Computer Networks 77 (2015) 153–168

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Mantis: Cloud-based optical network planning and operation tool

Aristotelis Kretsis [a,*], Panagiotis Kokkinos [b], Kostas Christodoulopoulos [a], Theodora Varvarigou [b], Emmanouel (Manos) Varvarigos [a]

[a] University of Patras, Department of Computer Engineering and Informatics, Patras, Greece
[b] National Technical University of Athens, Department of Electrical and Computer Engineering, Athens, Greece

ARTICLE INFO

Article history:
Received 22 January 2014
Received in revised form 29 September 2014
Accepted 19 November 2014
Available online 17 December 2014

Keywords:
Network planning and operation
Flexible optical networks
Cloud computing

ABSTRACT

We present a network planning and operation tool, called Mantis, for designing the next generation optical networks, supporting both flexible and mixed line rate (MLR) WDM networks. Through Mantis, the user is able to define the network topology, current and forecasted traffic matrices, CAPEX/OPEX parameters, set up basic configuration parameters, and use a library of algorithms to plan, operate, or run what-if scenarios for an optical network of interest. Mantis is designed to be deployed either as a cloud service or as a desktop application. Using the cloud infrastructures features Mantis can scale according to the user demands, executing fast and efficiently the scenarios requested. Mantis supports different cloud platforms either public such as Amazon Elastic Compute Cloud (Amazon EC2) and ~okeanos or private based on OpenStack, while its modular architecture allows other cloud infrastructures to be adopted in the future with minimum effort. The included planning and operation algorithms range from routing and wavelength or spectrum allocation, to equipment (e.g. transponders and regenerators) placement, and CAPEX/OPEX/energy analysis.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

The continuous growth of IP traffic [1], fed by the generalization of broadband access (through DSL and FTTH) and the emerging rich-content high-rate and bursty applications, such as video-on-demand, HDTV and cloud computing, lead to our dependence on optical transport networks. For the future, it is expected that the traffic will not only increase in volume (34% increase on average per year [1]) but will also exhibit high burstiness, resulting in large variations over time and direction [2,3].

The most promising technology to meet the requirements of the next generation transport networks [3] appears to be elastic/flex-grid optical network. In particular, the following years will be a period of rapid changes in optical networking technology, as fixed-grid WDM networks are evolving into mixed line rate (MLR) networks, and will soon give way to flexible ("elastic", "adaptive", or "tunable") networks. Flexible optical networks assume the use of tunable transponders and a flexible spectrum grid or flex-grid. Flex-grid's granularity is much finer than that of standard WDM systems: the spectrum is divided into spectrum slots (12.5 GHz, as standardized by ITU G.694.1) that can be combined to create channels that are as wide as needed. Flexible optical networks rely also on tunable optical transponders, also called bandwidth variable transponders (BVT), which are able to adapt several transmission parameters, such as the modulation format, the spectrum utilized, and the transmission rate, according to the needs. So connections use as many spectrum slots as actually needed and thus the huge but still limited optical bandwidth is allocated in a more efficient

* Corresponding author.

http://dx.doi.org/10.1016/j.comnet.2014.11.015
1389-1286/© 2014 Elsevier B.V. All rights reserved.

manner, but also flexible networks enable the dynamic sharing of the network resources. In particular, spectrum-flexible networks (even without considering network adaptability) promise 30% improvement in spectrum utilization over fixed-grid WDM systems [4]. Actually, this figure does not take into account all the possible optimization dimensions of tunable transponders, e.g. reach vs rate adaptivity, and thus it can be even higher. Using spectrum-flexible technology optical core networks, which currently rely on the slowly changing circuit switching paradigm, become more dynamic and move towards the software defined networking paradigm. Moreover, the boundaries between core and metro networks will disappear and the technologies used in these two segments will be unified and merged. The flexible optical networks will span across both the core and the metro segments, enabling a fine-granular, cost- and power-efficient network able to carry a wide range of signal bandwidths that will vary in real time, in direction and magnitude. Network equipment vendors and operators are already looking into this technology.

A large number of algorithms have appeared in the literature for the planning and operation of fixed-grid or flex-grid optical networks, including algorithms for the Routing and Wavelength Assignment (RWA) or the Routing and Spectrum Allocation (RSA) problem, that take (or not) into account various parameters such as Physical Layer Impairments (that affect optical transmission reach), energy consumption, CAPEX and OPEX [5–7]. Other algorithmic issues include equipment placement and fault detection and handling. The number of proposed algorithms is so large that one applauds the effort, but questions the ability of researchers and/or network operators to identify the best ones and make use of them. Generally, it is often difficult to compare these algorithms against each other, under a common set of assumptions and parameters, reducing the importance and relevance of the related research.

We believe that a key enabler for the introduction of flexible technology will be the creation of network planning and operation tools that will help plan and manage these networks. Apart from the need to accommodate connections using spectrum slots in flexible networks, instead of wavelengths in WDM networks, and to choose the configuration of the transponders, an additional difference between these two networking solutions relates to the time scale at which the operation algorithms are called. This is expected to be reduced from handling a few requests per couple of months, as currently done in WDM networks, to handling requests at much shorter timescales that could go down to hour or even minute.

It is clear that the emergence of flexible networks, the use of optical technology across all network segments, the different operation timescales, and new application methodologies (clouds, SaaS, social by design) require not simply the extension of existing network planning tools, currently available from several major players, but the implementation of new ones. In this work, we present **Mantis** a network planning and operation tool for designing the next generation optical networks, in an effort to address the aforementioned shortcomings.

Mantis can be used as a researchers' tool for developing and evaluating, under common conditions, existing (included in Mantis) and new (added by users/researchers) algorithms for optical networks. Mantis can also be used by equipment vendors when evaluating their devices and by network operators when designing or extending their network, as a means to increase clients' satisfaction and decrease CAPEX (Capital expenditure) and OPEX (Operational expenditure) related costs. Moreover, another interesting direction is to interface Mantis to existing optical network management system (NMS) tools (Fig. 1), to provide path computation element (PCE) functionalities [19], that is, to make Mantis the logic of a functioning NMS/network.

Mantis is the first tool of its kind, academic or commercial, targeting flexible and mixed line rate WDM optical networks. Additionally, Mantis was designed and implemented so as to accommodate both desktop and cloud execution. In cloud mode, Mantis performance scales well with the demand, as opposed to past design tools available only as desktop applications. Mantis can concurrently utilize different cloud platforms either public such as Amazon Elastic Compute Cloud (Amazon EC2) and ~okeanos (the GRNET's – Greek National Research and Education Network – cloud service) or private based on OpenStack. We perform extensive evaluation of Mantis performance and scalability, using actual cloud resources, which highlight the tool's benefits.

Mantis has been briefly presented in [8,9]. Ref. [8] is an initial, short study of the Mantis tool, presenting a draft of the tool's design. Ref. [9] is a high level description of Mantis, discussing mainly the basic algorithmic issues in planning and operating flexible optical networks, highlighting the challenges and differences from fixed-grid WDM networks. In the present paper, we substantially extend our previous works. In particular, we provide a more detailed analysis of existing planning and operation tools. We describe the overall architecture of Mantis and explain the particular role of each component of the architecture. We also provide a better understanding of how Mantis can be used through the user interface and introduce the exposed programming interfaces. Mantis implementation characteristics are discussed for the first time, such as the software related technologies used and the modularity of the implementation that makes possible to add new capabilities (e.g., new algorithms). Moreover, we present in detail the cloud capabilities of the tool and provide for the first time information for the related components (information provider, dispatcher, etc.). We also perform extensive tests in order to evaluate Mantis performance.

The rest of the paper is organized as follows. In Section 2 we comment on existing network planning and operation tools and discuss the importance of such tools for spectrum-flexible networks. Section 3 presents Mantis' architecture and components. In Section 4 we describe Mantis operation as a service in the cloud. Section 5 describes the main algorithmic requirements of such a tool along with the particular algorithms currently included in Mantis. Section 6 presents Mantis user interface and elaborates on its usage. In Section 7, we evaluate Mantis desktop and cloud operation. Future work and our conclusions are given in Sections 8 and 9 respectively.

