Adaptive Resource Management and Control in Software Defined Networks

Daphne Tuncer, Marinos Charalambides, Stuart Clayman, and George Pavlou

***Abstract*—The heterogeneous nature of the applications, technologies and equipment that today's networks have to support has made the management of such infrastructures a complex task. The Software-Defined Networking (SDN) paradigm has emerged as a promising solution to reduce this complexity through the creation of a unified control plane independent of specific vendor equipment. However, designing a SDN-based solution for network resource management raises several challenges as it should exhibit flexibility, scalability and adaptability. In this paper, we present a new SDN-based management and control framework for fixed backbone networks, which provides support for both static and dynamic resource management applications. The framework consists of three layers which interact with each other through a set of interfaces. We develop a placement algorithm to determine the allocation of managers and controllers in the proposed distributed management and control layer. We then show how this layer can satisfy the requirements of two specific applications for adaptive load-balancing and energy management purposes.**

***Index Terms*—Software defined networking, adaptive resource management, decentralized network configuration.**

I. Introduction

THE evolution of information and communication technology (ICT) over the past thirty years has heavily influenced the life of the modern consumer. The crucial role played by ICT today has catered for a persistent demand in terms of new services and applications with strict requirements in terms of availability, service quality, dependability, resilience and protection. This has resulted in increasingly complex networks and software systems that need to support heterogeneous applications, technologies and multi-vendor equipment, making the management of network infrastructures a key challenge.

The vision of Software-Defined Networking (SDN) as a key enabler for simplifying management processes has led to keen interest from both the industry and the research community, who have been investing significant efforts in the development of SDN-based solutions. SDN enables the control of networks via a unified plane which is agnostic to vendor equipment and operates on an abstract view of the resources. Among its advantages, flexibility and programmability are usually highlighted, in addition to simplification of management tasks and application deployment through a centralized network view [1]–[3].

Centralized management and control solutions have, however, limitations. In addition to resilience, scalability is an important issue, especially when dealing with operations that require dynamic reconfiguration of network resources. Centralized approaches are generally well-suited for implementing the logic of applications for which the time between each execution is significantly greater than the time to collect, compute and disseminate results. As a consequence, adaptive management operations with short timescales call for both distributed management and control approaches, which are essential enablers of online resource reconfigurations.

In this paper, we present a novel SDN-based management and control framework for fixed backbone networks. The proposed framework, based on SDN principles, follows a layered architecture where the communication between layers is achieved through a set of interfaces. Although different approaches have been proposed in the literature, these have either mainly focused on the control plane (e.g, [4], [5]) or considered centralized management solutions (e.g., [6], [7]). In contrast, the framework presented in this paper relies on a distributed management and control layer, which consists of a set of local managers (LMs) and controllers (LCs) forming separate management and control planes. The modular structure of this layer is a salient feature of our approach. This not only simplifies the integration of management applications, but also offers significant deployment benefits, allowing control and management functionality to evolve independently. For example, management and control components from different vendors can be used together, but also the functionality of management applications can be updated without disrupting active network services. The degree of distribution in each plane (number of elements) depends both on the physical infrastructure as well as the type of management applications to consider. The exchange of information between distributed elements in each plane is supported by the *management substrate* developed in our previous work [8], [9].

We investigate how the proposed framework can be used to support adaptive resource management operations which involves short timescale reconfiguration of network resources, and we discuss the main research issues/challenges associated with the deployment of such a distributed solution. In particular, we show how the requirements of the adaptive load-balancing and energy management applications proposed in our previous work [10]–[12] can be satisfied by the functionality and interfaces of the framework. In addition, we develop a placement algorithm to determine the allocation of LMs and LCs (both

Manuscript received November 15, 2014; revised February 2, 2015; accepted February 5, 2015. Date of publication February 11, 2015; date of current version March 17, 2015. This research was funded by the EPSRC KCN project (EP/L026120/1) and by the Flamingo Network of Excellence project (318488) of the EU Seventh Framework Programme. The associate editor coordinating the review of this paper and approving it for publication was F. De Turck.

The authors are with the Department of Electronic and Electrical Engineering, University College London, London WC1E 7JE, U.K.

Digital Object Identifier 10.1109/TNSM.2015.2402752

TABLE I
MAIN ACRONYMS

ARMA	Adaptive Resource Management Application
EM	Energy Management
FW	Forwarding
LB	Load-Balancing
LC	Local Controller
LCO	Local Controller Orchestrator
LM	Local Manager
LMO	Local Manager Orchestrator
MTR	Multi-Topology Routing
RLC	Router Line Card
RMA	Routing Management Application
TE	Traffic Engineering

in number and mapping to network equipment) according to topological characteristics of the physical infrastructure. Based on real network topologies, we show how the parameters of the algorithm can be tuned to control the allocation. We also evaluate the performance of the load-balancing and energy management applications in terms of resource utilization based on real traffic traces and compare their performance to different schemes. The results demonstrate that a significant reduction in terms of link utilization and energy consumption can be achieved in a scalable manner.

The remainder of this paper is organized as follows. Section II provides background information. Section III describes the main components and interfaces of the proposed framework and highlights the operations performed by each component. Section IV presents the placement algorithm. An overview of the adaptive resource management applications is provided in Section V. Section VI describes in detail how the requirements of these applications are supported by the proposed framework. The results of the evaluation are presented in Section VII and Section VIII discusses related work. Finally, conclusions and future directions are provided in Section IX.

II. DEFINITIONS AND BACKGROUND

In this section, we first define some of the basic terms and notations used in this paper and give background information on SDN. We also present the network management substrate, which forms the communication basis of the proposed architecture. In the last sub-section, we provide information about Multi-Topology Routing which serves as the routing mechanism for achieving path diversity in the context of the two management applications considered in this work. For clarification purposes, the main acronyms used in this paper are summarized in Table I.

A. Definitions

We consider network topologies represented by the sets of network links $\mathcal{L}$ and nodes $\mathcal{N}$. We refer to network edge nodes as the set of nodes generating and absorbing traffic and we represent this set by $\mathcal{N}_{\mathrm{E}}$. We refer to all other nodes as core nodes. For any pair of edge nodes $(i,j) \in \mathcal{N}_{\mathrm{E}}$, sd_{ij} represents the source-destination pair of source node i and destination node j. Each sd_{ij} is associated with a volume of traffic $v(sd_{\mathrm{ij}})$ that represents the traffic demand between source node i and destination node j. We define the traffic flow $F(sd_{\mathrm{ij}})$ as the 2-tuple $(sd_{\mathrm{ij}}, v(sd_{\mathrm{ij}}))$ and the set of traffic flows ϕ_{i} locally originating at edge node $i \in \mathcal{N}_{\mathrm{E}}$ as follows:

$$\forall\, i \in \mathcal{N}_{\mathrm{E}}, \quad \phi_{\mathrm{i}} = \{F(sd_{\mathrm{ij}}), \quad j \in \mathcal{N}_{\mathrm{E}}\} \tag{1}$$

B. Software-Defined Networking

The main principle of SDN lies in the decoupling of network control from forwarding hardware [13]. In the SDN architecture, physical network devices are represented as basic forwarding elements (usually referred to as switches), forming a data plane, and are supervised by a network-wide control platform consisting of a set of software components (the controllers) [5]. The control platform can be seen as a logically centralized control plane which operates on a global network view and which implements a range of control functions. The controllers interact with the switches via a standardized interface which is used to collect network state information and distribute control commands to be enforced in the network. The existence of a standard interface enables the separation of the control and forwarding logic and, as such, supports the independent evolution of both planes.

Although there is no formal requirement for the choice of the interface to use, OpenFlow [14] has progressively imposed itself as the *de facto* standard given the massive support from both academia and industry [3]. As stated in its specifications, it provides an open protocol to define the basic primitives for programing the forwarding plane of network devices [13]. In the OpenFlow model, the network traffic is identified as a set of flows which are defined according to a set of pre-defined match rules instantiated by the controller in the flow tables of the switches. Each flow is associated with a set of instructions used to control how traffic should be routed and treated in the network.

Recently, the Open Networking Foundation (ONF) has presented an architecture for SDN in a technical report [15]. The proposed architecture consists of the following three planes: (i) a data plane, comprising the set of network elements, (ii) a controller plane, with a set of SDN controllers which have exclusive control over a set of network resources, and (iii) an application plane, implementing a set of network/management applications which are executed through the control plane. The communication between the planes is realized in a hierarchical manner through a set of interfaces.

C. Network Management Substrate

In our previous work [8], [9], we developed an in-network management approach for fixed backbone networks, in which an intelligent substrate is used to enable the dynamic reconfiguration of network resources. Compared to traditional management solutions, where reconfigurations are decided offline by a centralized management system that has a global view of the network, reconfiguration decisions are directly taken in a decentralized and adaptive fashion by the decision-making entities distributed across network edge nodes, based on periodic feedback from the network. The decision-making entities are organized into a *management substrate* (MS), which is a

logical structure used to facilitate the exchange of information. In particular, it is used for coordination purposes since it provides a means through which decision-making points can communicate.

Any node in the substrate can directly communicate only with its neighbors, which are defined by the topological structure used. The choice of this structure can be driven by different parameters related to the physical network, such as its topology, the number of edge nodes, but also by the constraints of the coordination mechanism between the nodes and the associated communication protocol. The overhead incurred by the communication protocol in terms of delay and number of messages exchanged, for example, is a key factor that can influence the choice of the structure [9].

D. Multi-Topology Routing

To achieve their objectives, most resource management approaches employ routing protocols that can support path diversity. Multi-Topology Routing (MTR) [16], [17] is a standardized extension to the common Interior Gateway routing Protocols, i.e., OSPF and IS-IS, which can provide a set of multiple routes between any source-destination (S-D) pair in the network by enabling the virtualization of a single physical network topology into several independent virtual IP planes.

To determine to which topology packets need to be routed, these are marked at network ingresses with the Multi-Topology Identifier (MT-ID) of the routing topology to which the corresponding traffic flows have been assigned. A separate routing table needs to be implemented for each topology (i.e., routing scheme) in each router, so that upon receiving a traffic flow, the router analyzes the MT-ID marked in the packets and forwards the packets to the next-hop according to the relevant routing table [18]. The configuration of the different virtual planes is part of an offline process which computes a set of desired IP virtual topologies given the physical network topology.

