consistency, so that the application may undertake the appropriate exceptional measures. Otherwise, such events may be a trigger for unstable behaviors. Consider that network monitoring implements the feedback control for resource management, as shown in figure 4: from control theory, we know that a badly controlled feedback may drive the whole system into instability. For an example of an event that perturbs measurement consistency, consider that in VMware™ the infrastructure administrator has the capability to displace a virtual host, changing its location in the physical network. In this case the network measurements collected before the displacement do not match with the situation afterward.

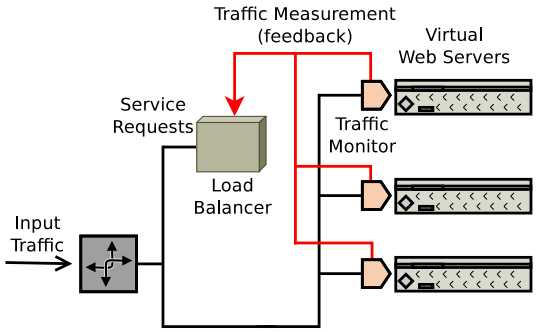

Figure 4: Closed loop load balancing with user level network monitoring (user application view)

One network monitoring aspect that is not very relevant at the user application level is the liveness and the reachability of virtual servers: the probability of host unreachability is drastically reduced by the infrastructure administration, which manages the underlying resources and may even relocate the virtual server in case of failure of the infrastructure.

3.2 Looking inside

Summarizing, software modules that implement virtual network interfaces play a significant role in the evolution of IaaS towards the provision of network infrastructures; such modules integrate knowledge accessible only within the infrastructure management level, with that provided by the user application. We now translate the conclusions of the above discussion in a series of guidelines for the design of a virtual network interface.

Considering the network monitoring infrastructure as a whole, we see that it consists of many software modules distributed throughout the IaaS infrastructure; virtual network interfaces are included in this number. The coordination of this complex infrastructure should be **distributed**, not concentrated in one server or registry. The rationale for a distributed management is primarily the resilience to network failures, but also better efficiency and an improved security. Virtual network interfaces participate to the distributed operation, and they are granted a way to identify themselves, and to communicate with other components.

Inside a virtual network interface the knowledge coming from the infrastructure management layer is used to map packet filtering rules from the virtual down to the real network. This part of the software component has privileged access to network management data not disclosed to the user, and participates in the distributed management of the network monitoring infrastructure.

On the user application side the network monitoring module is accessible through a **dedicated API**: it is used to configure the packet filtering rules and other details of the network monitoring activity. About this last point, we stress that a *by default* network monitoring is not viable: it is not realistic to assume that the IaaS provider collects all the network monitoring information a user might possibly ask.

The scope of a single network monitoring activity is bound to the traffic trunk allocated to the given user, in accordance with security and privacy requirements. This is the reason why monitoring is **end-to-end**, where ending points are the virtual network interfaces attached to the virtual hosts allocated to the user.

The virtual network monitoring infrastructure — the aggregation of virtual sensors and virtual connections that gather and transport

monitoring data — is **instantiated on demand**, not prepared in advance. The reason is that network monitoring requests cannot be flexibly anticipated. The IaaS provider allows the user application to indicate the plug-in that implements network monitoring and to describe traffic filters (including end-points, protocol etc.).

The **user interface** provided for this purpose should smoothly fit within the interface used for infrastructure configuration: we currently have a mature experience concerning the instantiation of elements that provide storage and processing capabilities. Standards are being elaborated in order to accelerate the evolution of services [10]. This experience should extend also to network elements, and to the attached monitoring services.

Monitoring activity outcomes are to be **delivered to the user**, not kept available to the infrastructure management: giving the user the responsibility of data management also ensures privacy. The results are returned to the user application either as a return value, or as a stream of measurements. As a consequence the destination of monitoring data is typically remote, and it is an option of the user to either consume the information instantly, or to store it for statistical purposes.

3.3 Grid legacy

Network monitoring infrastructures developed for Grids partially meet the above guidelines, and anticipate many of the issues found in IaaS infrastructures. In this section we focus on two of them and explore similarities and differences.

Monalisa [9] is a very successful product for network monitoring, which is diffused worldwide in production Grids. It implements a distributed infrastructure, and provides a sort of glue for binding together network monitoring tools developed independently. Instances of such tools are activated by administrators on specific subsystems and network measurements are collected and made available in an SQL database.