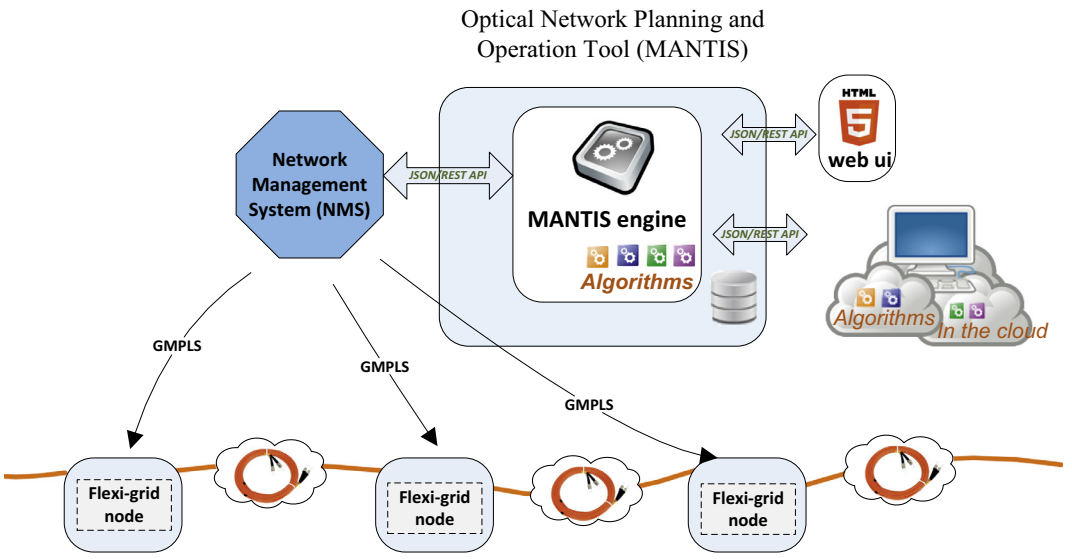

Fig. 1. Mantis – optical network planning and operation tool.

2. Network planning and operation tools

Planning and decision tools for optical networks have been a central research theme in the optical networking literature for the last decade and are also available in the commercial world. Fig. 2 gives a short list of such commercial tools, which are being used by network operators and equipment vendors to meet service level agreements, achieve capital expenditure savings, maximize network lifetime, and gain insight into the capabilities of their network.

Some of the most important functionalities these products provide are the following:

- Optimize routing and equipment placement to efficiently meet traffic demands.
- Produce equipment configuration and equipment requirements.
- Perform capacity planning, determining how to expand the network (e.g., purchase links) to handle traffic growth.
- Analyze the impact of failures and plan protection strategies to maximize resiliency.
- Analyse and minimize equipment costs.
- Evaluate various what-if networking scenarios, validating network changes or situations before deploying the production network.
- Perform traffic analysis and engineering.
- Visualize the above information.

These tools offer functionality for the IP and/or optical layers. Also, some tools support networking equipment from multiple vendors, while others only their proprietary products. In particular, Cariden MATE portfolio consists of a tightly integrated set of products (Design, Live, Collector) that support planning, engineering, and operational tasks

Tools	IP/MPLS capacity planning	IP/MPLS managing	Optical DWDM planning	Flexible optical networks
Cisco - Cariden	☑	☑		→
Cisco - Transport Planner			☑	
Opnet - NetOne Network Planner	☑			→
Opnet - NetOne Transport Planner			☑	
Wandl - IP/MPLS	☑	☑		→
Wandl NPAT			☑	
Aria - IP/MPLS-TE Operational Planning	☑	☑		→
Vpi - Multi-layer Transport Optimization			☑	
Infinera - Digital Network Administrator	☑	☑		
Infinera - Network Planning System			☑	→

Eventually will go towards this direction

Fig. 2. Major IP and WDM planning and operating tools overview.

for IP/MPLS networks [10]. OPNET's SP Guru Network Planner enables planning and design of multi-technology, multi-vendor IP/MPLS networks [11]. Opnet's SP Guru® Transport Planner [12] is a network planning solution that enables service providers and network equipment manufacturers to design resilient, cost-effective WDM, OTN, and SONET/SDH optical networks, using algorithms that minimize investment costs and optimize operational efficiencies. IP/MPLSView [13] is WANDL's multi-vendor, multi-protocol, and multi-layer solution for IP and/or MPLS networks, for design & planning, management & monitoring, and service creation & provisioning. NPAT (Network Planning & Analysis Tools) [14] is WANDL's solution for ATM, Frame Relay, TDM, Voice, and Optical Transport networks providing cross-vendor support for all stages of network planning, design, and analysis. Aria Networks IP/MPLS-TE Operational Planning solution [15] provides planning operation that analyze even the largest IP/MPLS networks. VPIsystems Multi-layer Transport Optimization solution [16] allows to visualize, understand and optimize transport networks, providing capacity analysis and planning, re-optimization, survivability analysis and greenfield planning. The Infinera Network Planning System (NPS) [20] provides users with offline graphical modeling, planning, and configuration capabilities for designing optical networks. Other companies (e.g., like Huawei, BTI, Alcatel-Lucent Bell, Nokia Siemens, Cisco) also offer similar solutions that are however more integrated with their products. Nokia Siemens Networks provides network providers with SURPASS TransNet and SURPASS TransConnect [17], in order to build efficient transport systems. The Cisco Transport Planner is a comprehensive WDM network design and design management tool. Cisco Transport Planner [18] uses optical transport technologies from the Cisco Optical portfolio.

Most existing commercial tools take the viewpoint that the network is rather static after its initial planning phase (except for handling failures, where again the protection/restoration actions to be taken are planned a priori, when the state of the network was probably different). Established connections (lightpaths) are almost never torn down, and new connections are added rather infrequently (e.g., a couple every few months). Planning is performed based on static traffic matrices (that estimate the traffic to be served for a given period between each pair of nodes) and even when impairment-aware RWA algorithms are used, that is algorithms that take into account the physical layer impairments, the effect of the impairments is evaluated based on analytic formulas and worst-case interference assumptions. Re-optimization very infrequently takes place, if ever, and the values of the hardware optical monitors play no role in these tools (were mainly used to notify failures/faults). All of these limitations were natural in a world where the WDM optical core network was oblivious to short/medium traffic changes.

Also, these tools offer functionality for the IP and optical WDM layers, most of the times separately. This is because changes in the two network layers occur at different time scales, as the WDM layer is considered to be rather static and to not really need operation at real time. The introduction of flex-grid technologies will force tools to reconsider these specifications, as the adaptability of this new architectural paradigm brings the optical layer closer to the IP layer. So, we can envision a scenario where the IP layer requests and controls the bandwidth that the flexible transponders use, meaning that operators no longer need to massively over-provision the optical core network to accommodate possible fluctuations in the IP layer. This implies that future tools will have to integrate more closely the IP and the optical core layers in order to achieve efficient resource utilization, including resources used for protection/restoration purposes, and save in capital and operational expenditures.

Finally, note that the tools in Fig. 2 are standalone applications and have very limited scalability when considering that most problems are computationally difficult (NP-hard), and also the problems will increase in size (convergence of metro and core), will become more complex (cross-layer design of IP and optical layers), and the timescales will decrease (more dynamic optical network to follow the IP layer).

3. Mantis implementation

Mantis network planning and operation tool targets flexible and mixed line rate WDM optical networks. Mantis was designed and implemented so as to accommodate both desktop and cloud execution.

3.1. Architecture

Mantis' components are organized in three layers: the *access layer*, the *application layer* and the *execution layer*. In addition, there are two common interfaces whose primary purpose is to provide loose coupling between the application layer and the other two layers. By using these interfaces we can have the same access and execution layers for both versions of the tool (desktop and cloud) while we can extend their functionality without breaking the implementation of the other components. Fig. 3 shows Mantis' architecture and its main components.

The *access layer* handles the interaction with the users through a web-based user interface and its exposed RESTful API. Through the Mantis' web-based interface users can have access to all tool functionalities, perform easily all the supported operations and collaborate with other users. Furthermore, a Python library that utilizes the RESTful API for communicating with the tool is available and provides almost the same functionality with the web-based interface. More interfaces to the Mantis functionalities can be added by extending the access layer to include a command-line interface (CLI) and to interface Mantis's algorithms to existing optical network management tools, providing path computation element (PCE) functionalities [19]. In other words, this will make Mantis the logic of a network management tool for a functioning network.

The *execution layer* consists of the execution engine and the library of available network planning and operation algorithms. Execution engine receives requests, for starting or terminating algorithm executions, through the common interface from the application layer and is responsible for

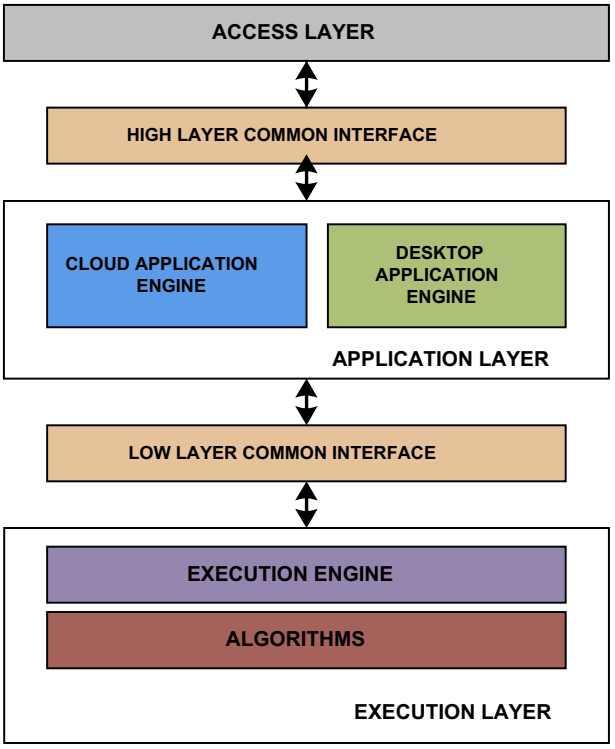

Fig. 3. Mantis general architecture and its main components.

performing all the required actions, including the preparation of the execution environment, the monitoring of the jobs' progress and the handling of the final results or possible failures. There is one execution engine when Mantis is deployed as desktop application, while at the cloud service deployment there is one execution engine at every computing node in the cloud infrastructure. When Mantis is deployed as desktop application, there is a server that contains the desktop application engine and the execution layer implementations (Fig. 4). The desktop application engine receives requests from the access layer and stores them in a local queue. Then the users' requests are forwarded to the local execution node. The desktop application engine is designed to limit the number of concurrent jobs based on the capabilities of the hosting machine in order to avoid resource overload. In particular, we have defined a default policy that limits the number of concurrently executed jobs to the number of cores of the hosting machine. Other policies can be defined, but the default policy is quite intuitive and straightforward.