Splitting ratios, enforced at network ingresses, can subsequently control the portion of an incoming traffic flow to be routed over each of the virtual planes.

III. Management and Control Framework

Resource management in fixed networks is usually performed by external offline centralized systems, which optimize network performance over long timescales. Typically, the central manager operates on a global view of the network, which facilitates the implementation of management applications. However, while centralized/offline solutions are adequate for network operations that do not require frequent reconfigurations (e.g., computation of MTR planes), they are not appropriate for applications that adapt to traffic and network dynamics (e.g., online traffic engineering). In addition to the single point of failure, these approaches have limitations especially in terms of scalability (i.e., communication overhead between the central manager and devices at runtime) and lag in the central manager reactions, which may result in sub-optimal performance. To overcome these limitations, dynamic management applications and control should rely on a distributed framework. In this section, we present a hierarchical resource management and control framework for fixed backbone infrastructures in the context of SDN environments.

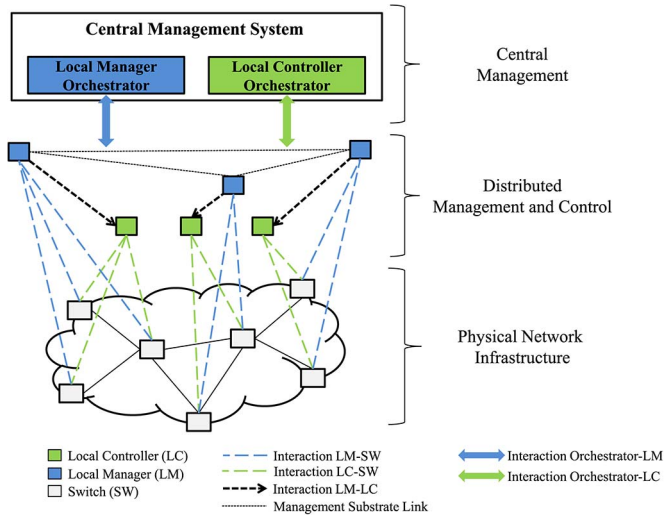

Fig. 1. Proposed framework.

A. Architecture

In the proposed framework, the network environment is conceptually divided into three layers as shown in Fig. 1. The bottom layer represents the physical infrastructure, which consists of network switches[1] and links, and can be defined as the data or forwarding plane. The second layer consists of a set of local controllers (LCs) and managers (LMs), forming the distributed control and management planes, respectively. Finally, the third layer represents the centralized management system.

A key characteristic of the proposed framework is its modular nature, which enables the separation between the management application logic (represented by LMs) and the control logic (represented by LCs). As a result, this allows the two to evolve independently, offering increased design choices and flexibility for the system vendors, as well as simplified integration of network applications, while maintaining interoperability.

More specifically, the LCs and LMs are software components which are in charge of controlling and managing the network resources (i.e., switches and links), respectively. Each LC is responsible for a set of network switches, which define its scope of control, so that a network switch is controlled by one LC only. In addition, each LC is logically associated with one or more LMs. The LMs implement the logic of management applications (e.g., traffic engineering) and are responsible for making decisions regarding the settings of network parameters—for example to compute new configurations that optimize resource utilization—to be applied in the switches under their responsibility. To take management decisions, the LMs communicate through the *management substrate*, as described in Section II-C and shown in Fig. 1. The substrate, which was proposed in our previous work [8], is implemented as an integral part of the framework and is used by LMs to exchange information about the network state and the configurations to apply. Configuration

[1]In this paper, we assume that each network node is represented by an OpenFlow switch.

decisions taken by LMs are provided to peering LCs, which define and plan the sequence of actions to be enforced for updating the network parameters. These actions are then mapped to OpenFlow instructions sent-to and executed-by the switches. It is not the intention of this paper to propose a specific planning mechanism and mapping functions. These are implementation choices that depend on the type of inputs from the management applications. The separation of concerns between LMs and LCs provides significant deployment benefits since changes can be applied to LMs in an operational environment independently of the LCs and vice versa. In particular, replacing or updating the management logic can be achieved without stopping the network as LCs can rely on existing rules.

The number of LMs and LCs to deploy, as well as the association between the two, can depend on different factors such as the size and topology of the physical network, or the type of management applications to support. In this paper, we adopt a configuration similar to the one depicted in Fig. 1. We consider an equal number of LCs and LMs and a one-to-one mapping between them. In addition, LCs and LMs interact with the same set of switches, i.e., there is a perfect overlap between their zones of responsibility. As shown in Section VI, such a model is well suited to the resource management application scenarios investigated in this paper.

The centralized management system consists of two components, namely, the Local Controller Orchestrator (LCO), which supervises all LCs, and the Local Manager Orchestrator (LMO), which supervises all LMs. These are responsible for longer term operations, for example those that pertain to the life cycle of LMs and LCs. In particular, they are used to determine the number of LMs and LCs to deploy, their location, as well as their zone of responsibility.

It should be noted that the proposed architecture is compatible with the generic ONF SDN model [15]. However, while the report does not elaborate on the specifics of each layer, we go a step further in this paper by investigating the issues that arise from the requirements and realization of the functionality of such an architecture. In particular, we investigate how the distributed management and control planes can support dynamic operations.

B. System Design and Interfaces

The three layers of the proposed architecture are realized with a set of functional components and interfaces that facilitate the interaction/communication between the various components. These are depicted in Fig. 2 and elaborated below.

1) Functional Components: From an architectural viewpoint, the system can be decomposed into four main components.

Central Management System The functionality of the central management system pertains to long term management operations. The LM Substrate Orchestrator module implements methods to compute the structure of the *management substrate*. The Application Orchestrator module is responsible for performing a high-level control of the management applications which are instantiated in the network. In particular, this module decides how the logic of each application should be distributed across the different LMs (i.e., to select the LMs which need to

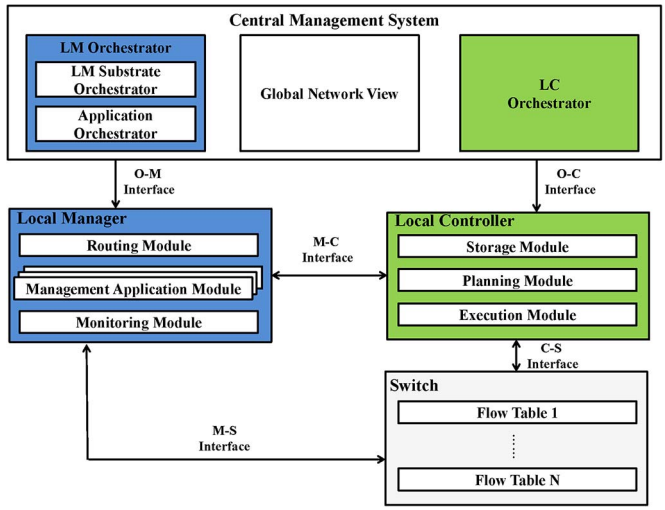

Fig. 2. Overview of system components and interfaces.

be involved in the decision-making process of a given application). The LC Substrate Orchestrator module is concerned with the supervision of LCs. The decisions taken by the central management system rely on information obtained from the Global Network View, which maintains a global knowledge about the environment, such as the physical network topology. It should be noted that, in this work, we do not elaborate on issues associated with gathering such information and generating a global network view. This component is included in Fig. 2 for completeness purposes to illustrate how this view can be used by other components of the framework.

Local Manager From a functional viewpoint, a LM can be represented by three main modules. The Monitoring Module is concerned with functions related to network monitoring, such as data collection, filtering, aggregation etc., and in particular, it enables each LM to create its own local network view. Furthermore, it also allows the LM to maintain consistency with the network view of other LMs. Network information collected and generated by the Monitoring Module can be stored on the local memory of the LM. The logic to perform management operations is realized by Management Application Modules, which maintain information tables and implement algorithms to decide on the configurations to apply. A module is defined for each management application and each LM can implement a different number of applications. The decision of whether a module should be instantiated on a given LM is made by the Application Orchestrator and depends on the application type. Finally, the Routing Module implements basic methods related to the routing functionality (e.g., shortest path computation).

Local Controller An instance of a LC is represented by three main modules. The Storage Module consists of a set of local storage structures, which are used to maintain information received from the LMs regarding the configuration output of management applications. Based on this information, the Planning Module determines the actions to take to (re)configure the switches, for example, according to mapping functions. This also encompasses scheduling methods to decide on how and when actions should be applied. The Execution Module is responsible for translating the planned actions into a set of

configuration commands (e.g., OpenFlow) to be enforced in the switches.

Switch The basic functionality of the network switches is forwarding. In the context of OpenFlow, switches perform packet lookups and forwarding. They are represented by one or more Flow Tables and an OpenFlow channel to an external controller [14]. Each table entry is mapped to a *flow* and is associated with a set of instructions to apply to matching packets. The number of tables to configure, as well as their structure, depends on the nature of the management applications supported by the system. It should be noted that different applications may have different requirements in terms of structures and capabilities to be embedded in the switches (e.g., support for hashing functions).

2) Interfaces: Some previous research initiatives on SDN (e.g., [1], [15]) have used the notion of northbound/southbound to refer to the different interfaces. The relevance of such a terminology presupposes, however, that the controller(s) can be regarded as the focal element(s) of an SDN architecture. In the proposed framework, both LMs and LCs act as the focal points and, as such, we define the name of the various interfaces based on the identity of the interacting components instead.

The interaction between the different components of the proposed architecture is supported by the interfaces shown in Fig. 2. The communication between the orchestrator components (LMO and LCO) and the LMs and LCs is supported by the O-M and O-C interfaces, respectively. The exchange of messages between the LMs and the LCs, for example regarding new configurations computed by the management application modules, is supported by the M-C interface. The M-S interface is defined between the LMs and the switches. It serves monitoring purposes, so that each LM can build its own local view of the network by directly collecting information from the set of switches under its responsibility. Since the network information is primarily needed by the LM, the M-S interface bypasses the LC to avoid additional processing and delay. Finally, the interaction between the LCs and the switches is supported by the C-S interface. Switches can report network events to the LCs, which, in turn, instruct them about configuration updates to apply (e.g., modification of table entries). This interface can be realized by the OpenFlow protocol, however, extensions will be required to enable the configuration of more than one table type.

