SigMA: Signaling Framework for Decentralized Network Management Applications

Dario Valocchi[†], Daphne Tuncer[†], Marinos Charalambides[†], Mauro Femminella[*], Gianluca Reali[*], George Pavlou[†]
[†]Department of Electronic and Electrical Engineering,University College London, UK
[*]Department of Engineering, University of Perugia, IT

***Abstract*—The management of network infrastructures has become increasingly complex over time, which is mainly attributed to the introduction of new functionality to support emerging services and applications. To address this important issue, research efforts in the last few years focused on developing software-defined networking solutions. While initial work proposed centralized architectures, their scalability limitations have led researchers to investigate a distributed control plane. Controller placement algorithms and mechanisms for building a logically centralized network view are some examples of challenges addressed in this context. A critical issue that requires specific attention concerns the communication between distributed entities involved in decision-making processes. To this end, we propose SigMA, a signaling framework that supports communication between the different entities of a decentralized management and control system. We also define the communication primitives and interfaces involved in such a decentralized environment. The benefits of SigMA are illustrated through three realistic network resource management use cases with different communication requirements. Based on simulation, we demonstrate the flexibility and extensibility of our solution in satisfying these requirements, thus effectively supporting advanced decentralized decision-making processes.**

***Index Terms*—Signaling, Decentralized network management, Software-based networks.**

I. Introduction

Network resource management is traditionally performed by software operating in an offline fashion, which is executed by a centralized management system. The primary objective of management software is to configure the network resources such that their their usage is optimized over long periods of time. While resource provisioning is essential to enable the operation of the network under a wide range of conditions, it has limitations when it comes to satisfying the requirements of emerging applications and services with stringent needs in terms of quality of service, resilience and security guarantees.

Despite the various efforts invested for the last 20 years towards the realization of adaptive resource management (*e.g.,* active networking, autonomic networks, *etc.*), we still have not seen any wide deployment of standardized operational solutions. The recent initiatives on the development of Software-Defined Networking (SDN)-based approaches and the emergence of concepts such as Network Function Virtualization (NFV) have however given a new impulse to these issues. In particular, these technologies have led to a redefinition of abstraction models (both in terms of resources and functions), which are essential for the realization of flexible, dynamic and adaptive management functionality.

In our previous work [1], we developed a novel SDN-based management and control framework to support adaptive resource management. The framework follows a layered architecture compatible with the generic SDN model defined by the Open Networking Foundation (ONF), and relies on distributed planes to implement the management and control functionality. Management operations are executed by distributed Local Managers (LMs), which communicate their decisions to distributed Local Controllers (LCs). The latter are responsible for planning the sequence of actions to enforce for updating the network configuration parameters. A key challenge associated with the development of such a distributed solution concerns the communication between the various entities in the system so that management decisions can be computed and enforced, while satisfying the application requirements and without incurring significant signaling overhead and complexity.

In recent years, some proposals for communication models in distributed environments have been discussed in the literature *e.g.,* [2][3][4][5]. Existing approaches, however, do not apply well to adaptive resource management as they either focus on specific problems (*e.g.,* mechanisms to build a global network view [3]), lack the required functionality (*e.g.,* absence of synchronization support [6]), or are simply too complex to deploy, *e.g.,* [7]. To address these limitations, we propose SigMA, a new signaling approach to support communication between the different entities of our decentralized management and control framework. SigMA is an extension of our previous work [8] that focused on the communication between the distributed management entities. To support the interaction between the different components of the framework, other interfaces are however essential, especially between the LMs and the LCs that need to exchange messages regarding the new configurations computed by the management applications.

In this paper, we extend our previous work in [8] by considering all interfaces of our architecture and by demonstrating how the proposed signaling approach can be used to enable communication between all the entities. More specifically, the main contributions of this paper are as follows: i) we formalize the functionality of local controllers and define an abstraction model to represent management and control functions, ii) based on this model, we present the sequence of operations required for translating configuration decisions computed by resource management applications into enforceable actions, iii) we describe in detail all interfaces required to execute these operations and investigate in particular the communication between LMs and LCs, and iv) in addition to the previously considered use case on cache management, we show the benefits of the proposed framework based on two new use cases for security management and online traffic engineering.

Our results, which are based on simulation experiments, demonstrate the flexibility and extensibility of our solution in meeting the requirements of different types of management applications and in controlling a heterogeneous set of network resources. They also show that by introducing negligible network or bandwidth overhead, the proposed solution enables the deployment of decentralized and distributed management applications.

The remainder of this paper is organized as follows. Section II provides background information on our network resource management framework and the placement of management and control functionality. In Section III, we describe the main operations performed by the LMs and the LCs and elaborate on the role of the main communication interfaces. The signaling protocol is presented in detail in Section IV. In Section V, we provide some information about the three use cases considered in this work and evaluate their performance in Section VI. Related work is discussed in Section VII. Finally, conclusions are provided in Section VIII.