Monalisa addresses many of the issues introduced in this paper: there is a dynamic control over monitoring resources which are distributed across an infrastructure with a geographic scope and can be configured in a rather dynamic way.

However, the overall approach is that of a provider that configures the collection of network monitoring data, with a user that queries the infrastructure for available data: although the configuration and activation of network monitoring modules is dynamically controlled through an ad-hoc protocol, the generic user is not allowed to directly interact with them.

In a more dynamic environment, where networking changes following user requests, such a rigid management is a major drawback.

The Gd2 prototype [5] — at the stage of a *proof of concept* — introduces features that are closer to those illustrated in this paper: the *User Application Layer* of figure 5 shows the relevant components of this architecture.

In summary, a user application running on a *Host* has access to the monitoring infrastructure through a network of *Proxy* nodes. The overall system is partitioned into *Network Monitoring Domains*; each proxy manages the requests coming from one of them, and coordinates its own activity with other proxies. The primary role of a proxy is of discovering an *Agent* that can implement the request. The agent is the component with network monitoring capabilities; it is usually located close to networking devices (routers, gateways), and its capabilities are registered in the local proxy. A proxy extends its search for an appropriate agent across domains, by forwarding the requests to neighbor proxies. The agent discovery phase is managed using the SOAP protocol in a way that is similar to the Session Initiation Protocol (SIP) [7] protocol, and terminates when an appropriate agent is found, and the user request is delivered.

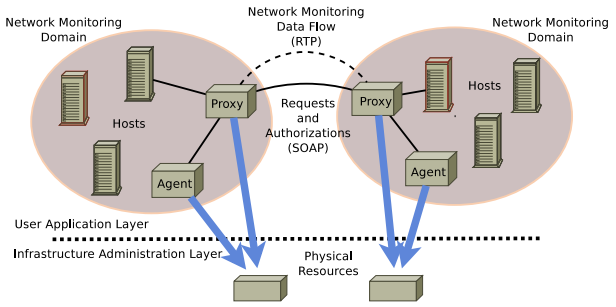

Figure 5: A simple case of gd2 monitoring infrastructure

The request contains all the data that are needed to setup a traffic monitoring filter, included the type of traffic, its endpoints, and the format to be used to return the data. When the monitoring activity starts, a data path is built across the network of Proxy components between the network monitoring agent and the user. At this point data start flowing as a unidirectional stream of Real Time Protocol (RTP) packets.

The Gd2 framework, which has been designed to provide network monitoring to complex workflows, matches many of the guidelines illustrated above. Its agents, like the infrastructure management component of virtual network interfaces, participate in the *distributed* management of the network monitoring infrastructure. The requests they receive allow them to *dynamically* setup the network monitoring activity, based on user application requests: authorization checks are performed at each step during the propagation of the request from the user application to the final agent. The networking monitoring data are *returned* to the requesting user application, and they may be encrypted, if this option is indicated in the network monitoring request.

Although Gd2 does not explicitly address virtualized resources, we may envision a transition path towards an IaaS monitoring infrastructure, as outlined by the blue arrows in figure 5.

The virtual network interface of a new virtual host is associated with a Gd2 agent: the measurement of traffic with an end on the new host is allocated to that agent. The agent waits for SOAP requests from the proxy; these requests are originated by possibly remote agents and acceptance criteria may be applied. The new virtual node will submit the requests from user applications to the local proxy. The translation of traffic specifications from the virtual infrastructure to the real infrastructure (using the VLAN mapping) takes place on the agents, where this information is available.

The location of agents and proxies depends on the logistics of the networking infrastructure: for instance, a *virtual rack* of hosts located inside the same server might be associated to an agent located in the same server, possibly corresponding to a virtual proxy in VMware terminology.

A more advanced perspective envisions network monitoring as embedded in the middleware that glues together the components of the virtual infrastructure. Network monitoring would be therefore considered as a part of the service provided to the user application, together with other sophisticated options like process migration, checkpointing and recovery. An essay in this direction has been put forward in the course of the CoreGRID European Program [4]. The questions here are how such advanced features are made effectively available, and how to design applications that take advantage of their presence: the Enterprise System Bus (ESB) concept is a suggestive paradigm in this direction.

4. CONCLUSIONS

The IaaS concept has the potential of evolving towards complex production infrastructures similar to Grids. Unlike a Grid, the deployment of a *networking-enabled IaaS* would take advantage of the basic *pay per use* axiom of cloud computing, with relevant financial consequences.