C. Operations

This subsection provides a detailed description of the main operations performed by each component of the architecture.

Management Operations The main difference between the management operations performed by the central management system and the LMs concerns the timescale at which they are executed. The central management system performs long term operations, which concern the computation of static configurations based on a global view of the network (e.g., connectivity/topology). These rely on a set of algorithms, which are usually invoked at long timescales (e.g., in the order of days/weeks) and are executed in an offline manner. The placement of LMs and LCs, the organization of the *management substrate* and the computation of virtual MTR planes are examples of such operations. To take management decisions, the central manager uses network-wide information maintained by the Global Network View component. This stores information related to the network topology, as well as to the structure of the distributed management and control planes. Any changes in the environment (e.g., node failure) are reported to the central system since these can affect the current settings and should subsequently trigger appropriate reconfigurations.

Short to medium term management operations are performed by the LMs in the distributed management plane. Short term operations are concerned with management decisions taken in the order of seconds/minutes, with failure recovery mechanisms and adaptive splitting ratio reconfiguration algorithms being representative examples. In contrast, medium term operations deal with configurations which need to be updated less often (e.g., every few hours), such as the route computation between two nodes. The decisions can be taken independently by each LM based on local knowledge of the network, which is acquired from switches under their responsibility. However, to avoid configuration inconsistencies, the LMs may also coordinate their decisions through the management substrate. In particular, they can exchange information available locally about network statistics. The characteristics of the coordination process are application-specific.

Control Operations Control operations are performed by the LCs on switches under their scope of control, based on directives received from the LMs. More specifically, the LCs are responsible for configuring the entries of the tables implemented in the switches by deciding *which* entry(ies) should be installed, removed or updated, and also *when* possible changes should be applied. They act as intermediate entities between LMs and switches, capable of translating the decisions of the management modules into commands to be executed to modify table entries in the switches. In addition, LCs can also control which configurations should be applied and when. For instance, a LC may decide to instantiate entries for a subset of the flows to satisfy the memory capacity constraint defined for a table. Configurations that are not directly enforced are stored in the local Storage Module.

At the network level, each incoming packet is matched against entries in successive tables implemented in the switches based on rules. These define whether the packet satisfies some characteristics (i.e., belonging to a given traffic flow). In case of a positive match, the actions defined for the matching entry are added to the action set associated with the packet. In case the packet does not match any entry, it is sent to the relevant LC through the C-S interface to determine how it should be processed. Upon receiving a packet request, the LC defines the set of actions to be applied based on the configurations stored in the Storage Module and instantiates the corresponding new table entries in all the switches under its zone of control.

IV. Local Manager/Controller Placement

A key deployment aspect of the decentralized management and control planes is the distribution of LCs and LMs. It was recently argued by Heller *et al.* in [19] that one of the key parameters to take into account when designing a SDN-based

architecture for large networks (i.e., WAN) is the propagation delay between the controller(s) and the network devices. In the case of the architecture proposed in this paper, a "good" configuration can be thought, from a qualitative point of view, as one that can reduce the communication delay between the LM/LCs and the network components without significantly increasing the management overhead (e.g., due to the coordination between LMs). For instance, while assigning a LM/LC to each network switch can optimize the communication delay, this may also significantly affect the complexity of the coordination mechanism required to harmonize management decisions (i.e., volume of messages and delay).

In this section, we present an approach to compute the placement of LCs and LMs in the distributed management and control planes for the specific case depicted in Fig. 1, where the mapping of LCs to LMs is one-to-one. Given a network topology, the approach aims at determining the number of LM/LCs to deploy, their location, as well as the switches these are connected to, with the objective of minimizing the distance (in terms of hop count) between the switches and the LM/LCs. To avoid overloading the text, we use the term LC to refer to the pair LM-LC in the rest of this section.

A. Placement Algorithm

The placement problem can be formulated as an uncapacitated facility location problem, which is known to be NP-hard. It has been addressed in several application domains, ranging from the selection of network service gateways (e.g., [20]) to the deployment of sensor networks (e.g., [21], [22]). In this work, we develop an approach based on a modified version of the leader node selection algorithm proposed by Clegg *et al.* in [23], which more closely relates to our application scenario. The algorithm, called *Pressure*, aims at determining, given a dynamic network environment, the subset of nodes on which to install monitoring points to minimize the average distance (in terms of hop count) between the monitoring entities and the network nodes. While *Pressure* has a similar objective to the one considered here, it was originally designed for dynamic network topologies and cannot be directly applied to the LC placement problem. To account for the requirements of a static topology, we modify the logic of *Pressure* and extend it to incorporate an initialization step and a terminating condition, which are essential in this case. The output of the new algorithm, which we refer to as *PressureStatic*, provides the number of LCs to deploy, their location, as well as the configuration of their mapping to network switches.

More specifically, *PressureStatic* follows a greedy approach, where the LCs are iteratively added in the network one-by-one. The main principle is to select, at each step, the location at which, if a LC is installed, will lead to the largest reduction in terms of average distance. To decide on the location, the algorithm maintains a list of locations at which it is still possible to install a LC and, at each step, it calculates the *Pressure* score of the node i associated with each of these locations as follows [23]:

$$P(i) = \sum_{j \in \mathcal{N}} \max(0, l_j - d_{i,j}) \quad (2)$$

TABLE II
NETWORK CHARACTERISTICS

Network	# Nodes	# Bidirectional Links
Abilene [24]	12	15
Geant [25]	23	37
Germany50 [26]	50	88
Deltacom [27]	92	129

where for all j in $\mathcal{N}$, l_j represents the distance between node j and the LC to which it is currently connected and for all i and j in $\mathcal{N}$, $d_{i,j}$ is the distance from node i to node j. The node with the highest score is then selected to attach the next LC and based on the updated list of selected locations, the algorithm finally determines to which LC each network node should be logically connected, so that a node is connected to its closest available LC.[2]

B. Algorithm Initialization and Terminating Condition

1) Initialization: Due to the greedy nature of the *PressureStatic* algorithm, the order according to which LC locations are selected can affect the resulting configuration. Given that this is directly driven by the choice of the first placement location, the objective of the initialization step is to determine how to best select this location. In practice, different strategies can be implemented, ranging from simple random selection methods to more sophisticated approaches which can take into account some topological characteristics of the locations. We analyze and discuss in detail the effects of the initialization step in Section VII.

2) Terminating Condition: Given that the objective of the proposed placement algorithm is to minimize the average distance of LCs to switches, the optimal configuration would be the direct one-to-one mapping of LCs to network switches. However, as previously explained, a fully distributed solution has inherent limitations in terms of scalability. The purpose of the terminating criterion is to provide a condition under which no more LC should be added to the network. To derive such a condition, we build upon the observations formulated by Heller *et al.* in [19], in which they investigated, for a wide range of network topologies, the "ideal" number of controllers to deploy to minimize the controller-to-switch communication latency. It was shown that, in most cases, the gain in terms of latency reduction tends to decrease as the number of controllers increases.

To evaluate the performance of *PressureStatic* with respect to the distance reduction improvement with the introduction of a new LC, we first define the terminating condition as a constraint on the maximum number of LCs to deploy. As such, the algorithm stops when the number of selected LCs is equal to the maximum authorized value. We then apply the algorithm to the four networks presented in Table II.

For each network, we vary the maximum authorized value from 1 to the total number of nodes in the topology and determine, for each case, the resulting average distance. We note $\bar{D}_k$ the average switch-LC distance in case the total number of LCs

[2]It should be noted that a network node is connected to one LC only.

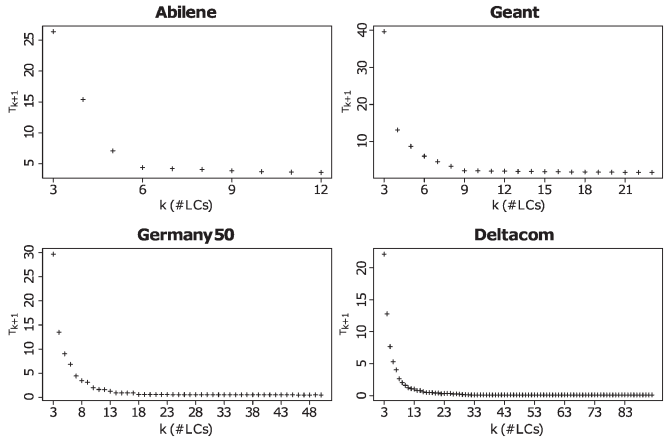

Fig. 3. Evolution of the value of τ_{k+1} with the number of LCs.

is equal to k. To measure the benefit of having multiple LCs, we then define, for all k in $[2; N]$, the parameter ω_k as follows:

$$\forall\, k \in [2; N], \quad \omega_k = \frac{\bar{D}_k}{\bar{D}_1} \tag{3}$$

ω_k represents the gain, in terms of distance reduction, when using $k \geq 2$ LCs compared to the case where only one LC is used. Based on the ω_k values, we then define, for all k in $[2; N]$, the parameter τ_{k+1} as follows:

$$\forall\, k \in [2; N-1], \quad \tau_{k+1} = \frac{\omega_{k+1} - \omega_k}{\omega_k}. \tag{4}$$

The value τ_{k+1} represents the improvement, in terms of distance gain, when an extra LC is added to the network. Fig. 3 shows the evolution of the τ_{k+1} values for the four considered topologies. As observed, all networks follow the same trend—the improvement in terms of distance gain reduction rapidly decreases until reaching a stage where it slowly converges to zero. These results corroborate the observations formulated in [19] and show that when reaching a certain number of LCs, the addition of an extra LC does not yield significant benefits in terms of average LC-to-switch distance reduction.

Based on these results, we define the terminating condition according to a threshold imposed to the gain improvement τ_{k+1}. The algorithm terminates if the value of τ_{k+1} is smaller than the input threshold. We further investigate the influence of the threshold value in Section VII. It should be noted that with the proposed approach, the minimum number of selected LCs is always equal to 2 and that the introduction of a new LC always leads to a positive improvement (i.e., reduces the average distance). The pseudo-code of the *PressureStatic* algorithm is presented in Fig. 4. Its time complexity is dominated by the number of nodes in the network, i.e., $O(N^2)$.