II. Background

In this section, we provide some background information on the SDN-based framework we developed in our previous work. We also briefly discuss how to distribute management and control functionality.

A. SDN-based Resource Management Framework

In our previous work [1], we developed a novel SDN-based network resource management and control framework to support both static and dynamic resource management applications in fixed backbone infrastructures. In the proposed framework, the network infrastructure is managed and controlled by a set of software-based Local Managers (LMs) and Local Controllers (LCs), forming distributed management and control planes, respectively. As depicted in Fig. 1, the framework follows a three-layer architecture designed based on the principles of the general SDN model defined by the ONF. The bottom layer represents the underlying network infrastructure. The middle layer implements distributed management and control functionality. Finally, the top layer represents the central management system. Interaction between the different components of the architecture is realized through a set of interfaces.

A key feature of this framework resides in its modular structure, which is represented by two levels of separation, *i.e.*, between management and control functionalities, on one hand, and between centralized and distributed management operations, on the other. Short to medium term management operations are performed by the LMs, which implement the logic of management applications (MAs), *e.g.,* traffic engineering, cache management *etc.* These are responsible for computing the configuration of the set of network resources under their supervision according to the objective of the applications which they implement. Configuration decisions taken by LMs are provided to the LCs, which define and plan the sequence of actions to be enforced for updating the network parameters. These actions are then translated to instructions sent to and executed by the relevant network devices. In contrast to the LMs, the centralized management system is responsible for longer term operations, for example those that pertain to the instantiation and life cycle of LMs and LCs. More specifically, this involves two components: the Local Manager Orchestrator (LMO) and the Local Controller Orchestrator (LCO)[1]. The LMO is responsible for performing high-level supervision of the LMs and MAs instantiated in the network, for instance to select the LMs which need to be involved in the decision-making process of a given application, or to determine how the decision-making process of a given MA is distributed.Example operations performed by the LCO include the computation of LC placement and the supervision of their state.

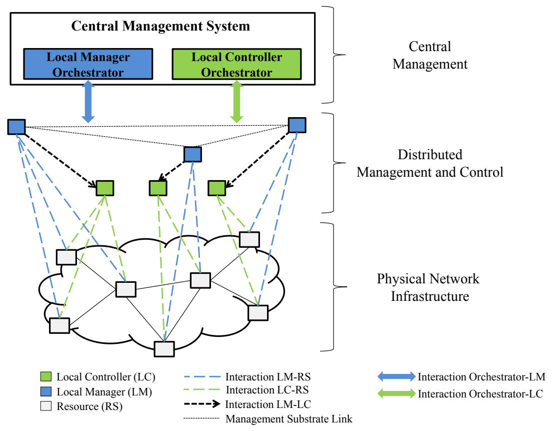
Central Management System
Local Manager Orchestrator
Local Controller Orchestrator
Central Management
Distributed Management and Control
Physical Network Infrastructure
Local Controller (LC)
Local Manager (LM)
Resource (RS)
Interaction LM-RS
Interaction LC-RS
Interaction LM-LC
Management Substrate Link
Interaction Orchestrator-LM
Interaction Orchestrator-LC

Fig. 1. Resource management and control framework proposed in [1].

While previous work considered management logic as an integral part of control functionality, e.g. [2][3], we believe that a clear distinction between the two provides several deployment benefits. This not only allows the two concerns to evolve independently, offering increased design choices and flexibility for the system vendors, it also simplifies the integration of new network applications, while maintaining interoperability.

B. Placement of Management and Control Functionality

The number of LMs and LCs to deploy, their location in the network, as well as the mapping between the two, depend on several factors, such as the physical infrastructure, the type of MAs to implement or the usage of the machines hosting the relevant software. For instance, the more frequent the interactions with the devices, the closer the management and control functionality is expected to be to the resource in order to minimize the reconfiguration latency [10]. The

[1]It is worth highlighting here that although we use the term orchestrator to define the two main modules of the centralized management plane, their functionality is different from the one foreseen in the ETSI NFV model of the Management and Orchestration (MANO) Framework [9].