The components of a networking-enabled IaaS communicate through a virtual network, whose deployment is part of the provided service. The VLAN concept, and the related technology based on the IEEE803.1Q standard, is available for the synthesis on demand of complex, yet secure, virtual networks.

The provision of an effective monitoring of network operation is mandatory, since the user of a production infrastructure wants to be granted control over resources. To suit the IaaS concept, network monitoring must adapt to the changing needs of the user, be flexibly interfaced with the applications it uses or develops, ensure the desired degree of confidentiality: the design of an infrastructure that provides monitoring for a networking-enabled IaaS is regarded as a challenge that cannot be circumvented.

The components of this infrastructure coordinate among themselves to share the description of the real network, and of the virtual overlay. Although we understand the presence of other components that contribute to the implementation of the service, we concentrate our attention on the design of the virtual network interface, which is the software component in charge of giving a virtual host the access to the virtual network.

The virtual network interface is split into two subcomponents: one participating in the network infrastructure implementation and accessing the real network resources, the other providing a front end abstraction of the virtual networking through an API.

Through this API the user has access to the configuration and to the results of virtual network monitoring activities: sessions are dynamically created on request, and monitor end-to-end traffic among virtual servers.

All this shares many points with Grid monitoring infrastructures, especially when they are designed for the management of complex, network-aware workflows. So the experience gained in designing Grid monitoring infrastructures can help.

The relevance of the topic is explained considering that the IaaS model is financially more efficient than the Grid model: if clouds were able to host the complex infrastructures needed to host process workflows, the transition of many production infrastructures, included scientific projects, from Grids to Clouds might take place. In addition, this would open IaaS providers to a new market, and Grid infrastructures might be converted to networking-enabled IaaS providers, with improved flexibility. This paper sheds some light on concepts that may lead such step, and on technologies that may support it.

5. REFERENCES

[1] Brian Adler. Load balancing in the cloud: Tools, tips, and techniques. Technical report, RightScale, Inc., April 2010.

[2] Amazon Web Services LLC. *Amazon Elastic Load Balancing*, 2010.

[3] CISCO Systems. *Overview of Routing between Virtual LANs*.

[4] Augusto Ciuffoletti, Antonio Congiusta, Gracjan Jankowski, Michal Jankowski, Norbert Meyer, and Ondrej Krajicek. Grid Infrastructure Architecture - a modular approach from CoreGRID. In Joaquim Felipe and Jos Cordeiro, editors, *Third International Conference, WEBIST 2007, Barcelona, Spain, March 3-6, 2007, Revised Selected Papers*, volume 8 of *Lecture Notes in Business Information Processes*, pages 72–84. Springer, 2008. CoreGRID milestone M.IRWM.05.

[5] Augusto Ciuffoletti, Yari Marchetti, Antonis Papadogiannakis, and Michalis Polychronakis. Prototype implementation of a demand driven network monitoring architecture. In Sergei Gorlatch, Paraskevi Fragopoulou, and Thierry Priol, editors, *Grid Computing - Achievements and Prospects*, volume 9, chapter 8, pages 85–97. Springer, 2008.

[6] Luis Martin Garcia. Programming with libpcap - sniffing the network from our own application. *Hakin9 - Computer Security Magazine*, 2-2008(2-2008):9, 2008.

[7] H. Handley, H. Schultzrinne, Schooler E., and J. Rosenberg. SIP: Session initiation protocol. Request for Comment 2543, Network Working Group, March 1999.

[8] IEEE Computer Society. *IEEE Standard for Local and metropolitan area networks - Virtual Bridged Local Area Networks*, 2005.

[9] I.C. Legrand, H.B. Newman, R. Voicu, C. Cirstoiu, C. Grigoras, M. Toarta, and C. Dobre. Monalisa: An agent based, dynamic service system to monitor, control and optimize grid based applications. In *Computing in High Energy and Nuclear Physics (CHEP)*, Interlaken, Switzerland, September 2004.

[10] Open Grid Forum. *Open Cloud Computing Interface - Core & Models*, January 2010. Available from www.ogf.org.

[11] Michael D. Johas Teener and Geoffrey M. Garner. Overview and timing performance of IEEE 802.1AS. In *International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS)*, pages 22–26, Ann Arbor (MI), September 2008.

[12] Jinesh Varia. Cloud architectures. Technical report, Amazon Web Services, 2008.

[13] VMware. *VMware Virtual Networking Concepts*.