Pseudo-code PressureStatic Algorithm
Inputs: Set of nodes; Terminating condition.
0. Select initial LC location.
1. Compute Pressure score of all nodes not already selected as a LC location.
2. Select the node with the highest score.
3. Check if the selection satisfies the terminating condition.
if it is satisfied **then**
  End algorithm.
else
  Add selected node to the list of LC locations and go to step 1.
end if
Outputs: Set of LC locations; Switch-LC mapping.

Fig. 4. Pseudo-code of the PressureStatic algorithm.

V. Adaptive Resource Management

Initial demonstration of management applications in SDN environments relied on centralized solutions (e.g., [7], [28]). While these are well-suited for computing long term network configurations, they have limitations which make them unable to efficiently deal with dynamic resource reconfigurations. Adaptive resource management approaches call for the development of distributed solutions. In our previous work, we investigated a new approach to support dynamic resource reconfiguration functionality in the context of load-balancing (LB) [10], [11] and energy efficiency management (EM) [12] applications. This section describes the main characteristics of the proposed approach.

A. Management Functionality

The resource management decision process is distributed across the network edge nodes, which are organized into a *management substrate* (Section II-C) and embedded with dedicated management logic that enables them to perform reconfigurations based on feedback regarding the state of the network. More specifically, based on path diversity provided by MTR, the reconfiguration decisions of both applications concern the traffic splitting ratios applied at network ingresses so that individual objectives are met.

In the case of the LB functionality, the objective is to balance the load in the network by moving some traffic away from highly utilized links towards less utilized ones to disperse traffic from hot spots. In order to minimize the maximum utilization in the network, the load-balancing algorithm iteratively adjusts the splitting ratios of the traffic flows, so that traffic can be moved away from the link with the maximum utilization $l_{\max}$.

Exploiting the fact that many links in core networks are bundles of multiple physical cables [29], the objective of the EM approach is to offload as many router line cards (RLCs) as possible, which can subsequently enter sleep mode. RLCs can be full if their load is equal to their capacity, utilized if their load is not zero and less than their capacity, and non-utilized if they have zero load. One of the key decisions when adjusting the splitting ratios concerns the bundled link to consider for (a) removing traffic from, and (b) assigning that traffic to. This decision is based on a ranked list of all utilized RLCs in the network according to their load. Traffic load is iteratively moved from the least utilized RLC to more utilized ones that can accommodate this load and thus potentially fill-up their remaining capacity, without activating new RLCs in the process.

The adaptation of the splitting ratios for both applications is performed in short timescales, for instance, every 15 minutes.

B. Adaptation Process

To adjust the splitting ratios of network traffic flows, both applications rely on an *adaptation process*, which is an iterative

process triggered periodically by the nodes in the substrate (i.e., managers). It consists of a sequence of reconfiguration actions decided in a coordinated fashion.

To prevent inconsistencies between concurrent traffic splitting adjustments, these are made in a collaborative manner between nodes involved in the process, so that only one node is permitted to take reconfiguration decisions at a time. The node selected at each iteration, which we refer to as the decision-making point, is responsible for executing locally a reconfiguration algorithm, which tries to adapt the splitting ratios of local traffic flows (i.e., originating at the corresponding edge node) so that the resource optimization objective can be met.

To select a unique decision-making point, each node in the substrate is associated with an unique identifier. This can be defined based on the actual network identifier of the node (e.g., address) or determined according to some of the characteristics of the node, for example with respect to its local traffic flows. The identifiers are used to compute a ranked list of nodes, which is made available at each node in the substrate using the communication protocol defined in [9]. The node with the highest identifier is initially chosen to be the decision-making point. Upon completing the reconfiguration locally, it sends a message to the next node in the list (i.e., node with the second highest identifier), which then becomes the decision-making point for the next reconfiguration interval, and so on. The adaptation process terminates if no further adjustments can be performed or if the algorithm reaches the maximum number of permitted iterations, which is a parameter of the system.

C. Reconfiguration Algorithm

The reconfiguration algorithm is executed by the decision-making point at each iteration of the *adaptation process*. It follows a greedy process that successively recomputes the splitting ratios of the local traffic flows of the decision-making point. Based on information available locally and received from other nodes in the substrate, the algorithm determines the link l_{rem} from which traffic should be removed. In case of the LB approach, this is the link with the maximum utilization, while for the EM approach, this is the link with the least utilized RLC. The algorithm then tries to adjust the splitting ratios of the flows which contribute to the load on l_{rem} to move traffic away from the link. The outcome of the algorithm at each iteration is either positive, which means that part of a local flows can be diverted from l_{rem}, or negative if this is not possible.

The algorithm first identifies the local traffic flows that can be diverted from l_{rem}. In particular, a flow qualifies if it is routed over l_{rem} in at least one virtual topology but not all topologies, i.e., there exists at least one alternative topology in which the traffic flow is not routed over l_{rem}. Those flows are then considered iteratively until the first one that can lead to an acceptable configuration is determined. In this case, the splitting ratios of the set of topologies in which a flow is routed over l_{rem} are decreased by a factor δ^- while others are increased by a factor δ^+. The process terminates when no further local adjustments can be performed. The pseudo-code of the reconfiguration algorithm is presented in Fig. 5. Its time complexity is theoretically defined by the number of local traffic flows to reconfigure and is in the order of $O(N^2)$ in the case of a PoP-level topology with N nodes.

```
Pseudo-code Reconfiguration Algorithm
Inputs: Link statistics; Splitting ratios.
while adjustments are possible do
  Determine the link l_rem and L_flows list of flows traversing l_rem.
  if L_flows is empty then
    End algorithm.
  else
    f ← first flow in L_flows
    while canContinue and nbTested ≤ size list L_flows do
      Adjust ratios of f.
      if valid configuration then
        canContinue = false
      else
        f ← L_flows.next()
        Increment nbTested.
      end if
    end while
    Update link statistics.
  end if
end while
Output: Updated splitting ratios.
```

Fig. 5. Pseudo-code of the reconfiguration algorithm.

VI. Management Application Requirements

In this section, we show how the requirements of the two adaptive resource management applications described in the previous section can be satisfied by the functionalities and interfaces of the proposed SDN architecture.

A. Local Manager Substrate

As described in Section III-C, short to mid term management decisions are taken by the LMs organized into a *management substrate*. In practice, the range of operations that a LM can perform depends on the Management Application modules which the LM implements. Whether a LM should be involved in the decision-making process of a given application is driven by the characteristics of the switches (e.g., edge/core switch) under its responsibility.

To enable the proposed adaptive resource management approach, two functions need to be supported by a LM. The first one concerns routing decisions, which are taken by a Route Management Application (RMA) module and the second concerns the reconfiguration of splitting ratios, which is implemented by an Adaptive Resource Management Application (ARMA) module (one instance of RMA and ARMA per application). The configuration used in this paper assumes that the allocation of LMs is driven by the placement of LCs, in which case each LM is responsible for the set of switches attached to its peer LC. As such, two scenarios can be considered:

- the switches under the responsibility of a LM are core switches only. In this case, the LM implements the RMA module only.
- there is at least one edge switch under the LM responsibility. In this case, the LM implements both the RMA and ARMA modules and is responsible for configuring the splitting ratios of the local traffic flows of all the edge switches to which it is connected.

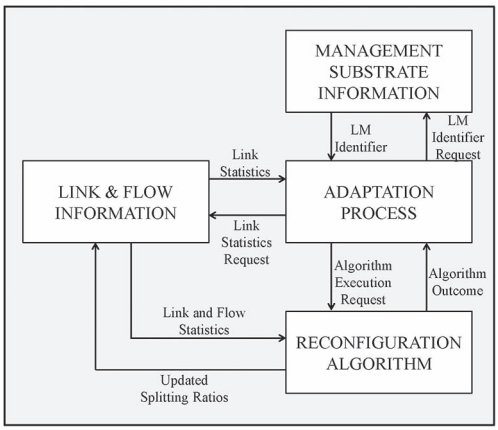

Fig. 6. Adaptive Resource Management Application module.

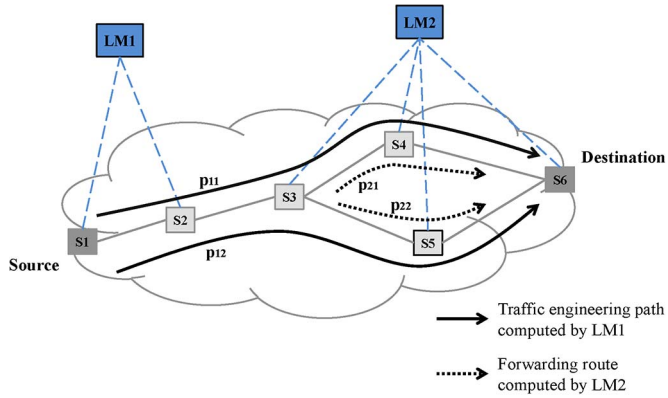

Fig. 7. Example of possible inconsistencies between routing decisions.

To realize the functionality of a given application, the LMs involved in the decision-making process may need to communicate through the *management substrate*. In the case of multiple applications, a separate management substrate needs to be computed for each application and implemented in the relevant ARMA module. Each substrate defines the set of neighbors of the LMs and their substrate identifier. These identifiers are used by the ARMA module to compute the ordered list of nodes in the substrate and to select the decision-making point at each iteration of the *adaptation process* (see Section V-B).

B. Management Application Functionality Requirements

Long Term Configurations Long term configurations are computed by the centralized management system. In the context of the resource management scenario considered here, these concern the computation of the MTR planes and the structure of the substrate associated with each management application. The MTR plane computation algorithm is executed in an offline fashion by the Application Orchestrator. Based on information retrieved from the Global Network View component about the physical network topology, the algorithm determines the number and structure of the virtual topologies needed to satisfy the path diversity requirements. The configuration of each plane is then passed to the RMA module of all LMs through the O-M interface. The structure of each substrate is computed by an offline algorithm implemented in the LM Substrate Orchestrator. The resulting structure information is passed to the RMA and ARMA modules of associated LMs.