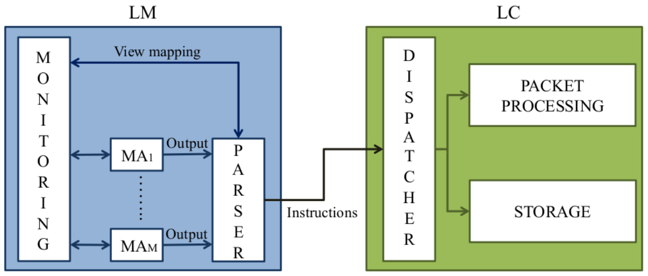

Fig. 2. The Local Manager (LM) and Local Controller (LC) components.

modular structure of our SDN-based resource management framework enables flexibility in the placement of the LMs and LCs, which is not bound to specific configurations. Based on the targeted objective(s) and resources, our framework allows LM and LC to be co-located on a single host or to reside on separate remote machines. In general, deciding on the placement of management and control functionality constitutes a trade-off between reducing the communication delay from the LMs/LCs to the network resources without significantly increasing the signaling overhead between the distributed entities [11][12][13].

III. Management and Control Planes

A. Components

1) Local Manager: As explained in Section II-A, LMs are responsible for performing short to medium term management operations. Each reconfiguration process involves at the LM level a set of three modules as depicted in Fig. 2.

The Monitoring Module executes functions related to network monitoring (*e.g.,* data collection, filtering, aggregation *etc.*) and enables each LM to create its own local network view. It is responsible for translating the collected low-level network statistics (*e.g.,* port counters, CPU cycles *etc.*) into a high-level abstracted view of the status of the network resources (*e.g.,* link utilization, server load *etc.*). To realize the translation, the association between application-level parameters and low-level resources is maintained in a mapping table which can be modified over time as new abstractions are identified and introduced.

The statistics collected by the Monitoring Module are passed as inputs to MAs instantiated at the LM level as different modules[2]. Each MA encompasses the logic to reconfigure the network resources according to some high-level objectives (*e.g.,* minimize delay, maximize cache hit ratio *etc.*). It maintains information tables and implements algorithms to decide on the configurations to apply. To assist the decision-making process, the requirements of the MA in terms of monitoring (*i.e.,* information needed and update mode) are passed to the Monitoring Module, which is responsible for converting these requirements into monitoring instructions. Each MA operates on its own abstract view of the resources (connectivity and configuration parameters), which can differ from application to application. The output of a MA is a set of application-level configurations.

[2]In practice, each LM can instantiate a different number of MAs.

The outputs are passed to the Parser Module, whose role is to cast application-level decisions into low-level configuration actions. The conversion is performed based on information retrieved from the mapping table maintained in the Monitoring Module. In contrast to the MAs, which are agnostic to the mechanisms implemented in the network to realize their objective, the Parser Module has knowledge about the low-level functions and protocols (*e.g.,* OpenFlow syntax). It acts as an intermediate between the management and the control logic by computing the value of the arguments used to control the configuration of these low-level mechanisms. Let's for instance consider the case of a traffic load-balancing application with the objective to determine the volume of traffic to send over the multiple paths from any source-destination pair of nodes in the network. In this example, the MA-level configuration represents the vector of splitting ratios that define the volume of traffic associated with each path. From the parser perspective, the configuration concerns instead the actual traffic splitting mechanism implemented in the underlying network switches (*e.g.,* set of forwarding rules). The processed decisions are finally sent in the form of instructions to the associated LC.

In a similar fashion to the framework, the modular structure of the LM provides multiple advantages. In particular, defining the MAs as separate modules allows each logic to be developed independently. In addition, by abstracting the reconfiguration logic, MA decisions are not bound to specific low level implementation, which provides more flexibility in the development of the applications. For instance, it is not necessary to know all the details of the traffic splitting solution used to design a new traffic load balancing application. To enable the integration of the applications into the system, the implementation needs to follow the abstraction model defined by the management framework, which can come in the form of a set of libraries or a Software Development Kit (SDK)[3]. From the parser point of view, the modularity allows the refinement of the application-level decisions based on the targeted resources without the need to implement the full functionality of the interfaces used to configure the relevant resources.

2) Local Controller: The LC is responsible for planning and executing the decisions taken by MAs based on the instructions received from the LM Parser Module that indicate the targeted type of resources. In this paper, we identify two types of resources: i) those concerned with packet processing, such as switches, firewall, routers *etc.*, and ii) the ones associated with data storage (*i.e.,* caches). As can be seen in Fig. 2, each resource type is represented by a separate module.

Messages received from the LM are first processed by the LC Dispatcher that determines to which resource module the messages should be redirected. Each module implements three main functions:

- Interpreter: the objective of the interpreter is to extract and translate the set of actions received from the LMs into low level reconfiguration instructions. These are ex-

[3]We refer the reader to [14][15] as examples of relevant initiatives on these issues.

pressed according to the specification of the protocol used to interact with the data plane resources (*e.g.,* OpenFlow, NETCONF *etc.*).

- Scheduler: the scheduler is concerned with scheduling the execution of the different reconfiguration instructions. In particular, actions sent pro-actively in anticipation to some network event (*e.g.,* link failure, congestion *etc.*) are stored in an internal data structure for future enforcement.
- Enforcer: the role of the enforcer is simply to execute the planned instructions.