The *application layer* implements the application logic and orchestrates the execution of user requests. It is the only layer that differs between cloud service and desktop application deployment as there are different requirements and operations that should be performed.

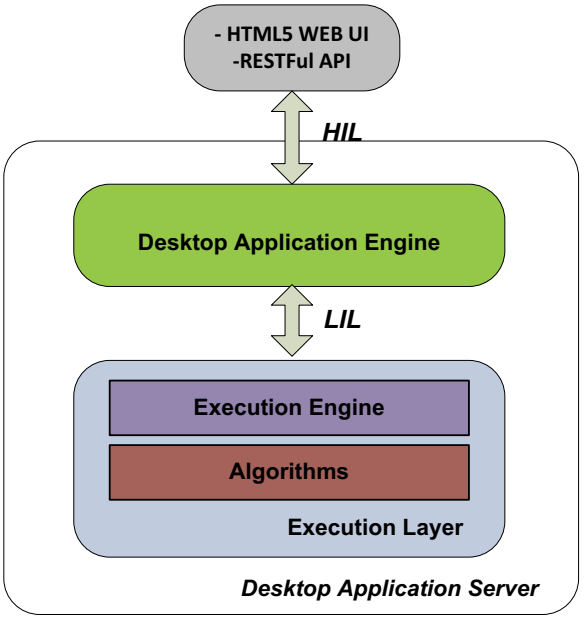

Fig. 4. Mantis as desktop application.

3.2. Plug-in mechanism

Mantis' algorithms are accessed from the execution engine through a custom plug-in mechanism. This mechanism enables new algorithms to be added in the tool without any modification in the application layer and the execution engine. Furthermore, through this mechanism the users will upload and plug their own algorithms into the Mantis core platform, to evaluate their performance and compare them with existing ones. This feature is not

yet available but will be included in the next version of the tool.

The execution engine's plug-in mechanism exposes a common interface, independent of the implementation technology, which determines explicitly the syntax of the input parameters and the results for all algorithms. In addition, the common interface uses the JavaScript Object Notation (JSON) as data-interchange format for providing the input and output parameters for all Mantis algorithms. In particular, every algorithm in Mantis returns the output results as JSON object and takes as input the following four parameters, using predefined JSON schemas, in the order they are presented: *network topology description*, *traffic demands description*, *algorithm specific parameters*, *CAPEX/OPEX parameters*. Fig. 5 shows an example for a network topology with four nodes and five unidirectional links along with its description in Mantis JSON schema.

3.3. Implementation technologies

For Mantis implementation we used software technologies and toolkits that are open source, mature, widely supported, have good performance and are able to scale in order to have a robust software system that fulfils all the design goals.

The access layer implementation is based on Rails [31] web application development framework, which is written in Ruby language. The web-based user interface is a client-side rich internet application based on the Dojo [32] JavaScript toolkit. Regarding the system database we have selected the PostgreSQL [33], an open source and robust object-relational database system that features all the necessary characteristics and is compatible with all the other implementation technologies. The cloud application engine is written in Python, with each of its components implemented as a Python module. The cloud application engine utilizes for its communication and interaction with the required Amazon Web Services (AWS) [34–36] the boto framework [37], an open source Python interface to AWS. In addition, for ~okeanos, the GRNET's public cloud service, Mantis' cloud application engine uses the available Synnefo API [38] for the compute, storage and network services. In execution layer the execution engine is written in C++ programming language using the cross-platform Qt [39] framework that is widely used for developing application software The Mantis algorithms are written in C++, or Java, or in Python/Cython [40] and are accessed from the tool through the execution engine's plug-in mechanism.

4. Mantis as a service

When Mantis is deployed as cloud service, the application layer implements the cloud application engine that handles the interaction with the cloud infrastructures. This engine harvests the power of the cloud so as to absolve the user from the burden of resource management, and provide scalability with respect to the number of users employing the tool and the number of scenarios evaluated simultaneously. To this end, it distributes the available jobs and executes them on multiple nodes in parallel in order to obtain the results of the executed algorithms in a reasonable amount of time.

Fig. 6 presents Mantis when deployed as cloud service. In this case, there is one cloud engine but multiple execution engines, one at every computing node in the available cloud resources. The cloud application engine has been designed to be modular in order to support multiple cloud platforms with minimum effort and changes. In the current version Mantis supports the public platforms Amazon Web Services [27], ~okeanos [28] GRNET's cloud service for the Greek Academic Community and private clouds based on OpenStack [29]. What is particularly interesting is that it is possible to use hybrid cloud infrastructures, a private cloud to serve the algorithms executions using the available local resources and additionally public cloud resources to cope with increased demands. Furthermore, the selected modular design of the cloud application engine enables the enhancement or upgrade of its building blocks without affecting the overall functionality of the tool.

The cloud application engine supports a number of configuration parameters that can be used to define the cloud computing services and their detailed characteristics that will be utilized for the algorithms executions. Cloud application engine can be configured to use either one or a combination of the supported cloud platforms. Moreover, for each cloud computing service it is possible to define the default machine types to be used, the shut-down behavior and the total available monthly budget (for paid services as is the case of the AWS).


```
{
    "number_of_nodes":4,
    "links": [
    {"nodes":[0,1],"length":280},
    {"nodes":[1,0],"length":280},
    {"nodes":[0,2],"length":280},
    {"nodes":[2,0],"length":280},
    {"nodes":[1,2],"length":250},
    {"nodes":[2,1],"length":250},
    {"nodes":[1,3],"length":340},
    {"nodes":[3,1],"length":340},
    {"nodes":[2,3],"length":340},
    {"nodes":[3,2],"length":340} ]
}
```


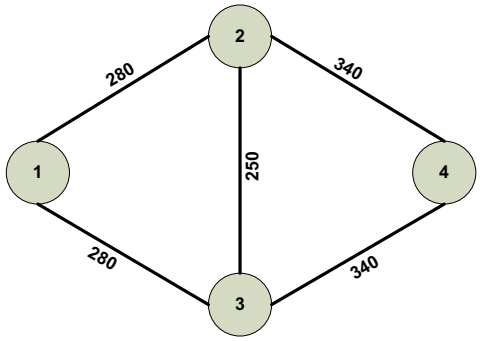

Fig. 5. Network topology description in Mantis.

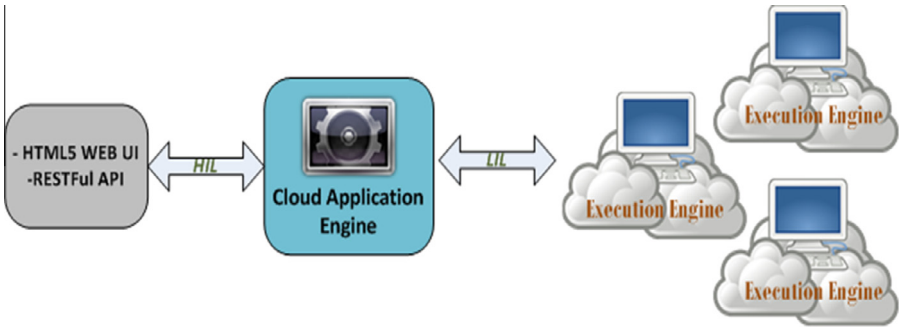

Fig. 6. Mantis as cloud service.

The cloud application engine consists of the following components: *the request and response queues*, *the information provider* and *the dispatcher* (Fig. 7). The cloud application engine receives requests from the access layer and stores them in the request queue and locally in a disk file. This provides a simple fault tolerance mechanism, eliminating the possibility of requests getting lost or not served due to network or cloud application engine problems. The response queue provides a central point where the available execution nodes send messages to the cloud application engine regarding the usage of their resources and the status of the executed jobs. These messages are then forwarded either to the dispatcher, if they contain information on the executed jobs, or to the information provider, if they describe the current status of an execution engine.