Adaptive Resource Management The logic to execute the load-balancing and energy management functionality is implemented by the ARMA module of each involved LM. As depicted in Fig. 6, the ARMA implements four components. The Adaptation Process component controls the execution of the reconfiguration algorithm (see Section V-C), which adjusts the splitting ratios of the local traffic flows controlled by the LM. The Management Substrate Information component maintains information about the structure of the management substrate, such as the set of neighbor LMs and their ordered list. The Link & Flow Information component consists of multiple tables containing information about the network links (e.g., capacity, utilization etc.) and characteristics of the local flows (e.g., splitting ratio configuration, demand etc.). Network statistics are updated based on information received from other LMs through the substrate or retrieved from the local LM monitoring module. Based on the ARMA module, the LMs associated with at least one edge switch periodically (e.g., every 15 minutes) compute the vectors of splitting ratios for every source-destination pair in the network. These are then sent to the LCs to which the LMs are connected and stored for future enforcement.

C. Routing Functionality Requirements

The RMA module interacts with the Routing module which implements methods to compute routing parameters (e.g., Dijkstra shortest path). In the scenario considered here, two routing configurations need to be computed that we refer to as traffic engineering (TE) and forwarding (FW). The TE configuration is computed at every LM involved in the splitting ratio reconfiguration process and represents the full network path from any source switch controlled by the LM to any destination switch in every MTR plane. The FW configuration is computed at every LM and is used to determine the interface on which packets (received at any network switch) need to be sent to reach their destination via the shortest path in each MTR plane. The results are sent by each LM to its attached LC, which then configures the forwarding policies in all the switches under its responsibility.

In most network topology cases, the switches along the path between a S-D pair may not all be under the responsibility of a single LM. As a result, inconsistencies between the TE and FW configurations may occur. In particular, this can happen when multiple equal cost shortest paths exist. To illustrate this issue, we consider the simple example depicted in Fig. 7. All links have a weight equal to 1. The full path from S1 to S6 is computed at LM_1. There are two equal shortest paths between S1 and S6: path $p_{11} : \{1; 2; 3; 4; 6\}$ and path $p_{12} : \{1; 2; 3; 5; 6\}$. Assume that path p_{12} is selected by LM_1. To route packets, forwarding policies need to be implemented in each of the switches. Due to its scope of responsibility, LM_1 can only decide how to forward packets from S1 and S2; it does not have any control on how the packets for S6 are routed from S3 onwards. Switches S3, S4, S5 and S6 are controlled by LM_2. There are two equal cost shortest paths from S3 to S6: path $p_{21} : \{3; 4; 6\}$ and path $p_{22} : \{3; 5; 6\}$. To ensure the consistency

with the TE path considered by LM_1, LM_2 should choose path p_{22} when deriving the forwarding policies to apply for packets from S1 to S6. To avoid inconsistent decisions, all LMs should therefore apply a common path selection strategy (for example based on the identifier of the interfaces). This ensures that the FW decisions are in accordance with the TE decisions.

D. Switch Configuration

To enforce TE decisions at the network level, incoming packets need to be marked at the network edges with the identifier of the MTR plane to which they have been assigned based on the computed splitting ratios. Although OpenFlow does not currently support MTR, the latest version of the specifications introduces support for MPLS labeling and VLAN tagging [14]. In addition, a proposal for multipath routing [30], which relies on the group option defined in the OpenFlow protocol, has been released by the ONF. As such, we believe that the current protocol could be easily extended to support MTR.

The TE application requires that traffic splitting happens at network edges only. Packets that belong to the same TCP flow are always assigned to the same topology and no further adjustment is permitted along the route. This ensures that all packets in one TCP flow follow only a single path to the destination, as such avoiding out-of-order delivery issues that deteriorate the performance of TCP [31]. As a result, this has implications on the way incoming packets need to be processed in the different switches along the path between a S-D pair. More specifically, switches can act as source or transit depending on how they process packets. A switch acts as source switch for incoming packets if a) it is an edge switch, *and* b) packets belong to one of the switch's local traffic flows. In this case, the switch needs to assign packets to the relevant MTR plane and execute the following steps:

1) Determine to which local traffic flow the packet belongs.
2) Enforce the relevant splitting ratios to determine the MTR plane on which to route the packet.
3) Mark the header with the identifier of the selected plane.
4) Forward the packet according to the configuration of the selected MTR plane.

A switch acts as a transit for incoming packets if a) it is an edge switch but incoming packets do not belong to one of the switch's local traffic flows, *or* b) it is a core switch. In this case, packet processing consists mainly in forwarding the packets according to the configuration of the MTR plane to which they have been assigned, i.e.,

1) Determine to which local traffic flow the packet belongs and to which MTR plane it is assigned.
2) Forward the packet according to the configuration of the relevant MTR plane.

Each packet is processed according to the information retrieved from the packet header (e.g., source IP, destination IP, MTR_ID etc.), which is used to match the packet against flow entries in the different Flow Tables implemented in each switch. Table entries are pro-actively configured by the LC to which each switch is connected based on routing and splitting ratio configurations information maintained by the Storage module. In case of a table miss (i.e., no entry for the traffic flow to which the packet belongs), switches should be configured to send the packet to their LC, which then decides on the processing to apply (i.e., to create new table entries).

To balance the traffic across the different MTR planes, a hashing scheme, such as the one proposed by Cao *et al.* in [32], could be implemented in each edge switch and parametrized for each traffic flow (i.e., S-D pair) according to the values of the splitting ratios. This, however, suggests the availability of a mechanism to enable the programmability of the hashing function, which is currently not possible with OpenFlow. An alternative solution could use a simple hack based on the available options (e.g., by configuring the bitmasks associated with some of the flow entry matching fields). Although this may have the advantage of not requiring any extension to the protocol, it may not provide the same level of control as the hashing method.

VII. Evaluation

The logic of the LM and LC Orchestrator Modules and the Adaptive Resource Management Application Module has been implemented in a Java-based simulated environment. This section presents the results of the performance evaluation of the LC placement algorithm and the load-balancing and energy management approaches based on real network topologies and traffic traces.

A. Placement Algorithm Performance

We evaluate the performance of *PressureStatic* based on the four network topologies presented in Table II. Given that its output is affected by the location on which the first LC is added, we consider, for all topologies, all possible initial locations, covering as such the totality of the input space.

1) Influence of the Initial and Terminating Criteria: As described in Section IV-B2, the terminating condition is defined according to the threshold imposed to the distance reduction gain improvement. From Fig. 3, it can be inferred that setting the value of the threshold more than 10% will result, for all topologies, in the selection of 2 LCs only. To investigate how the threshold influences the number of selected LCs, we apply the algorithm using improvement values of 2.5%, 5% and 10%. The results are presented as boxplots in Fig. 8.

Several observations can be made from the results. It can first be noted that, for all topologies, the number of selected LCs decreases as the value of the threshold increases, which is consistent with the results depicted in Fig. 3. Larger threshold values force the algorithm to terminate prematurely. In addition, it can be observed that the size of the network (i.e., number of nodes) is not the main factor affecting the number of selected LCs. On average, a similar number of LCs are selected in the Geant, Germany50 and Deltacom networks for all cases. This can be explained by the definition of the terminating threshold which considers distance reduction gain in absolute values. Finally, the boxplots depict a variation in the number of selected LCs, which shows that the choice of the first LC location influences the output of the algorithm.

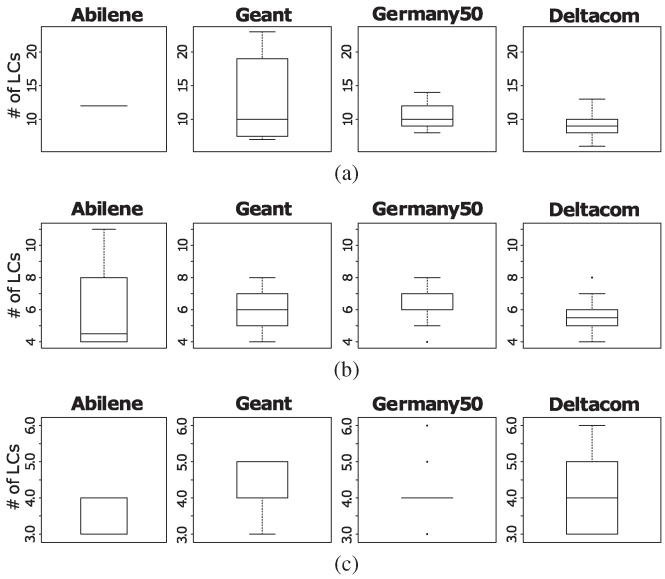

Fig. 8. Number of selected local controllers. (a) Threshold 2.5%. (b) Threshold 5%. (c) Threshold 10%.

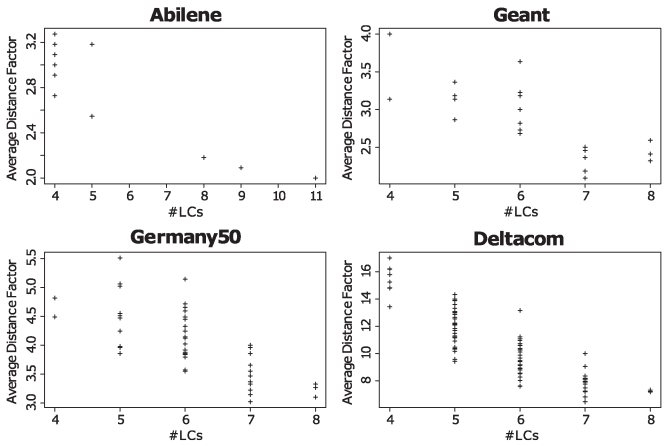

Fig. 9. Number of selected LCs vs. average distance factor.

To determine how to choose the initial location, we investigate the existence of a correlation between the number of selected LCs and the actual topological location of the node to which the first LC is connected. Given that the objective of the placement algorithm is to reduce the average LC-switch distance, we characterize the node location according to its average distance factor which is defined as follows:

$$\forall\, i \in \mathcal{N}, \quad \Delta(i) = \frac{\sum_{j \in \mathcal{N} \setminus \{i\}} d_{i,j}}{|\mathcal{N}|^2} \tag{5}$$

The value of Δ represents the proximity of a node in terms of average distance to the rest of the network nodes. The smaller the value is, the closer the node is, on average, to the other network nodes.[3] For each network node i, we calculate its value $\Delta(i)$ and record the number of selected LCs when the first LC is connected to node i. The correlation between the values of the average distance factor of the initial LC location and the number of selected LCs is depicted in Fig. 9 for the four considered networks with a threshold value of 5%.