As opposed to the LM, the LC implements all the interfaces to the network resources. In terms of functionality, however, it is a much lighter weight component which mainly acts as an actuator of the reconfiguration decisions.

B. Interfaces

1) Management Application to Management Application: To make a decision, the instance of a MA at a specific LM can either act independently based on local information or communicate with other MA modules located in remote LMs. The distributed instances of a particular MA can interact, for instance, to share information, to harmonize their decisions or to synchronize their reconfiguration cycle. To support the decision-making process of a specific MA, the set of LMs involved in the execution of that application are organized into a management substrate [5][10]. The substrate is a logical structure used to facilitate the exchange of information between decision-making entities for coordination purposes. Their number depends on the number of applications implemented in the distributed management plane (one substrate per application). Each substrate can follow different structures (*e.g.,* full-mesh, ring, hybrid). The choice of the structure is driven by different parameters related not only to the topology of the underlying infrastructure, but also to the constraints of the coordination and communication mechanisms supporting each application. 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. To communicate through the substrate, each MA module stores the list of its neighbors in the substrate, represented by the identifier of the corresponding LMs. The list, as well as application-specific information, is retrieved from the Local Manager Orchestrator module in the centralized management plane (Fig. 1) at the deployment phase.

2) Local Manager to Local Manager: The LMs in the distributed management and control plane provide a common execution environment for different applications. The LM-to-LM interface enables the communication between the MAs hosted locally and the MAs hosted on remote LMs, constituting as such an *east-west* interface[4]. To communicate with other instances, each LM maintains locally a translation table associating the identifier of the LMs to their IP address.

[4]It should be noted that in our framework the *east-west* interface is implemented where the management intelligence lies, *i.e.,* between the LMs, but in practice it can also be used to enable communication between LCs.

3) Local Manager to Local Controller: The interface between the LM and the LC is used to exchange the processed reconfiguration outputs which are encapsulated into messages that contain two main fields:

- Resource Type: this field indicates the resource type to which the configuration applies (*i.e.,* packet processing or storage). The Resource Type is retrieved by the Dispatcher Module of the LC which then redirects the message to the appropriate resource module.
- Instruction: this field contains two elements: (i) Targets: list of network resources (identified by their resource ID) to which the configurations should be applied, and (ii) Actions: list of reconfigurations to be applied to each resource with the arguments indicating the new parameter value.

To communicate with its peer LCs, each LM implements a translation table indicating the association of the identifier of the LCs to their IP address.

4) Local Manager/Local Controller to Underlying Resources: As explained in Section III-A1, the Monitoring Module in each LM is responsible for gathering information within the scope of the LM and making it available to local MA instances. The data collection is realized through a LM-to-RS (resource) interface connecting the LM directly to the network resources. Since the network information is primarily needed by the LM, the LM-to-RS interface allows to bypass the LC to avoid additional processing and delay. Configuration decisions taken by the management application are applied to the underlying resource by the LCs through a LC-to-RS interface. The technology and protocols used to implement this interface depend on the type of resource to configure, as well as on the resource vendor (*e.g.* OpenFlow switch, RESTful API, etc.)

5) Local Orchestrator to Distributed Plane: The interface between the Local Orchestrator modules in the centralized management plane and the LMs/LCs in the distributed management and control plane is used for the long term configuration of the LMs and the LCs. For each LM, the Local Management Orchestrator maintains its identifier, the address of the server hosting it, the list of managed resources under its responsibility (*i.e.,* switches, routers, caches), and the list of hosted MAs. In a similar fashion, the Local Controller Orchestrator maintains the list of LCs represented by their identifier, the address of the servers hosting them and the list of resources under their control.

This information is pre-computed and provided to the LMs and the LCs, respectively, during the deployment phase, and updated, if needed, during a long term reconfiguration cycle.

IV. Signaling Framework

To enable communication between the entities of the proposed management and control framework, a suitable signaling protocol is required. Three key aspects were taken into account when designing the new protocol: i) identify the minimum amount of information required in order to provide a signaling service to the entities of the management framework, ii) determine how to efficiently format this information and iii)

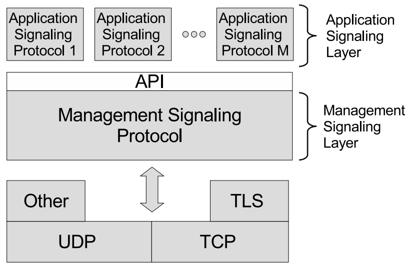
Application Signaling Protocol 1
Application Signaling Protocol 2
Application Signaling Protocol M
Application Signaling Layer
API
Management Signaling Protocol
Management Signaling Layer
Other
TLS
UDP
TCP

Fig. 3. Signaling architecture.