The information provider collects all the necessary details on the available cloud resources, their capabilities, their current load and the tasks assigned to each one for execution. The dispatcher is the main component of the cloud application engine, leveraging the available information from the information provider to perform a number of important operations regarding request/job and cloud resources handling. In particular, when the dispatcher receives a request for a new execution, initially it looks for some available execution node among the existing computation resources. If an execution node is available then the dispatcher will forward to it the new job, otherwise the dispatcher considers to request new computation resources from the cloud platforms.

One of the most useful features of cloud infrastructures is the ability to automatically scale an infrastructure vertically by changing the capacity of the used resources and horizontally by changing the number of the available resources to match changes in demand with little or no impact to the applications running in the infrastructure. The obvious benefit of cloud scaling is that we pay only

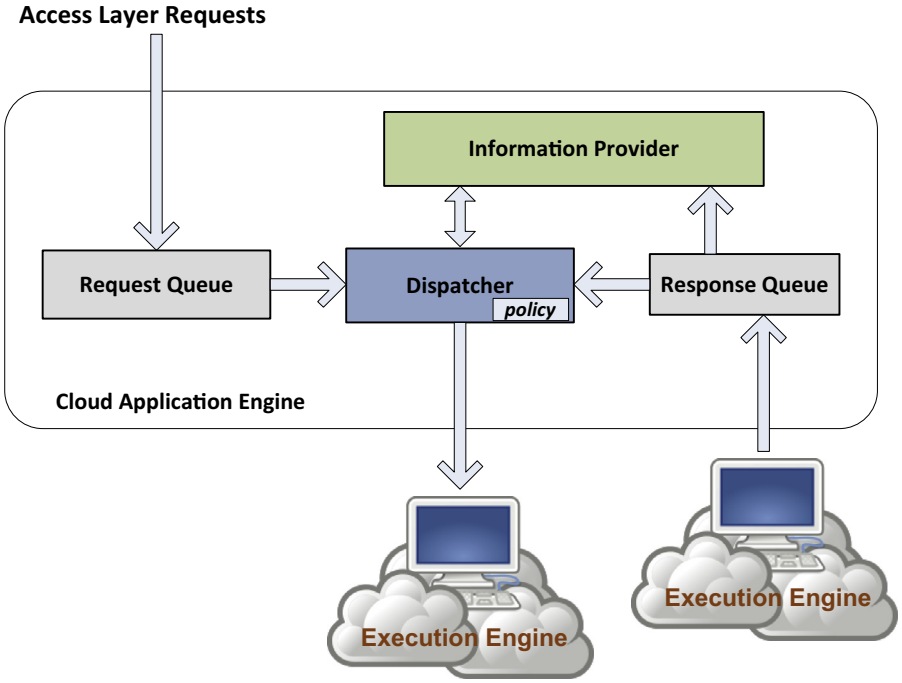

Fig. 7. Cloud application engine components.

for the resources we use. However, if someone does not take into consideration the capacity planning of his application then the cloud scaling can become a major issue. In this case, the over-reliance on scaling capabilities of the cloud infrastructures can lead a system to respond to demand by adding new cloud instances, without actually having a real performance benefit.

The dispatcher has a default allocation/de-allocation policy that determines how to scale the cloud platform by adding new resources or removing unused ones. Based on this policy, there are always available a minimum number of computational resources. The decision for allocating new resources is based on the number of queued requests, the current cost for the used resources and the total available budget, while these additional resources are de-allocated (removed) either immediately or after a period of time when the number of requests can be served from the default resources. The implementation of the cloud application engine enables more policies to be easily applied. In the future, we plan to apply more sophisticated policies based on the information available (from Mantis database) regarding the execution times of the different requests, leading to better allocation decisions.

Fig. 8 sketches a complete execution process in the cloud deployment. The arrows show the direction of the communication between the involved components and services.

5. Network issues and Mantis algorithms

5.1. Planning and operation

Today, several design and operation issues of optical networks are under investigation and re-evaluation from network operators, equipment vendors and researchers. Network design/planning, which typically occurs before a network is deployed, is focused on how to better accommodate the future network traffic, assuming that any equipment (transponders, regenerators) required can be purchased and deployed. In network operation phase, the demands are generally processed upon their arrival, one at a time or as a set, and it is assumed that the traffic must be accommodated using whatever equipment is already deployed in the network (or if allowed, new equipment can be purchased and deployed). Therefore, the operation process must take into account any constraints posed by the current state of the deployed equipment, which, for instance, may force a demand to be routed over a sub-optimal path.

5.2. Network resource assignment

One of the most important problems, both for planning and operation, is allocating network resources to traffic requests. The problem of establishing connections in fixed-grid WDM networks is typically referred to as the Routing and Wavelength Assignment (RWA) problem. The network consists of optical fibers and optical switches that add/drop local traffic and also forward traffic all-optically from an input to an output fiber (optical bypass). The spectrum is divided into wavelengths (50 or 100 GHz) and establishing a connection, called a lightpath, requires finding the route (path) and also assigning the wavelength to use. Different connections cannot use the same wavelength over a fiber and also a lightpath has to use the same wavelength over all fiber of its path, or it can change at the points where wavelength conversion or regeneration is performed. The basic RWA problem described above is NP-hard and there are many extensions/additional parameters that have to be accounted for in real optical networks, such as deciding the placement of transponders, accounting for physical layer impairments (signal quality deterioration) and regeneration placement, traffic grooming, etc., that make the problem quite complicated. Connection establishment in flexible networks is more complicated for several reasons. First, in contrast to WDM networks where each connection is assigned a single wavelength, in flexible networks spectrum slots can be

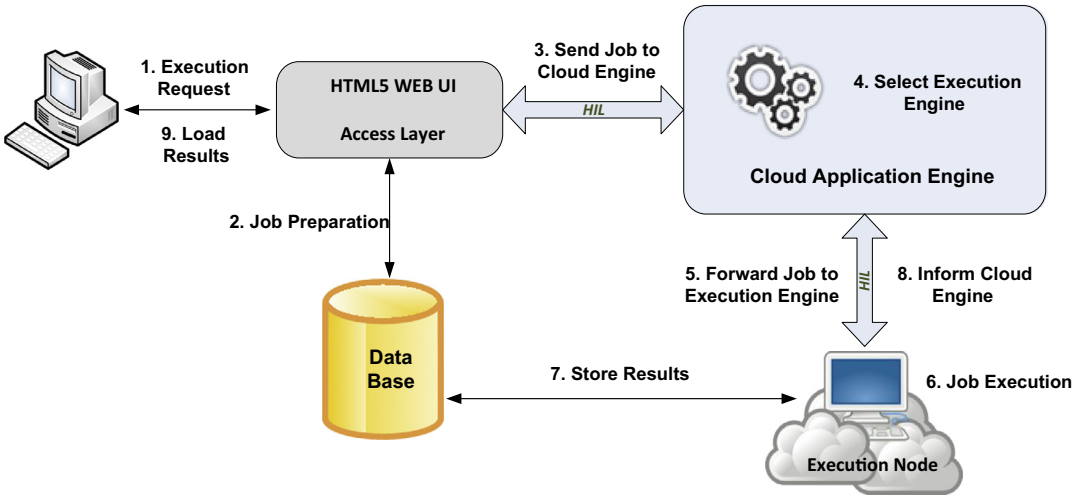

Fig. 8. Mantis execution sketch.

combined to form variable width channels, leading to the so-called Routing and Spectrum Allocation (RSA) problem. Apart from the difference in allocating spectrum resources, the choice of the transmission parameters of the tunable transponders present in flexible networks affect, directly or indirectly, the resource allocation decision and make the problem even more complicated [1], resulting in the Routing, Modulation Level and Spectrum Allocation (RMLSA) algorithms.

5.3. Physical layer impairments

Optical transport networks have evolved over recent years from opaque (point-to-point) to transparent networks, as a way to reduce CAPEX and OPEX costs. In the latter case, optical switches are configured to transparently handle transit traffic; the signal remains in the optical domain, bypassing the switch, saving on the cost of transponders used in the past to terminate and retransmit traffic at intermediate hops. Since optical connections may span over many and long links, physical layer impairments (PLIs), such as noise, dispersion, interference, and nonlinear effects accumulate and affect the quality of transmission (QoT). Accounting for PLIs is a challenge for algorithm designers, especially with respect to their exact modeling and the interdependencies introduced.

PLIs affect both fixed-grid WDM and flexible networks, but there are distinct differences between the two cases. With the introduction of coherent detection and DSP, impairments, particularly those related to dispersion, will be substantially reduced or fully compensated. However, the additional degrees of flexibility available in flexible networks make the minimization of these effects more complicated from an algorithmic perspective. On the other hand, physical layer impairments, even though more significant, can be accounted for quite accurately in WDM networks, where fewer parameters are involved (non-tunable transponders and constant guardband) and analytical models successfully capture these effects.