[3]By definition, the Δ values depend on the size of the network.

TABLE III
PLACEMENT ALGORITHM PARAMETER SETTINGS

Network size	Number of nodes	Average distance factor	Threshold value
Small	less than 30	$\max(\Delta)$	10%
Medium	30 to 100	$\min(\Delta)$	5%
Large	more than 100	$\min(\Delta)$	2.5%

In all cases, the number of LCs tends to decrease as the value of Δ increases. In other words, when the initial location is on average close to every other node, a large number of LCs tends to be selected by the algorithm. In contrast, if the initial location is off-centered, a small number of LCs tends to be selected. As indicated by (4), the value of the improvement factor τ_{k+1} depends on the previous gain ω_k. When the first LC is attached to an off-centered node, the algorithm tends to select a more "central" location for the second LC in order to maximize the distance reduction. As a result, this leads to a substantial gain improvement. In comparison, the subsequent additions of LCs produce lower reduction in terms of distance, and, as such, do not satisfy the threshold constraint. In contrast, when the initial location is more "central", the rates of distance reduction associated with each additional LC tend to be on average lower but comparable between themselves, which leads to the selection of a larger number of LCs. Similar results were obtained with the other threshold values but are not reported here due to space limitations.

The results demonstrate that the number of LCs selected for a given network can be tuned by controlling the settings of the terminating condition and the initial LC placement. In practice, it is expected that the number of LCs should increase with the size of the network topology in order to minimize both the LC-switch and LC-LC distances. This can be translated into the following parameter settings: low threshold value and central initial LC position for large scale networks, and high threshold value and off-centered initial position for smaller scale networks. The settings for the topologies considered in this paper are presented in Table III.

2) Heuristic Performance: Clegg *et al.* showed in [23] that the placement computed by the *Pressure* algorithm significantly outperforms any random configuration in terms of LC-switch distance. Another factor to consider for evaluating the performance of the proposed placement heuristic is to compare its output to the optimal placement of the same number of LCs (i.e., optimum average switch-LC distance). In order to derive the optimum value, we formulate the placement of n LCs as an Integer Linear Programming (ILP) problem with the followings parameters. Let N_{LC} be the total number of LCs to deploy. For all i and j in $\mathcal{N}$, let $d_{i,j}$ be the distance between i and j. For all i in $\mathcal{N}$, let x_i be the binary variable equal to 1 if a LC is attached to i, 0 otherwise. In addition, for all i and j in $\mathcal{N}$, let $y_{i,j}$ be the binary variable equal to 1 if node i is connected to LC attached to node j, 0 otherwise. Finally, for all i in $\mathcal{N}$, let l_i be the distance between node i and the LC to which it is connected.

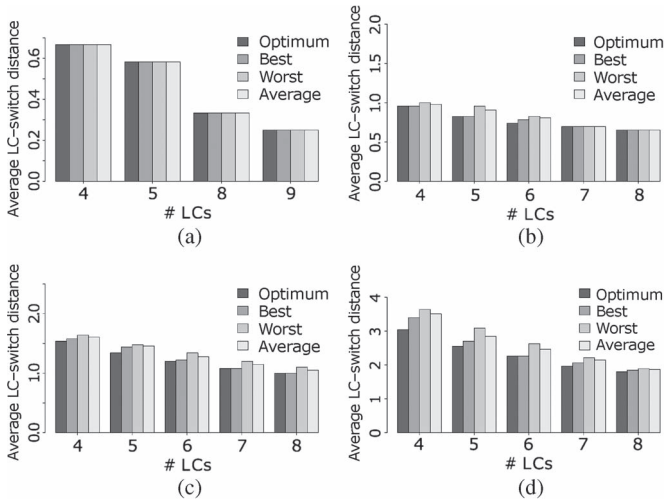

Fig. 10. Performance of PressureStatic with threshold 5% vs. optimum average LC-switch distance. (a) Abilene. (b) Geant. (c) Germany50. (d) Deltacom.

The objective of the ILP is to determine the values of x_i and $y_{i,j}$ which minimize the average LC-switch distance, i.e., formally:

$$\text{minimize} \frac{1}{|\mathcal{N}|} \cdot \sum_{i \in \mathcal{N}} l_i \tag{6}$$

subject to the following constraints:

$$\forall\, i \in \mathcal{N}, \quad \sum_{j \in \mathcal{N}} d_{i,j} \cdot y_{i,j} = l_i \tag{7}$$

$$\forall\, i \in \mathcal{N}, \quad \sum_{j \in \mathcal{N}} y_{i,j} = 1 \tag{8}$$

$$\forall\, i \in \mathcal{N}, j \in \mathcal{N}, \quad y_{i,j} \leq x_j \tag{9}$$

$$\forall\, j \in \mathcal{N}, \quad x_j \leq \sum_{i \in \mathcal{N}} y_{i,j} \tag{10}$$

$$\sum_{j \in \mathcal{N}} x_j \leq N_{LC}. \tag{11}$$

Constraint (7) defines the LC-switch distance. Constraint (8) ensures that each switch is connected to one LC only. Constraint (9) guarantees that a switch is associated with a location only if a LC is attached there and constraint (10) forces the ILP to position a LC on a location only if at least one switch is associated with this location. Finally, constraint (11) ensures that the total number of LCs is at most equal to N_{LC}.

For each network topology in Table II, we apply the ILP for all values of N_{LC} relevant to that network (derived from Fig. 9). Based on the algorithm output, we then determine the optimal LC-switch distance and compare it to the one obtained with *PressureStatic* for the same number of selected LCs. The results are shown in Fig. 10. In the case of *PressureStatic*, the best, average and worst distance values recorded for the considered configuration are reported. It is worth noting that since switches and LCs can be physically attached to the same location (distance equal to 0), the average LC-switch distance can be lower than 1.

As observed, the placement computed by *PressureStatic* gives close to optimal performance in terms of average LC-switch distance. In the case of the Abilene network, the proposed algorithm is able to compute the optimal placement. In all other cases, the deviation from the optimum decreases as the number of LCs increases. The highest deviation (16%) is obtained with Deltacom and 4 LCs.

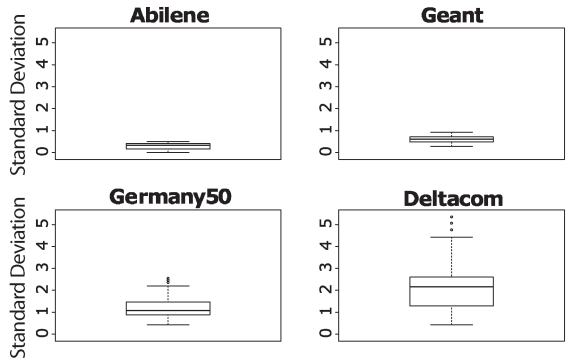

Fig. 11. Standard deviation of the cluster size with a threshold of 5%.

TABLE IV
NETWORK BUNDLED LINKS (BLS) CONFIGURATION

Network	Abilene		Geant			Germany50
# MTR planes	4		4			4
# LCs	4		3			7
# Switches per LC	3;3;3;3		10;8;5			5;10;7;7;7;8;6
Type (T)	T1	T2	T1	T2	T3	T1
BL capacity (Gbps)	10	2	10	2	1	10
BL size	4	4	4	2	2	4
RLC capacity (Gbps)	2.5	0.5	2.5	1	0.5	2.5
# BLs	28	2	37	17	20	176
# RLCs	112	8	148	34	40	704

3) Placement Optimization Objective: The algorithm aims at minimizing the average LC-switch distance. The number of LM/LCs, however, may also be driven by other parameters. In the proposed architecture, each LC is logically connected to a set of switches forming clusters. In practice, the number of switches attached to an LC can affect the volume of information which needs to be maintained and processed by the LC. To investigate the effect of *PressureStatic* on the cluster size distribution, we compute the standard deviation of the size of the clusters attached to each LC for each network and configuration. To account for the variation incurred by the choice of the first LC location, we plot the results as boxplots in Fig. 11. The closer the value of the standard deviation is to 0, the more uniformly distributed in terms of size are the clusters.

The results show that *PressureStatic* can lead to unbalanced clusters. The heterogeneity in terms of cluster sizes tends to increase as the number of nodes in the network increases. The trade-off between latency reduction and homogeneity of the volume of information to maintain at each LC could therefore be taken into account in the placement objective to control the clustering of switches. Increasing the homogeneity of the cluster size may, however, lead to the selection of a larger number of LM/LCs, which raises challenges regarding the choice of the management substrate structure to use to organize the LMs (Section II-C). In [11], we showed that while simple structures such as the full-mesh or ring models suit well the case where a small number of nodes is involved in the substrate, these have limitations when this number increases. In this case, the use of a more sophisticated structure such as the one presented in [9] should be considered. The proposed structure, which is

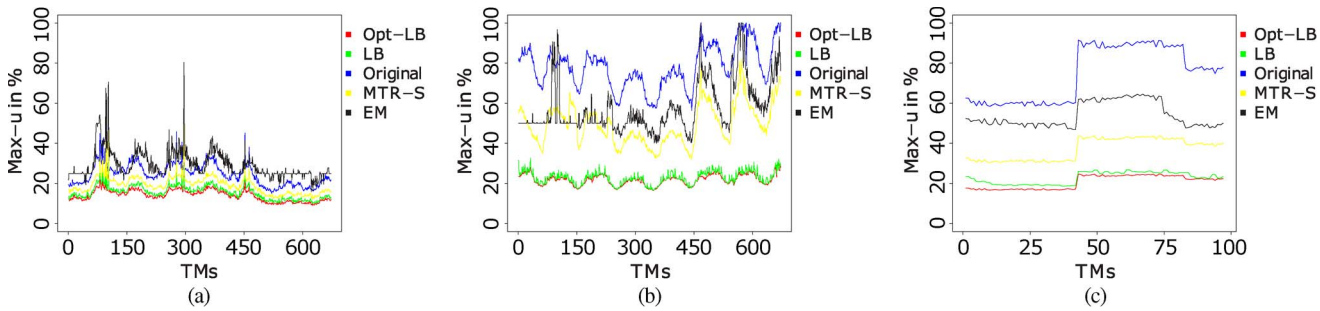

Fig. 12. Evolution of max-u. (a) Abilene. (b) Geant. (c) Germany50.

a hybrid model, offers a trade-off between the full-mesh and the ring models in terms of communication cost and volume of information that needs to be maintained by each substrate node.