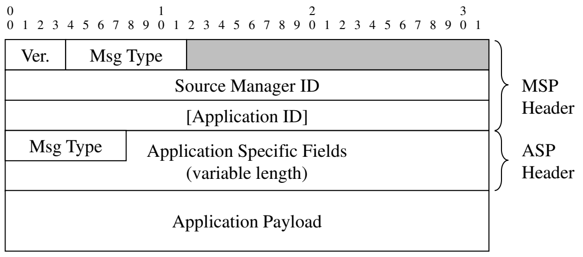

Fig. 4. MSP and ASP packet format.

define the internal statuses of the signaling nodes. This section presents our proposed solution, SigMA.

A. Design Principles

Different management applications can have different requirements in terms of how and when they need to share information. Due to this heterogeneity, a monolithic signaling protocol would not be able to cope with the needs of different applications. Furthermore, the type of management applications that could emerge in the future, as well as their signaling requirements cannot be easily foreseen. Another important aspect for the design of the signaling framework is the relationship between the management and control planes. Different types of applications would require to control different categories of network resources, which can be classified, for example, into SDN switches, routers, and network caches. Based on this classification, different controllers would be required to control and enforce decisions on different resource types. As such, the signaling framework should be flexible and allow the MAs to interact and communicate with different types of LCs, through the abstraction provided by LMs.

In addition, the signaling framework should also support the parallel evolution of the management and control planes. The definition of new features, or the introduction of new MAs, new categories of resources or new and more advanced LCs, should not require any modification of the signaling architecture. SigMA is designed to ensure both flexibility, in terms of enabling the communication between different types of applications and different types of controllers/resources, and extensibility to cope with the evolving nature of the management and control planes.

To achieve these objectives, the proposed architecture decouples the infrastructure signaling, necessary to deploy and coordinate the system entities, from the management signaling required by the MAs to execute their decision-making process. This decoupling leads to a two-layer architecture as depicted in Fig. 3: the Management Signaling Layer (MSL) deals with the infrastructure signaling, whereas the Application Signaling Layer (ASL) implements the signaling logic needed by the MA modules.

The main functions of the MSL are the following:

1) Exchange packets used by the system to deploy modules over the network and update their internal status (*e.g.*, configure LM, setup MAs, distribute substrate information, *etc.*).
2) Multiplex and demultiplex packets coming from and directed to the MAs.
3) Provide the communication interface between the MAs and the LCs.

The ASL has two main functions:

1) Exchange packets to share information needed by MA modules to make decisions.
2) Exchange packets to synchronize the different phases of the decision-making process of a given application.

We refer to the Management Signaling Protocol (MSP) as an implementation of the MSL, and to the Application Signaling Protocols (ASPs) as the implementations of different ASLs. The decoupling between the MSL and the ASL allows the MA developers to design their own information format and communication scheme, which relies on a common layer to provide message transport and addressing services. In addition, the MSL provides a common interface for the different MAs to push their decisions to the relevant type of LCs (*e.g.*, for SDN switches or caches) associated with the LM, regardless of the specific vendor implementations. The evolution of existing LCs, or even the introduction of new LCs and/or resource categories, will not affect the signaling architecture, and will only require the definition of new identifiers for the different LCs, and the development of a relevant MA to interact with/manage them. Another advantage of this architecture is that, in case of LM relocation, the decoupling between the MSL and the ASL allows the Local Management Orchestrator to inform the other LMs about the new address of the relocated LM. This alleviates the need to compute and distribute a new network-level substrate for all the MAs affected by the relocation, which provides gains in terms of computing time and network bandwidth.

In addition to intra-MAs communication, the MSL addressing service allows different MAs, not necessarily deployed on the same location, to communicate. This enables developers to design and build more complex management systems, splitting complex management tasks in smaller and simpler jobs to be handled by small interacting MAs, which share information and synchronized decisions through the relevant substrates.

B. Management Signaling Protocol Operations

Instead of designing a new IP based transport protocol, we designed the MSP to rely on existing transport and session protocols, which provide the communication primitives, such as datagram messaging (UDP), reliable sessions (TCP), or secure and reliable sessions (TLS over TCP).

The ASP maintains information about the substrate of the relevant MA, represented by a list of LM identifiers. The interface between the ASP and the MSP allows the ASP to specify a list of LMs, which are the signaling destinations, its payload, and the signaling requirements (*i.e.*, in terms of reliability, security, *etc.*). The MSP adds its header to the payload, uses its internal table to resolve the IP addresses of the destinations and chooses, among the available transport services, the one matching the ASP requirements. It can then send the packet to the selected destinations. Fig. 4 shows the packet format of the proposed signaling framework. The header of the MSP is designed to be as general as possible in order to cope with the different requirements in terms of flexibility and extensibility, as described in Section IV-A. The Message Type field in the MSP header is used to identify packets carrying ASP payloads, and packets coming from or directed to LCs, or to the Local Management Orchestrator. The MSP header also contains the Source Manager Id field that is fetched with the payload to the ASP when a message is received. As such, the ASP does not need to execute a reverse lookup on the MSP table in order to identify the source of the message.