5.4. Energy issues

Even though ICT is already bringing massive environmental benefits (e.g., through the use of telecommuting, video conferencing, electronic news, etc.), the need to keep the related power consumption growth under control is also becoming evident. The continuing deployment and upgrade of optical telecommunication networks drive up power consumption, in a way that makes operators worry that future power consumption levels may pose constraints on the growth of telecommunications infrastructures. So it seems that an energy-aware approach is increasingly needed during the design, implementation, and operation of optical networks [22], which carry more than 80 percent of the world's long-distance traffic. Two different approaches can be explored to reduce power consumption in optical networks: the improvement of the energy efficiency of the equipment and the energy awareness of the algorithms used. In Mantis we focus on the second approach. Components in Mantis are characterized apart from their monetary cost, but also with their energy consumption, and detailed energy consumption models are included, so that each calculation performed by Mantis (establishing connections when planning or operating the network, etc.) reports also the energy-related cost. Energy-aware optimization algorithms have that target the minimization of such cost [25,26] are also included in the library of available algorithms, as will also be discussed in the next section.

5.5. Mantis algorithms

The current Mantis version includes a rather complete library of efficient network planning and operation algorithms for fixed-grid and flex-grid optical networks that can be used for both transparent (without regenerators) and translucent (with regenerators) networks.

The IA-RSA (IA stands for Impairment Aware) algorithm [21] considers the planning problem of a flexible optical network under physical layer impairments. It serves the demands for their requested rates by choosing the route, breaking the transmission in multiple parallel connections, placing regenerators if needed, and allocating spectrum to them. The IA-RWA-MLR and IA-RWA-SLR algorithms consider the planning problem of mixed line rate and single line rate fixed-grid optical networks under physical layer impairments, respectively. Physical layer effects are incorporated in the definition of the feasible transmission options of the transponders that can be used in each case. Online versions of these algorithms are also available that take as input the output of the offline case and serve new additional demands defined by the users.

Algorithms that consider in more details the physical layer in standard single line rate WDM networks, for which physical impairments analytical models are available, are the online IA-RWA [23] and the offline IA-RWA [24] algorithms. The online IA-RWA considers the routing and wavelength assignment problem of transparent optical networks and adopts a multi-cost approach that assigns a vector of cost parameters to each link, from which the cost vectors of candidate lightpaths are calculated. The lightpaths calculated by the aforementioned multi-parametric scheme are evaluated in terms of physical layer blocking using a function that combines these cost parameters. The offline IA-RWA algorithm [24] takes into account the physical impairments and the interference among lightpaths in its formulation and performs a cross-layer optimization between the physical (select lightpaths that have acceptable QoT) and the network (serve the connection requests using a small number of wavelengths) layers.

Furthermore, the EA-RWA (EA stands for Energy Aware) algorithms [25] aim at minimizing the energy consumed by the optical layer components when planning translucent WDM optical networks. The Joint-ILP algorithm solves the energy aware routing and wavelength assignment problem based on an integer linear programming (ILP) formulation that incorporates energy consumption and physical impairments. It jointly chooses the placement of the regenerators and the lightpaths to be used. On the contrary, the decomposition algorithm decomposes the problem into a regeneration placement problem and an EA-RWA problem for transparent networks, where each

sub-problem is addressed separately and sequentially. The decomposition technique uses a linear programming (LP) relaxation to address the problem in large scale networks. Details and performance evaluation of the algorithms included in the Mantis library can be found at the corresponding references.

6. Mantis user interface and usage

6.1. Mantis UI

A primary purpose of Mantis is to create a common benchmarking environment with social characteristics where researchers share topologies, traffic matrices and CAPEX/OPEX parameters, and also share and evaluate their algorithms under common conditions. In this way, Mantis could also evolve into an online collaboration platform for optical network researches, improving the comparability, quality and reliability of the results presented in various research articles and projects. Mantis current version can be found in [30].

Mantis comes with a clean and simple web-based user interface though which the users get access to all tool functionalities. In the user interface there is a separation of the different steps: network topology and traffic demands creation, algorithms selection and configuration definition, execution and results presentation. Fig. 9a and b shows the available interface for network topologies and traffic demands, respectively.

A new configuration, which defines a particular experiment–evaluation, can be created for each algorithm by selecting a network topology, the traffic demands and specifying all the other required parameters (Fig. 10a). Also, users can define the energy and monetary cost of various devices used in optical transport networks, such as transponders/muxponders, regenerators, amplifiers, switches. Algorithms use these values to calculate the monetary and energy cost for their solutions. Mantis automatically checks all provided parameters and informs the users of possible mistakes before permanently saving any configuration. Users can always check the status of their running or finished instances, a configuration under execution, and have access to useful details for all the instances including the instance name, configuration name, execution status, creation date and execution date (Fig. 10b). In addition, users can terminate running instances, view the results from successfully executed instances, or filter the displayed instances either by their execution status or their configuration name.

For every successfully executed instance the user can view analytic results and export the proposed solution for further analysis. Mantis is designed to report a detailed solution that, depending on the executed algorithm, may include: required bandwidth to serve the demands, established lightpaths, number and configurations for the required transponders and regenerators, placement of transponders and regenerators, total monetary cost and total power consumption, connections that could not be established due to physical layer impairments or bandwidth unavailability. Finally, users can create charts to visualize the results from various executions in order to have a better evaluation of the different scenarios.

6.2. Mantis Python library

The access layer exposes all Mantis functionality through a RESTful API that enables using Mantis directly over HTTP. The RESTful API can be useful for users who want to utilize Mantis services without using its web-based interface or who want to integrate their own tools and environments. All requests to the RESTful API are authenticated with HTTP Basic Authentication, which is based on the users' username and password in Mantis, while the responses are formatted in JSON (the default option) or in XML.

Also, we have developed a Python library that utilizes this RESTful API so that users can easily install it and interact programmatically with the tool, while the only requirement is to have a valid account in the online tool. Mantis Python library is composed of four main modules *traffic*, *topology*, *configuration* and *instance* and appropriate methods that handle the interaction with the corresponding entities in the tool. A user through the Mantis Python library can create, edit, delete, clone and make public or private network topologies, traffic demands and configurations. In addition, the library contains methods for filtering the returned information based on various parameters, monitoring the execution of all running instances, executing configurations and querying their current status.

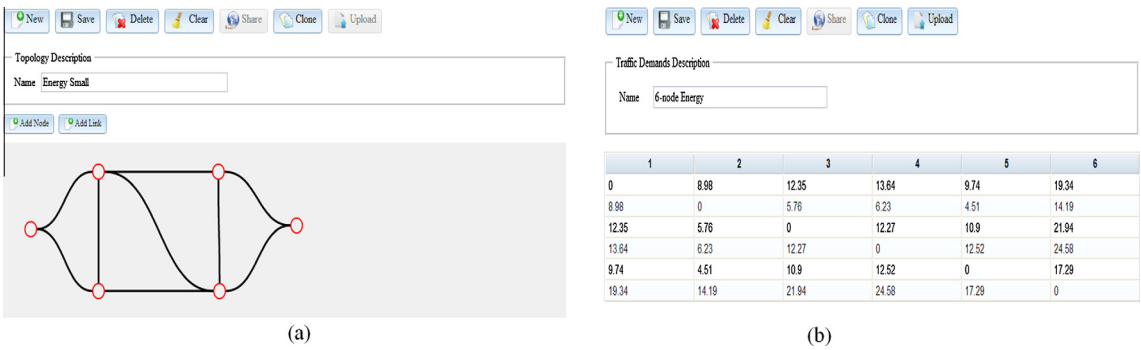

Fig. 9. (a) Creation of network topology and (b) definition of traffic requests.

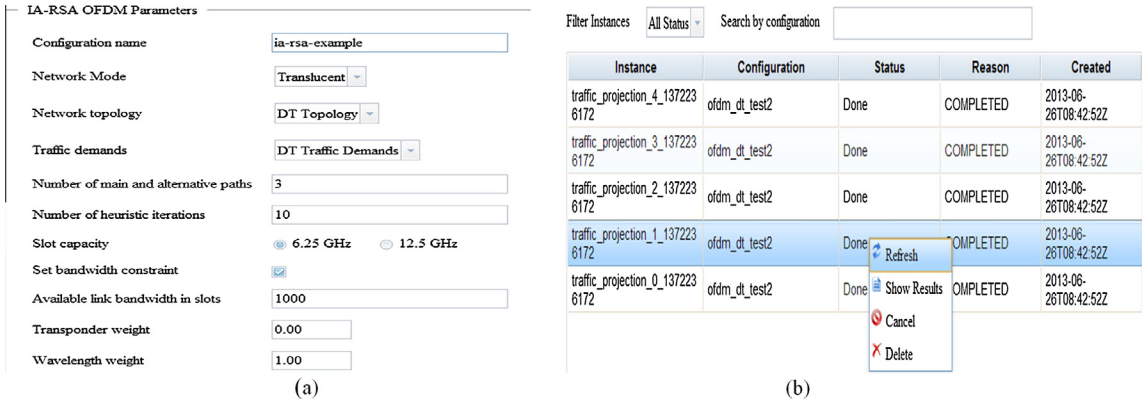

Fig. 10. (a) Creation of configuration and (b) information for all instances.