B. Adaptive Resource Management Scheme Performance

We evaluate the performance of the load-balancing (LB) and energy management (EM) approaches, described in Section V, under the LM plane configuration determined by the placement algorithm for the Abilene, Geant and Germany50 networks for which real data traces are available (this is not the case for Deltacom). In order to take into account a wide range of traffic conditions, we consider a period of 7 days for both Abilene [33] and Geant [25]. In the case of Germany50, a period of one day is considered given the available data [34]. In all cases, adaptation is performed at a frequency of 15 minutes. The configurations used in each topology are summarized in Table IV.

We compare the performance in terms of maximum link utilization (max-u) and number of active line cards (nbRLCs) obtained for each traffic matrix (TM) with the five following schemes:

- **Original**: the original link weight settings are used in the original topology and no adaptation is performed.
- **Static MTR (MTR-S)**: static splitting ratios (do not adaptively change) are set equal to the inverse of the capacity of the bottleneck bundled link in each virtual topology.
- **Load-Balancing (LB)**: the considered LB approach.
- **Energy Management (EM)**: the considered EM approach.
- **Optimum Load-Balancing (Opt-LB)**: the routing problem is defined as a MultiCommodity Flow problem [35] and the glpsol GLPK (GNU Linear Programming Kit) linear programming solver [36] is used to compute the optimal max-u for each traffic matrix.

The evolution of the max-u at 15 minute intervals obtained with the five schemes over the time period considered in each network is shown in Fig. 12. As can be observed, LB outperforms the Original, MTR-S and EM schemes in all cases and obtains close to optimal performance. The deviation from the optimum max-u is equal to 8.78%, 5.6%, and 10.1% in the Abilene, Geant and Germany50 networks, respectively. In addition, a deviation of less than 10% is obtained for 96.42% of the TMs in the case of Abilene, and 98.25% and 92.8% for Geant and Germany50, which indicates that the proposed scheme performs uniformly well.

To analyze the performance of the EM approach in terms of energy gain, we determine the deviation between the number of

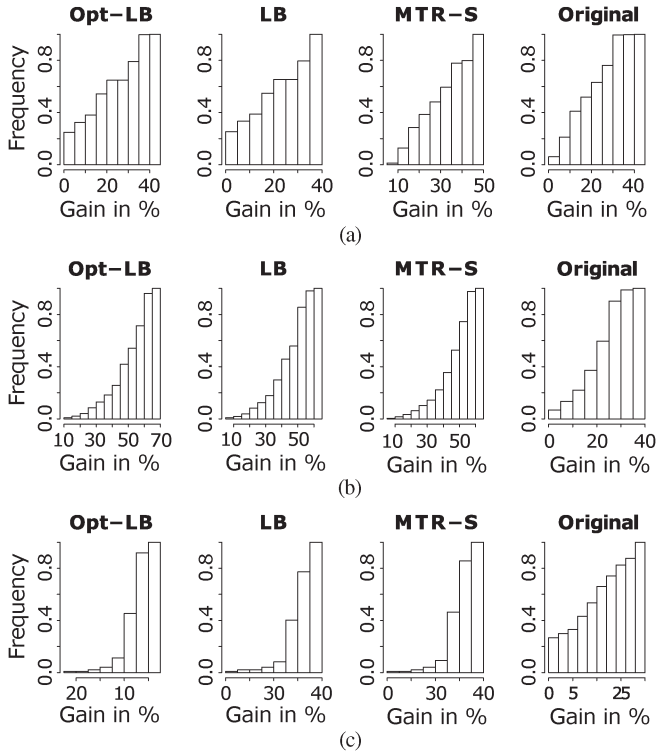

Fig. 13. Cumulative frequency graphs of the gain in terms of active line cards obtained by EM compared to the other schemes. (a) Abilene. (b) Geant. (c) Germany50.

active line cards used by EM and the one obtained with the other schemes. The results are shown as cumulative frequency graphs in Fig. 13. It can first be observed that a positive gain is obtained in all cases, which shows that EM always uses the lowest number of RLCs to route the traffic. The best performance is achieved when compared to the schemes that balance the traffic (Opt-LB, LB and MTR-S), which can be explained by the antagonistic nature of the two objectives. The gain compared to LB is on average equal to 20.90%, 44.82%, and 30.47% for the Abilene, Geant and Germany50 networks, respectively. In addition, EM performs better than the Original scheme by concentrating the traffic on a smaller number of links. In this case, the gain is equal to 19.21%, 21.05%, and 10.08% for the three networks, respectively.

The results demonstrate that a significant reduction in terms of resource utilization can be achieved by the proposed schemes under the configuration of the distributed management plane computed by the placement algorithm.

VIII. Related Work

In contrast to traditional network architectures, local control functions are moved away from network elements to remote controllers in SDN. As a result, this can lead to the creation of new bottlenecks and potentially significant overhead, depending on the type of management applications to consider [2]. While using a centralized controller with a network-wide view has the benefit of facilitating the implementation of the control logic, it also presents limitations, especially in terms of scalability as the size and dynamics of the network increase. Different approaches have been proposed in the literature to overcome the limitations of the single centralized controller model, e.g., [4]–[6], [37].

The approach presented in [4] by Yeganeh *et al.* is based on two levels of controllers. Distributed controllers in the lower level operate on locally-scoped information, while decisions which require network-wide knowledge are taken by a logically centralized root controller. A hierarchical solution for Wide Area Networks (WAN) has also been proposed by Ahmed *et al.* in [6]. In their architecture, the network is divided into multiple zones of control, on top of which a centralized management layer implements management operation functionality and services. In contrast to hierarchical solutions, fully distributed designs have been proposed in [5] and [37]. The platform presented in [5] aims at facilitating the implementation of distributed control planes by abstracting network resources as data objects stored in a Network Information Base. In [37], the authors introduce HyperFlow, a physically distributed but logically centralized event-based OpenFlow control plane. Due to its centralized control logic, HyperFlow requires state synchronization mechanisms and targets pro-active management operations. The impact of control state consistency on the performance of a load-balancing application has been investigated by Levin *et al.* in [38]. While most of the previous approaches have focused on the interface between the data and control planes, the policy-based framework developed by Kim *et al.* targets the interface between the control platform and the network management logic [1]. Jain *et al.* report their experience in deploying a SDN-based WAN to connect Google datacenters [7]. Each datacenter site is controlled by a set of OpenFlow-based network control servers connected to a centralized SDN gateway which implements a logically centralized traffic engineering (TE) application.

The approaches described above realize distributed control planes. Most of them, however, consider a centralized solution to implement network applications, which is not adequate for reactive and adaptive functionalities. The framework proposed in this paper advances the state-of the art by enabling adaptive resource management through a distributed management plane (LMs), while relying on the support of a centralized management system for long term operations. In addition, it is interesting to note that most of the SDN approaches emanating from outside the network management community do not make a clear separation between control and management functionalities (e.g., [4], [5]) and usually disregard the implications of management operations (especially dynamic ones). In this paper, we advocate a model that separates control and management logic. This distinction was also taken into account in the approach presented in [6]. However, in contrast to our framework, this relies on a centralized management plane.

A key issue when deploying a distributed control plane concerns the controller placement problem. In [39], Bari *et al.* proposed an approach to dynamically determine the number and location of controllers based on the network conditions. In practice, the allocation of controllers should also be driven by the requirements of the network applications to implement. In our algorithm, this is taken into account through the initial and terminating conditions. A dynamic approach is thus more geared towards applications sensitive to traffic fluctuations. A different objective has been considered by Hu *et al.* in [40] where the goal is to maximize the reliability in terms control paths. Their approach assumes that the number of controllers to deploy is given, which may not be easy to determine *a priori* and is considered as a variable in our work.

Another line of research related to the work presented in this paper concerns network resource management for load-balancing purposes and energy savings. To overcome the limitations of current offline TE functionality, online approaches that are able to react to current conditions in short timescales have been developed [41]. The objective is to dynamically adapt the settings based on real-time information received from the network in order to better utilize network resources. Most of the previous approaches rely on a centralized manager to compute new configurations (e.g., [42], [43]). While distributed solutions have been proposed in [44]–[46], these target MPLS-based networks. In contrast, this paper presents an adaptive and decentralized approach for IP networks. A decentralized IP-based TE mechanism has also been proposed in [47]. Compared to our approach, however, decisions are made by each node in the network and as such, may be associated with a non negligible signalling overhead. Finally, in the context of SDN, Agarwal *et al.* proposed an approach to perform TE in SDN environments [28] in the case where not all switches are embedded with SDN capabilities. In this case, SDN-compliant switches are controlled by a centralized SDN controller while others implement traditional hop-by-hop routing functions.

IX. Conclusion

This paper presents a new SDN-based management and control framework for fixed backbone networks, which provides support for both static and dynamic resource management applications. Its architecture is compatible with the generic ONF SDN model and consists of three layers which interact with each other through a set of interfaces. Based on its modular structure, the framework makes a clear distinction between the management and control logic which are implemented by different planes, offering as such improved deployment advantages. To demonstrate the benefits of the proposed framework, we show how its functionality and interfaces can be used to support the requirements of two distributed adaptive resource management applications whose performance is evaluated in terms of resource utilization reduction. We also present a new placement algorithm to compute the configuration of the distributed management and control planes and investigate how

the degree of distribution can be controlled based on different parameters.

In future extensions of this research, we plan to enhance the proposed placement algorithm by considering other costs and constraints (e.g., maintaining the homogeneity of cluster size, applying constraint on the volume of information stored at each LC etc.). We also plan to develop a mechanism to enable unequal cost splitting in OpenFlow. Finally, future work will investigate how the proposed framework can be used to realize other types of management applications (e.g., cache management).

REFERENCES

[1] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114–119, Feb. 2013.