In contrast, the ASP packet format strongly depends on the specific ASP implementation. A Message Type field should be available to allow the finite state machine of the ASP to identify the format of the carried MA payload. For example, the ASP header of a time-driven MA could provide synchronization fields, such as a configuration cycle identifier, an iteration identifier or a timestamp, and the ASP header of an event-driven application could include the triggering event identifier.

In the set of functions implemented by the MSL, packets multiplexing and demultiplexing are fundamental tasks. To offer these services, an IP resolution table, mapping LM identifiers to their corresponding IP address, needs to be configured during the LM deployment and subsequently maintained. Given that the LM deployment is centrally managed by the LMO, a centralized approach is also followed by the MSP to maintain the IP resolution table. More specifically, the table is configured by the LMO during the deployment phase in each LM, using the standard MSP packet header (Msg. Type = 1, LM-ID = 0, Application ID = 0) that encapsulates a list of ordered pairs (LM_ID_i, IP_i). With this signaling session, each LM is able to configure its own address table with the address of the LMO and other LMs. Once the deployment phase is completed, the LMO relies on a pull strategy to periodically perform an *health-check* on each LM. In that case, an empty MSP packet with Msg. Type set to 3, LM-ID and App. Id set to 0, is sent by the LMO to the LM. If the MSP is running, the LM replies with the same message type, with a non-zero LM-ID, and a list of Application-IDs representing the MA hosted locally. Finally, in case of failures or LM migration, the LMO can broadcast the changes of the overall LM-to-IP mapping using the message format of the initial configuration phase.

V. Use Case Description

To illustrate the role and benefits of SigMA, we investigate how this can be used to enable distributed instances of management applications as defined in Section III-A1 to communicate in the context of three use cases, which are described in this section. As summarized in Table I, these applications have different characteristics in terms of the traffic they generate, the periodicity and frequency of execution.

TABLE I
Application Characteristics.

Application	**Load**	**Periodicity**	**Impact**
Cache Management	Heavy	Fixed	Infrequent
Security management	Low	Event-driven	Infrequent
Online TE	Variable	Fixed	Frequent

A. Cache Management

The first use case concerns a cache management application that was originally developed in [16] and graphically represented in Fig. 5. In this ISP-operated caching scenario each network node is associated with caching capabilities and is used to locally store a set of content items. The configuration of each cache is computed based on the logic of the management application implemented by a set of LMs, which coordinate their decisions through the management substrate defined for this application. The objective is to determine which content items to cache, and where, based on content characteristics such as the popularity and origin (geographical) of requests.

The cache reconfiguration algorithm is executed periodically by cache managers, once a quasi-synchronous timer expires. The objective of the timer is to introduce a degree of desynchronization between the distributed managers in order to illustrate the ability of our framework to support a synchronization mechanism. In that case, the ASP deals with the synchronization issue by sending a set of MA-dependent signaling messages. The decision process is composed of two phases. The first phase is iterative and involves the use of the ASP. At the start of this phase, cache managers exchange content popularity information, as perceived from their local view, through the ASP. The collected information is subsequently aggregated by each manager separately to build a data structure representing the global view of content popularity. The next part of this phase involves an iterative process to decide on which content items to store at each of the available caching locations. At each iteration, every manager selects a variable number of items to cache locally based on, (a) information extracted from the global popularity structure, and (b) the cache status of other managers in the substrate. Given that (b) needs to be updated at each iteration, management applications use the ASP to share their cache status once a new placement has been computed. The first phase ends in an asynchronous fashion and a final cache status exchange is therefore required in order to synchronize the global content placement view between the different cache managers. In addition to communicating information relevant to the first phase of the algorithm, the ASP is also used to exchange messages to keep the cache managers synchronized with the current step of the algorithm. A detailed description of the algorithm can be found in [17]. The second phase of the

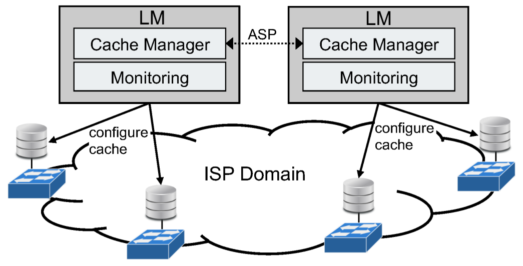

Fig. 5. Cache management use case.

algorithm is carried out by each manager independently with the objective of filling up any remaining local cache capacity and, as such, does not involve the ASP.