6.3. Social characteristics

Since one of Mantis' primary purposes is to create a common benchmarking environment, the tool provides a private and a public workspace. The private workspace includes the user's network topologies, traffic matrices or demands, configurations, results and charts. Network topologies, traffic demands, configurations and charts can be shared with other users. The shared items are available in the public workspace; the other users have only read access to them, but they can create their own copies.

7. Evaluation–performance

Most of the interesting network planning problems in optical networks are unfortunately NP-complete, that is, the number of operations required to solve them is non-polynomial to the size of the input problem. Since solving realistic problem instances requires a large number of variables and constraints, solving such problem becomes computationally intractable. Heuristic algorithms are thus typically employed that have suboptimal performance but acceptable running time. Even with heuristics there are certain scalability issues, that are not present in current planning tools but are slowly emerging: the networks will increase in size (convergence of metro and core optical networks), resource allocation will become more complex (cross-layer design of IP and optical layers), and the timescales of executing the algorithm will decrease (more dynamic optical network so as to follow the IP layer). Thus, running many what-if scenarios with big inputs, as typically required by such tool when planning networks, or responding quickly to more stringent time requirements as will be required in operating a dynamic optical network, would require a more scalable tool than current desktop solutions. Mantis utilizes the power of the cloud to provide scalability with respect to the number of concurrent scenarios evaluated by distributing the execution of the various experiments on multiple cloud resources in parallel in order to get reasonable solution times.

Mantis design and implementation was thoroughly validated by using it in our own algorithmic research as a tool for developing new innovative algorithms both for fixed-grid and flex-grid optical networks. In what follows, we present a number of tests we performed for evaluating Mantis robustness and performance but also the benefits of its dual, cloud and desktop, operation:

- A basic test to evaluate the parallelization benefits by adding new resources to cope with the current demand.
- A test to evaluate the fair access to the available resources in competition.
- A scalability test to evaluate the ability of the tool to scale based on the user demands executing fast and efficiently the requested scenarios.
- A stress test to evaluate the operation of the tool under heavy load with limited resources available to serve the user requests.

In the tests performed Mantis utilizes public cloud resources from Amazon Elastic Compute Cloud (Amazon EC2) and ~okeanos, the GRNET's (Greek National Research and Education Network) cloud service. Amazon's cloud resources are practically unlimited with various capacity offerings but come with a cost, while ~okeanos resources are free for the Greek academic and research community, but are limited in number and in capacity.

In the following paragraphs we provide the detailed results of evaluation.

7.1. Basic evaluation

Initially, we evaluated the average execution time of a variable number of standalone experiments (instances) when Mantis operates in the desktop and in the cloud mode. In particular, we considered the planning of an optical network with 14 nodes and using the IA-RWA-MLR algorithm. Different instances had different aggregated traffic load, ranging from 400 Gbps to 2 Tbps. Similar, instances where also created for the other tests (presented in the following subsections).

We parallelized the execution of instances by allocating one instance per CPU core. The desktop application server ran Ubuntu 12.04 and had 4 GB RAM and 4 CPU cores, affording to run 4 instances in parallel, while the remaining instances are queued when waiting for their execution. The cloud application engine scales dynamically, adding new resources to the pool when running out of cores; each

virtual resource utilized had the same characteristics as the desktop one. For the cloud deployment we considered two scenarios; in the first (Scenario-1) the cloud application engine could add up to two virtual resources (8 cores in total), while in the second (Scenario-2) it could add up to three virtual resources (12 cores in total).

Fig. 11 illustrates that the cloud deployment offers high degree of parallelization, decreasing considerably the time required to run a set of experiments (instances). We observe that for multiples of 4 instances the average execution time increases linearly for both the desktop and the cloud, with the cloud case being relatively static when the number of instances is smaller than the number of available virtual cores (8 and 12 instances for the Cloud Scenario 1 and 2, respectively). The rate at which the execution time increases with the number of instances is much smaller for the two Cloud Scenarios than it is for desktop execution, the Cloud Scenario 2 being the best among all tested since it employs the more cores. In any case, we should note that the benefits of using cloud resources come at a price, especially when we utilize resources from public cloud infrastructures. Hence, the tool's capability to utilize public and private cloud platforms can be very useful to maintain a good balance between execution times and expenses for hiring the extra computation resources needed.

7.2. Evaluation of fair access to available resources

We also evaluated the fairness of Mantis's dispatcher, in cloud operation mode, when serving instances of different users and the available resources are limited. The evaluation scenario is as follows: we assume that initially the resources are filled executing other instances and there are 5 users in total, from which the first user submits a burst of instances while the rest start their submissions after the first user. For simplicity, all instances have the same workload. The cloud application engine is configured to use only three virtual resources, with each one having 4 cores, without the capability to add extra resources. Hence, the system is able to execute in parallel up to 12 instances.

We evaluated the instances' average execution time and the average queue time in two dispatcher policies:

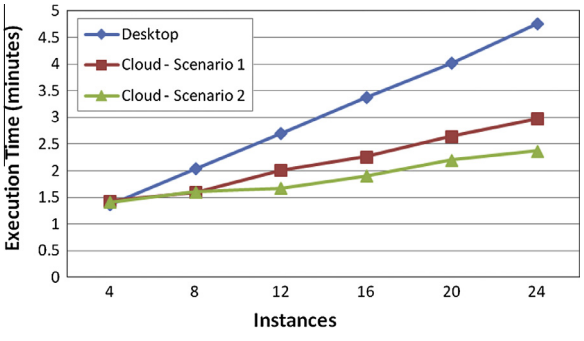

Fig. 11. Mantis instances average execution time in desktop and cloud deployments for various number of instances and different available cloud resources.

- First Come First Serve (FCFS): Mantis dispatcher serves the instances based on their arrival order. In the case where the available resources are limited and some user submits a burst of instances then the instances submitted later by other users face significant queuing times.
- Fair: Mantis dispatcher inspects the queued jobs and handles them in a round-robin manner on a per user basis, so as to achieve more fair allocation of the limited resources.

Fig. 12 shows the average queue and execution time for the instances submitted by each user. As expected, when the fairness policy is used the queue and execution time of different users' instances are more balanced.

7.3. Scalability test

Next, we evaluated the ability of the tool to scale according to the user demands. In this test, the cloud application engine was configured to utilize only the public cloud platforms (Amazon and ~okeanos) and we did not use our private cloud, based on OpenStack. Initially, the cloud application engine serves the submitted instances using ~okeanos cloud service, utilizing three resources at maximum (each with 4 CPU cores and 8 GB RAM, thus serving up to 12 instances in parallel). When more instances concurrently request service the cloud application engine starts utilizing resources from Amazon EC2.

Amazon EC2 provides a number of different types of resources [41] that combine various CPU, memory, storage and networking characteristics, while each resource type includes one or more sizes. Since all algorithms in Mantis are CPU intensive for this scalability test we used two of the compute-optimized Amazon EC2 resources, namely c1.medium and c1.xlarge with cost $0.165 and $0.66 per hour [42], respectively.

In the dispatcher we used the following policies for allocating additional resources from Amazon EC2 cloud service:

- **Policy-1**: the dispatcher adds a c1.medium Amazon EC2 resource when the total queued instances are more than 20, while the limit of concurrent instances in each additional resource is set to 8 jobs.
- **Policy-2**: the dispatcher adds a c1.medium Amazon EC2 resource when the total queued instances are more than 10, while the limit of concurrent instances in each additional resource is set to 8 jobs.
- **Policy-3**: the dispatcher adds a c1.xlarge Amazon EC2 resource when the total queued instances are more than 10. The limit of concurrent instances that the dispatcher can assign to each new resource is up to 20 jobs.

In Table 1 we present the number of additional Amazon EC2 resources that each dispatcher policy used to scale so as to serve the submitted loads. As expected, the number of the resources increases as the number of the submitted instances increases. The number of resources also depends on the selected limit for the maximum number instances in queue after which the dispatcher adds a new resource.

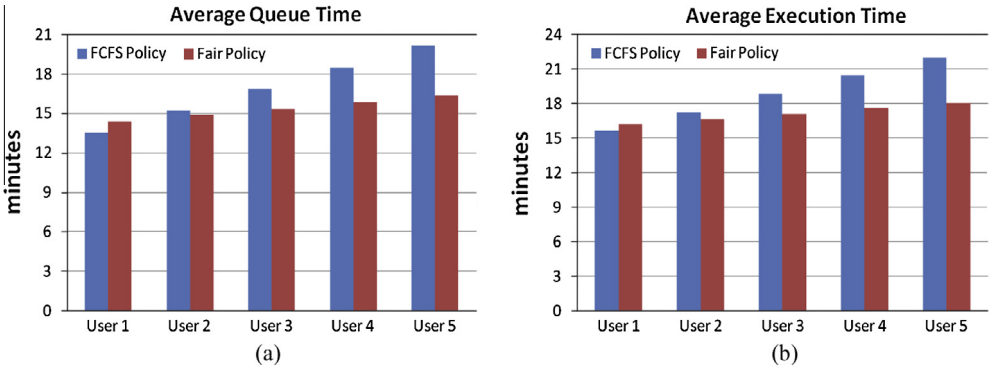

Fig. 12. (a) Average queuing and (b) execution time of instances per user in the scenario where the first user submits a burst of instances, followed by the other users.