[2] S. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 136–141, Feb. 2013.

[3] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, future of programmable networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, 2014.

[4] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proc. HotSDN*, Helsinki, Finland, 2012, pp. 19–24.

[5] T. Koponen *et al.*, "Onix: A distributed control platform for largescale production networks," in *Proc. USENIX*, Vancouver, BC, Canada, 2010, pp. 1–6.

[6] R. Ahmed and R. Boutaba, "Design considerations for managing wide area software defined networks," *IEEE Commun. Mag.*, vol. 52, no. 7, pp. 116–123, Jul. 2014.

[7] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined wan," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, Oct. 2013.

[8] M. Charalambides, G. Pavlou, P. Flegkas, N. Wang, and D. Tuncer, "Managing the future Internet through intelligent in-network substrates," *IEEE Netw.*, vol. 25, no. 6, pp. 34–40, Nov. 2011.

[9] D. Tuncer, M. Charalambides, H. El-Ezhabi, and G. Pavlou, "A hybrid management substrate structure for adaptive network resource management," in *Proc. ManFI*, Krakow, Poland, May 2014, pp. 1–7.

[10] D. Tuncer, M. Charalambides, G. Pavlou, and N. Wang, "Towards decentralized and adaptive network resource management," in *Proc. CNSM, Mini-Conference*, Paris, France, Oct. 2011, pp. 1–6.

[11] D. Tuncer, M. Charalambides, G. Pavlou, and N. Wang, "DACoRM: A coordinated, decentralized, and adaptive network resource management scheme," in *Proc. NOMS*, Maui, HI, USA, Apr. 2012, pp. 417–425.

[12] M. Charalambides, D. Tuncer, L. Mamatas, and G. Pavlou, "Energy-aware adaptive network resource management," in *Proc. IM*, Ghent, Belgium, May 2013, pp. 369–377.

[13] "Software-defined networking: The new norm for networks," Open Networking Found., Palo Alto, CA, USA, Apr. 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[14] "OpenFlow Specifications v.1.4.0," Open Networking Found., Palo Alto, CA, USA, Oct. 2013. [Online]. Available: https://www.opennetworking.org/sdn-resources/onf-specifications/openflow

[15] "SDN architecture," Open Networking Found., Palo Alto, CA, USA, Jun. 2014. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf

[16] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault, "Internet Engineering Task Force," Multi-Topology (MT) Routing in OSPF, RFC 4915, Jun. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4915.txt

[17] T. Przygienda, N. Shen, and N. Sheth, *M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)*, RFC 5120, Feb. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5120.txt

[18] S. Gjessing, "Implementation of two resilience mechanisms using multi topology routing and stub routers," in *Proc. AICT-ICIW*, Guadeloupe, French Caribbean, Feb. 2006, pp. 29–34.

[19] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. HotSDN*, Helsinki, Finland, 2012, pp. 7–12.

[20] R. Cohen and G. Nakibly, "A traffic engineering approach for placement and selection of network services," *IEEE/ACM Trans. Netw.*, vol. 17, no. 2, pp. 487–500, Apr. 2009.

[21] S. Ray, R. Ungrangsi, D. Pellegrini, A. Trachtenberg, and D. Starobinski, "Robust location detection in emergency sensor networks," in *Proc. INFOCOM*, San Francisco, CA, USA, Mar. 2003, vol. 2, pp. 1044–1053.

[22] D. F. Pellegrini and R. Riggio, "Leakage detection in waterpipes networks using acoustic sensors and identifying codes," in *Proc. IEEE PerSense*, San Diego, CA, USA, Mar. 2013, vol. 2, pp. 1044–1053.

[23] R. Clegg, S. Clayman, G. Pavlou, L. Mamatas, and A. Galis, "On the selection of management/monitoring nodes in highly dynamic networks," *IEEE Trans. Comput.*, vol. 62, no. 6, pp. 1207–1220, Jun. 2013.

[24] The Abilene Internet 2 Topology. [Online]. Available: http://www.Internet2.edu/pubs/200502-IS-AN.pdf

[25] The GEANT Topology, 2004. [Online]. Available: http://www.dante.net/server/show/nav.007009007

[26] The Germany50 Topology, 2004. [Online]. Available: http://sndlib.zib.de/

[27] The Deltacom Topology, 2010. [Online]. Available: http://www.topology-zoo.org/maps/Deltacom.jpg/

[28] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *Proc. INFOCOM*, Turin, Italy, Apr. 2013, pp. 2211–2219.

[29] *IEEE Standard for Local and Metropolitan Area Networks: Link Aggregation*, IEEE Std. 802.1AX, Nov. 2008.

[30] OpenFlow Multipath Proposal. [Online]. Available: http://archive.openflow.org/wk/index.php/Multipath_Proposal

[31] M. Laor and L. Gendel, "The effect of packet reordering in a backbone link on application throughput," *IEEE Netw.*, vol. 16, no. 5, pp. 28–36, Sep./Oct. 2002.

[32] Z. Cao, Z. Wang, and E. Zegura, "Performance of hashing-based schemes for Internet load balancing," in *Proc. INFOCOM*, Tel Aviv, Israel, 2000, vol. 1, pp. 332–341.

[33] The Abilene Topology and Traffic Matrices Dataset, 2004. [Online]. Available: http://www.cs.utexas.edu/~yzhang/research/AbileneTM/

[34] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0-survivable network design library," in *Proc. INOC*, Spa, Belgium, Apr. 2007, pp. 1–6.

[35] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Upper Saddle River, NJ, USA: Prentice-Hall, 1982.

[36] GNU Linear Programming Kit (GLPK). [Online]. Available: http://www.gnu.org/software/glpk/

[37] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for openflow," in *Proc. INM/WREN*, San Jose, CA, USA, 2010, pp. 3–9.

[38] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized?: State distribution trade-offs in software defined networks," in *Proc. HotSDN*, Helsinki, Finland, 2012, pp. 1–6.

[39] M. Bari *et al.*, "Dynamic controller provisioning in software defined networks," in *Proc. CNSM*, Zurich, Switzerland, Oct. 2013, pp. 18–25.

[40] Y.-N. Hu, W.-D. Wang, X.-Y. Gong, X.-R. Que, and S.-D. Cheng, "On the placement of controllers in software-defined networks," *J. China Univ. Posts Telecommun.*, vol. 19, no. S2, pp. 92–97, Oct. 2012.

[41] N. Wang, K. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for Internet traffic engineering," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 1, pp. 36–56, 2008.

[42] N. Wang, K. Ho, and G. Pavlou, "Adaptive multi-topology IGP based traffic engineering with near-optimal network performance," in *Proc. NETWORKING*, vol. 4982, *Lecture Notes in Computer Science*, A. Das, H. Pung, F. Lee, and L. Wong, Eds., Berlin/Heidelberg, 2008, pp. 654–666, Springer.

[43] M. Zhang, C. Yi, B. Liu, and B. Zhang, "GreenTE: Power-aware traffic engineering," in *Proc. ICNP*, Kyoto, Japan, 2010, pp. 21–30.

[44] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: responsive yet stable traffic engineering," in *Proc. SIGCOMM*, Philadelphia, PA, USA, Aug. 2005, vol. 35, pp. 253–264.

[45] F. Cuomo, A. Cianfrani, M. Polverini, and D. Mangione, "Network pruning for energy saving in the Internet," *Comput. Netw.*, vol. 56, no. 10, pp. 2355–2367, Jul. 2012.

[46] V. Foteinos, K. Tsagkaris, P. Peloso, L. Ciavaglia, and P. Demestichas, "Operator-friendly traffic engineering in IP/MPLS core networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 11, no. 3, pp. 333–349, Sep. 2014.

[47] S. Fischer, N. Kammenhuber, and A. Feldmann, "REPLEX: Dynamic traffic engineering based on wardrop routing policies," in *Proc. CoNEXT*, Lisboa, Portugal, 2006, pp. 1–12.

Daphne Tuncer received the "Diplôme d'ingénieur de Télécom SudParis" in 2009. She received the Ph.D. from the Electronic and Electrical Engineering at University College London, U.K., in November 2013. She is a postdoctoral researcher in the Department of Electronic and Electrical Engineering at University College London, U.K. Her research interests are in the areas of software-defined networking, network self-management, adaptive network resource management, energy efficiency and cache/content management.

Marinos Charalambides received the B.Eng. degree (First Class Hons.) in electronic and electrical engineering, the M.Sc. degree (Distinction) in communications networks and software, and the Ph.D. degree in policy-based management, all from the University of Surrey, U.K., in 2001, 2002 and 2009, respectively. He is a senior researcher at University College London. He has been working in a number of European and U.K. national projects since 2005 and his current research interests include software-defined networking, in-network caching, energy-aware networking and on-line traffic engineering. He is on the technical program committees of the main network and service management conferences.

Stuart Clayman received the Ph.D. degree in computer science from University College London in 1994. He has worked as a Research Lecturer at Kingston University and at UCL. He is currently a Senior Research Fellow at UCL EEE department. He co-authored over 30 conference and journal papers. His research interests and expertise lie in the areas of software engineering and programming paradigms; distributed systems; virtualised compute and network systems, network and systems management; networked media; and knowledge-based systems. He has been involved in several European research projects since 1994. He also has extensive experience in the commercial arena undertaking architecture and development for software engineering, distributed systems and networking systems. He has run his own technology start-up in the area of NoSQL databases, sensors and digital media.

George Pavlou received the Diploma in engineering from the National Technical University of Athens, Greece, and the M.Sc. and Ph.D. degrees in computer science from University College London, U.K. He is a Professor of Communication Networks in the Department of Electronic and Electrical Engineering, University College London, U.K., where he coordinates research activities in networking and network management. His research interests focus on networking and network management, including aspects such as traffic engineering, quality of service management, autonomic networking, information-centric networking, grid networking and software-defined networks. He has been instrumental in a number of European and U.K. research projects that produced significant results with real-world uptake and has contributed to standardization activities in ISO, ITU-T and IETF. He has been on the editorial board of a number of key journals in these areas, he is the chief editor of the bi-annual IEEE Communications network and service management series and in 2011 he received the Daniel Stokesbury award for "Distinguished technical contribution to the growth of the network management field".