B. Security Management

The second use case concerns security in mobile network operator environments, where malicious user devices can flood a server(s) within the network with fake requests. This can deteriorate the performance of running services or block them altogether due to server overloading and/or excessive network traffic. In contrast to the previous use case in which the caching algorithm was executed at fixed periodic intervals, reconfigurations in this case are triggered by security events.

We consider a set of distributed security management applications executing in LMs that have at least one edge node under their scope of responsibility, *i.e.,* from where malicious traffic emanates, as shown in Fig. 6. These applications receive monitoring information, *e.g.,* user request statistics, and perform initial analysis for detecting abnormal behavior. Upon such an event, and alarm procedure is triggered by which (a) other application instances are pro-actively informed using the ASP, (b) a server location is determined where traffic can be further analyzed, *e.g.,* a deep packet inspection (DPI) engine, and (c) relevant LCs are instructed to install rules for redirecting potentially malicious requests to the DPI engine. The latter responds to the LM that initiated the process with the attack profile and the security application communicates this information to other application instances through the ASP. As a response, security applications decide on the remedy actions, for example dropping traffic from specific sources or ports, which are enforced by the relevant LCs. Taking proactive remedy actions at remote locations is particularly useful in mobile environments since malicious user devices can move and connect to different base stations over time, but also because an attack can propagate among users.

C. Online Traffic Engineering

Resource reconfigurations in the context of the previous two use cases are not frequent. In order to evaluate the proposed framework under a more dynamic environment, we consider an online traffic engineering (TE) application in which reconfiguration decisions are executed in short timescales, for example every few seconds. This involves the adaptation of traffic splitting ratios for achieving traffic engineering objectives, such as load balancing. While some prior research proposed that splitting decisions are enforced at ingress routers, *e.g.,*

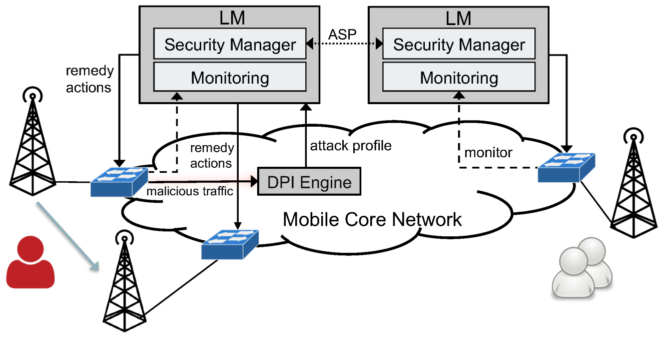

Fig. 6. Security management use case.

[18], other approaches proposed that all nodes in the network are responsible for dynamically splitting the traffic between the available next hops, *e.g.,* [19] [20]. In this use case we adopt the latter and investigate the ability of the signaling framework to effectively support a set of distributed online TE applications that require link utilization updates at a high frequency. Although in some distributed scenarios information would be aggregated, or only alerts would be communicated in an effort to reduce the management overhead, we consider the worst case to test scalability: at each reconfiguration interval, application instances exchange the local raw measurements so that they can all build a global view of the network state.

VI. Evaluation

In this section, we present the results of the experiments we performed to demonstrate the benefits of the proposed signaling framework based on the three use cases described in Section V. For each use case, we designed and implemented the relevant management application and its associated ASP. For all these use cases, results were obtained based on custom simulators [5]. In all cases, we used the Deltacom topology [21] with 92 nodes for layer 3 routing. We assume that each link in the network has an available bandwidth of 10 Gbit/s, which is a conservative capacity value in real networks (*e.g.*, [22]) and 5 ms of network latency per hop [23][22]. The placement of LMs and LCs is determined based on the algorithms presented in [13]. It is provided as an input and is used to compute the relevant IP distances between pairs of entities.

A. Cache Management

As explained in Section V-A, the objective of the cache management application is to determine which content items to cache and where, based on content characteristics such as popularity and geographical origin of requests. In this subsection, we first describe the experiment settings and then present the obtained performance results.

1) Experiment Settings: We generated user demand as follows. The number of requests per content, for each manager and for each reconfiguration cycle, is provided as an input to our simulator. To determine the number of requests per content, we generate one hour of random requests based on

[5]Code available at http://clayfour.ee.ucl.ac.uk/sigma/SigmaSource.zip

a Poisson distribution with inter-arrival time $\lambda = 5000$ req/h (based on the hourly request pattern of the video on demand (VoD) trace presented in [24]) and a Zipf distribution with skew factor $\alpha = 1.2$ (based on the characteristics of the VoD dataset used in [25]). For the geographical distribution of interests (*i.e.*, number of distinct locations from which a content is requested), we use the model proposed in [16] with $\beta = 0.8$. In this case, the predominance of popular content items can be preserved without strongly discriminating less popular ones. The caching space available in the network is evenly divided between the deployed LMs and, as such each LM is responsible of managing the same amount of caching space.