Table 1
Number of additional Amazon EC2 instances that have been used from each dispatcher policy.

Submitted instances	Policy-1 (c1.medium)	Policy-2 (c1.medium)	Policy-3 (c1.xlarge)
100	1	3	2
200	2	5	3
300	3	6	5
400	5	8	7
500	7	11	10
600	9	14	13

Thus, both the second and third policies (both afford 10 jobs in their queues) utilize more additional Amazon EC2 resources compared to the first policy (affords 20 jobs in its queue).

Fig. 13a shows the average execution time for the submitted instances for each dispatcher policy. It is evident that in all policies the tool scales well with the demand, serving the submitted instances in reasonable time. As expected, the average execution time increases with the number of submitted instances, due to the increased queuing time. Nevertheless, by adding new resources Mantis scales efficiently, by dynamically increasing the number of instances that are executed simultaneously and thus keeping low the instances queuing time.

Fig. 13b illustrates the maximum number of instances that are executed in parallel for each dispatcher policy. As the dispatcher utilizes more cloud resources the number of concurrently executed instances increases, leading to improved average execution time, since the queued instances are decreased. We observe that, even when utilizing the same Amazon EC2 resource type (Policy-1 and Policy-2), a different value for the maximum number of queued instances affects the performance. Choosing a smaller maximum value results in the utilization of more resources from the dispatcher, which in turn increases the number of concurrently executed instances and thus improves the average total execution time.

In Fig. 13c we present the corresponding cost ($) for the additional Amazon EC2 resources that the dispatcher used in each case. In all policies the cost increases with the number of submitted instances since the dispatcher utilizes more resources to serve the increased load. We observe that the total cost increases almost linearly for all policies, but for Policy-1 it grows with the smallest rate, while Policy-2 comes second best. Note the significant difference in the required cost between Policy-2 and Policy-3 even though in both cases the dispatcher utilized almost the same number of resources (Table 1). This is because the cost for the Amazon EC2 resources used in Policy-3 (c1.xlarge) is four times higher than the cost for the resources used in Policy-2 (c1.medium).

Hence, the benefits of using public cloud resources come with a cost that could be high if we do not take it into consideration. For example, the daily usage cost only for an Amazon EC2 instance c1.medium and c1.xlarge is $3.96 and $15.84, respectively. Furthermore, the total cost for executing all cases with Policy-1, Policy-2 and Policy-3 was $4.445, $7.775 and $26.4, respectively, while the cost for all Amazon EC2 resources that we have used during the all evaluation tests was $105. Thus, it is crucial to choose the right dispatcher policy and also utilize Mantis' feature that control the budget per month (as shortly discussed in Section 4).

The scalability evaluation performed above exhibits also Mantis ability to utilize transparently virtual resources from more than one cloud (public) providers (Amazon EC2 and ~okeanos). Apart from two public clouds, a hybrid private-public cloud infrastructure can also be used: basic calculations can be done in a private cloud based on OpenStack, utilizing the local resources, while when required to serve load peaks and time-dependent calculations then acquire additional resources from a public cloud such as Amazon EC2. With the proliferation of cloud providers of various properties and costs, this feature of Mantis is very important.

7.4. Stress test

Finally, to evaluate the robustness of Mantis implementation we tested it beyond the limits of its normal operation. For this reason we limited to 4 the number of the Amazon EC2 additional resources (c1.medium) that the cloud application engine could use. The dispatcher was

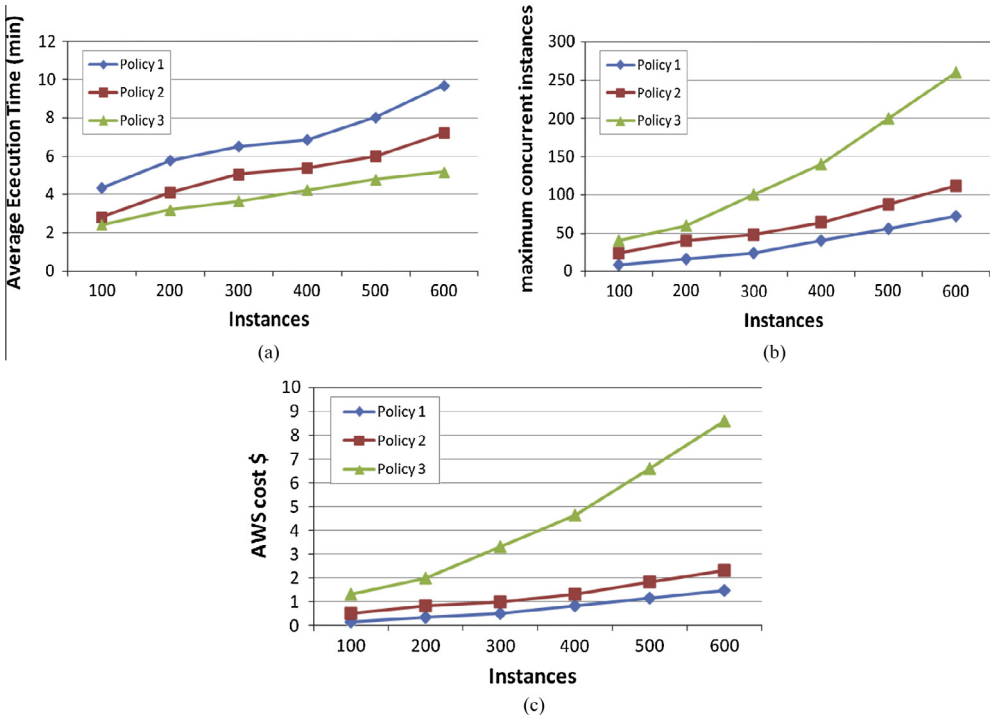

Fig. 13. (a) Average execution time for submitted instances for each dispatcher policy, (b) number of concurrent executed instances and (c) corresponding cost for the additional Amazon EC2 resources.

configured to add a new Amazon EC2 resource when the number of the queued instances exceeded 20. Furthermore, we created heavy load conditions by assuming multiple users who submit concurrently bursts of instances. Each burst consisted of 100 instances and its submission duration was 50 s (2 instances/sec). We evaluate the operation of the Mantis using 1, 3, 5, 7, 9 and 11 users that submit concurrently their bursts resulting to loads of 100, 300, 500, 700, 900, 1100 instances, respectively.

Fig. 14 shows the average queuing and execution time as a function of the total number of submitted instances. Although few resources were available, Mantis managed to execute successfully all the submitted instances for all loads examined. As expected, both metrics increase as the total number of instances increases, while the queue time dominates the total execution time, since the instances are submitted in bursts during a short time interval and there are limited resources.

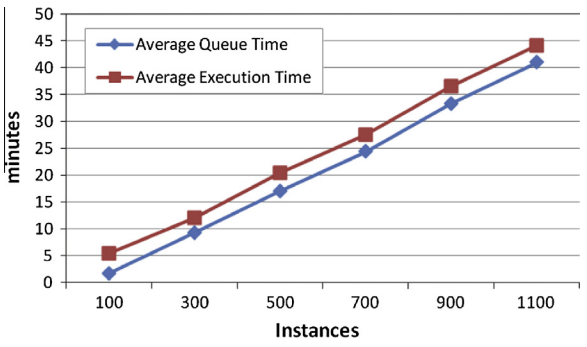

Fig. 14. Average queueing and execution time when Mantis utilizes limited resources while multiple users submit concurrently bursts of instances.

8. Future work

A number of extensions are planned in all layer of Mantis in the near future. These include adding new functionality both in the web-based user interface and in the exposed RESTful API, and developing a command-line interface (CLI) for users to interact with the tool. Also, we will add more optical devices with characteristics that correspond to actual network devices along with enriched energy consumption models of the components used in optical transport networks. In addition, more sophisticated dispatcher policies will be implemented that will use information from the database to estimate the instances execution times and improve load balancing and fairness. New algorithms for flexible optical networks will also be integrated in the execution engine library, especially for spectrum allocation and defragmentation, restoration, and energy efficiency. Also, we will further automate and simplify the process of adding new algorithms developed by users to the tool. Finally, we plan to provide appropriate interfaces to integrate Mantis's online algorithms with optical network management tools so that Mantis will provide the functionality of the path computation element (PCE).

9. Conclusions

In this paper we presented Mantis, a network planning and operation tool for next generation optical networks. Mantis is, to the best of our knowledge, the first complete tools implemented for planning flexible optical networks and includes novel and efficient algorithms. Mantis can be deployed either as cloud service (SaaS) or as desktop application, with the former being the primary implementation of interest so as to be available over the internet to the users. Users (researchers, operators, vendors) can use Mantis to perform studies as desired but also implement their own algorithms and compare them against the ones already incorporated in the tool. Mantis can create a common benchmarking environment with social characteristics where researchers share topologies, traffic matrices and CAPEX/OPEX parameters, and evaluate their algorithms under common conditions. In this way, Mantis could also evolve as an online collaboration platform for optical network researches, improving the comparability, quality and reliability of the results presented in various research articles and projects.

The performance evaluation tests show that by utilizing the power of the cloud, Mantis scales well with respect to the number of scenarios evaluated by distributing the experiments execution on multiple cloud resources in parallel. Furthermore, the stress test verified the robustness of Mantis implementation under heavy load conditions, where even with limited resources the tool was able to execute successfully all the submitted experiments. Finally, the scalability evaluation manifested Mantis ability to utilize transparently virtual resources from multiple public and private cloud infrastructures.

Acknowledgements

Part of this work is implemented within the framework of the Action Supporting Postdoctoral Researchers of the Operational Program Education and Lifelong Learning (Action's Beneficiary: General Secretariat for Research and Technology), and is co-financed by the European Social Fund (ESF) and the Greek State. Part of this work has been supported by the ICT IDEALIST project (grant agreement number 317999).

References

[1] Cisco Visual Networking Index – Forecast and Methodology, 2012–2017.
[2] T. Stern, G. Ellinas, K. Bala, Multiwavelength Optical Networks: Architectures, Design and Control, second ed., Cambridge University Press, 2008.
[3] O. Gerstel, M. Jinno, A. Lord, S.J. Ben Yoo, Elastic optical networking: a new dawn for the optical layer?, IEEE Commun Mag. 50 (2) (2012).
[4] Evolution to flexible grid WDM, <http://www.lightwaveonline.com/articles/print/volume-30/issue-6/features/evolution-to-flexible-grid-wdm.html> (last seen June 2014).
[5] J. Sole-Pareta, S. Subramaniam, D. Careglio, S. Spadaro, Cross-layer approaches for planning and operating impairment-aware optical networks, Proc. IEEE 100 (5) (2012) 1118–1129.
[6] G. Zhang, M. De Leenheer, A. Morea, B. Mukherjee, A survey on OFDM-based elastic core optical networking, IEEE Commun. Surv. Tutorials 15 (1) (2013) 65–87.
[7] E. Vavrarigos, K. Christodoulopoulos, Algorithmic aspects in planning fixed and flexible optical networks with emphasis on linear optimization and heuristic techniques, IEEE/OSA J Lightwave Technol 30 (4) (2014) 681–693.
[8] A. Kretsis, E.A. Varvarigos, Mantis: optical network planning and operation tool, in: Workshop on Network Planning and Design Tools of ICTON, 2013.
[9] A. Kretsis, K. Christodoulopoulos, P. Kokkinos, E. Varvarigos, Planning and operating flexible optical networks: algorithmic issues and tools, IEEE Commun. Mag. 52 (1) (2014) 61–69 (special issue on Advances in Network Planning).
[10] Cariden, <http://www.cariden.com/products/mate-product-family> (last seen January 2014).
[11] Opnet NP, <http://www.opnet.com/solutions/network_management/spguru_network_planner> (last seen January 2014).
[12] Openet TP, <http://www.opnet.com/solutions/network_management/transport_planner.html> (last seen January 2014).
[13] Wandl mplsview, <http://www.wandl.com/html/mplsview/ipmplsview.php> (last seen January 2014).
[14] Wandl npat, <http://www.wandl.com/html/npat/npat.php> (last seen January 2014).
[15] Aria, <http://www.aria-networks.com/solutions/ip-mpls-te-operational-planning> (last seen January 2014).
[16] VPI, <http://www.vpisystems.com/solutions/multi-layer-transport-planning—optimization> (last seen January 2014).
[17] Nokia Siemens, <http://www.nokiasiemensnetworks.com/portfolio/products/transport-solutions/network-management-planning> (last seen January 2014).
[18] Cisco, <http://www.cisco.com/en/US/prod/collateral/optical/ps5726/ps11348/data_sheet_c78-658849.html> (last seen January 2014).
[19] RFC 4655, A Path Computation Element (PCE)-Based Architecture, 2006.
[20] Infinera, <http://www.infinera.com/products/mgmtsuite.html> (last seen January 2014).
[21] K. Christodoulopoulos, P. Soumplis, E. Varvarigos, Planning flexgrid optical networks under physical layer constraints, IEEE/OSA J. Opt. Commun. Networking 5 (11) (Nov. 2013) 1296–1311.
[22] R. Tucker, R. Parthiban, J. Baliga, K. Hinton, R. Ayre, W. Sorin, Evolution of WDM optical IP networks: a cost and energy perspective, IEEE/OSA J. Lightwave Technol. 27 (3) (2009) 243–252.
[23] P. Kokkinos, K. Christodoulopoulos, K. Manousakis, E. Varvarigos, Multi-Parametric Online RWA based on Impairment Generating Sources, in: IEEE Global Conference on Communications (GLOBECOM), Honolulu, USA, December 2009.
[24] K. Christodoulopoulos, K. Manousakis, E. Varvarigos, Offline routing and wavelength assignment in transparent WDM networks, IEEE/ACM Trans. Networking 18 (5) (Oct. 2010) 1557–1570.
[25] K. Manousakis, A. Angeletou, E. Varvarigos, Energy efficient RWA strategies for WDM optical networks, J. Opt. Commun. Networking 5 (4) (2013) 338–348.
[26] K. Christodoulopoulos, K. Manousakis, E. Varvarigos, M. Angelou, Considering physical layer impairments in offline RWA, IEEE Network Mag. 23 (3) (2009) 26–33.
[27] Amazon Web Services, <http://aws.amazon.com> (last seen January 2014).
[28] OKEANOS – GRNET cloud infrastructure, <https://okeanos.grnet.gr> (last seen January 2014).
[29] OpenStack, Open source software for building private and public clouds, <http://www.openstack.org> (last seen January 2014).
[30] Mantis – Online Optical Network Planning and Operation Tool, <http://www.mantis-tool.net> (last seen January 2014).
[31] Ruby on Rails open source web framework, <http://rubyonrails.org> (last seen January 2014).
[32] The Dojo toolkit, <http://dojotoolkit.org> (last seen January 2014).
[33] PostgreSQL open source object-relational database system, <http://www.postgresql.org> (last seen January 2014).
[34] Amazon Elastic Compute Cloud (Amazon EC2), <http://aws.amazon.com/ec2> (last seen January 2014).
[35] Amazon Simple Queue Service (Amazon SQS), <http://aws.amazon.com/sqs> (last seen January 2014).
[36] Amazon Simple Storage Service (Amazon S3), <http://aws.amazon.com/s3> (last seen January 2014).
[37] boto – A Python interface to Amazon Web Services, <http://docs.pythonboto.org> (last seen January 2014).
[38] Synnefo cloud software, <http://www.synnefo.org/docs/synnefo/latest/index.html> (last seen January 2014).
[39] Qt framework, <http://qt-project.org> (last seen January 2014).
[40] Cython C-Extensions for Python, <http://www.cython.org> (last seen January 2014).

[41] Amazon EC2 instance types, <http://aws.amazon.com/ec2/instance-types> (last seen January 2014).
[42] Amazon EC2 pricing, <http://aws.amazon.com/ec2/pricing> (last seen January 2014).

Aristotelis Kretsis graduated in 2006 from the University of Patras and was awarded his Diploma of Computer Engineer and Informatics. In 2009 he received his M.Sc. degree in computer science and engineering and in 2014 his Ph.D from the same department. His research interests are in the areas of grid computing, distributed computing, and software management tools.

Panagiotis Kokkinos is a research fellow at NTUA, Greece. He received his Ph.D. in 2010 from the Computer Engineering and Informatics Department of University of Patras, Greece, in the field of optical grid networks. He also holds an M.Sc. degree (2006) and a Diploma (2003) from the same department. His current research activities are in the areas of distributed computing and networks.

Kostas Christodoulopoulos is a research fellow at the Department of Computer Engineering and Informatics (CEID), University of Patras. He received his Diploma in Electrical and Computer Engineering from the National Technical University of Athens (NTUA), his M.Sc. degree from Imperial College London, and his Ph.D. degree from CEID. He worked as an adjunct assistant professor at CEID, as a research fellow at CTVR, Trinity College Dublin, and as a contractor for IBM Research, Ireland. His research interests are in the areas of algorithms and protocols for optical networks.

Theodora Varvarigou received the B. Tech degree from the National Technical University of Athens, Athens, Greece in 1988, the MS degrees in Electrical Engineering (1989) and in Computer Science (1991) from Stanford University, Stanford, California in 1989 and the Ph.D. degree from Stanford University as well in 1991. She is a Professor at the National Technical University of Athens. Prof. Varvarigou has great experience in the area of semantic web technologies, scheduling over distributed platforms, embedded systems and grid computing.

Emmanouel (Manos) Varvarigos received a Diploma in Electrical and Computer Engineering from NTUA in 1988, and M.S. and Ph.D. degrees in electrical engineering and computer science from MIT in 1990 and 1992. He is a professor in the Department of Computer Engineering and Informatics at the University of Patras, and the scientific director of the Network Technologies Sector of the Computer Technology Institute and Press – Diophantus.