The performance has been evaluated both in terms of the time consumed by the MA during one reconfiguration cycle, and the traffic generated during the decision-making phase. The **Network Time** is defined as the time used by each instance of the MA to send signaling messages to its peers in the substrate. This time is divided into two components; the transfer time and the network delay as follows:

- The **Network Delay** is measured assuming that each IP hop in the topology incurs a 5 ms delay (based on the values reported in [23]).
- The **Transfer Time** is computed by dividing the size of the packet with the link bandwidth.

The network time introduced by the signaling is measured and accumulated by each manager during each reconfiguration cycle.

As for the traffic generated, the **Network Traffic** is computed by summing up the size of each packet multiplied by the length of the path between its source and its destination, measured in terms of IP hops. This measurement assumes that a TCP socket is used by the MSP to transport the messages.

Each MA module is represented by an object that implements the cache reconfiguration algorithm and the ASP described in Section V-A. The simulator runs the MA algorithms and signaling routines in real time, which enables the direct collection of reliable time measurements. At the end of each reconfiguration cycle, the average value and the standard deviation for the network time are computed based on the values measured by each node, while the network traffic metric is obtained by summing up the contribution of each node. For each test we compute the average value along with the standard deviation for all the relevant metrics, collected for 10 reconfiguration cycles.

2) Results: We collected values for the described metrics for different configurations of three main parameters of the system: the content catalogue size, the number of managers (*i.e.*, caches), and the caching space available at each caching location. We consider a content catalogue with a number of items ranging from 10^4 to 10^5 based on the catalogue sizes reported in [26]. The number of managers, and their location in the topology, is computed using the placement algorithm proposed in [1], and it ranges from 4 to 82. As for the available caching space per location, we also used [26] and selected four values for the ratio between the total caching space in the network and the catalogue size (5%, 10%, 15% and 20%), but due to space limitations, we present results only with the two extreme cases (5% and 20%)[6]. For each ratio, we compute the available space per location by dividing the resulting total caching space by the number of managers.

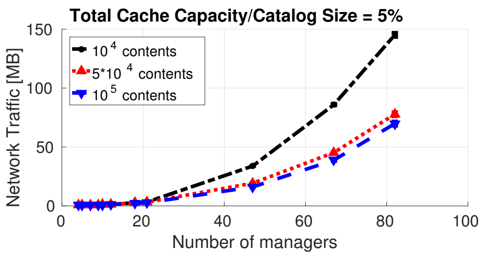

Fig. 7. Average Network Traffic with 95% confidence interval for a ratio between the total caching space and the catalogue size of 5%, for different values of the catalogue size, as a function of the number of managers.

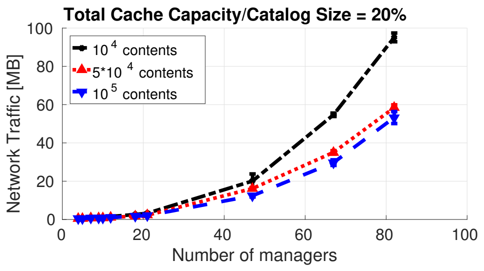

Fig. 8. Average Network Traffic with 95% confidence interval for a ratio between the total caching space and the catalogue size of 20%, for different values of the catalogue size, as a function of the number of managers.

Figures 7 and 8 show the results for the network traffic, and Figures 9 and 10 show the results obtained for the network time metric. The evaluation demonstrates that the proposed signaling framework can support the complexity of the communication scheme of the Cache Management application. This relies on a N-to-N communication scheme, leading to a complexity in the order of $O(N^2)$, with N being the number of managers. The results show that the quadratic relationship between the number of managers and the generated traffic is satisfied: no further overhead or complexity is incurred by the signaling system. In addition, an increase of the caching space at each manager leads to an increase in the performance in terms of network metrics. Increasing the caching space leads to a decrease in the number of iterations needed to complete a reconfiguration cycle, thus resulting to a decrease in the number of sharing tasks, and consequently to performance improvement. This can be explained by the cache reconfiguration algorithm, which relies on a parameter p to control the number of content items to consider for caching at each iteration of the first phase. The value of p depends on the total available caching space in the network and an increase in its value means that more items can be cached at each iteration. This leads to a decrease in the number of iterations needed to complete the first phase.

An interesting result is the performance improvement in terms of network metrics when the number of content items in the catalogue increases. When a lower number of items is

[6]It is worth noting that similar results were obtained with the two other ratio values.

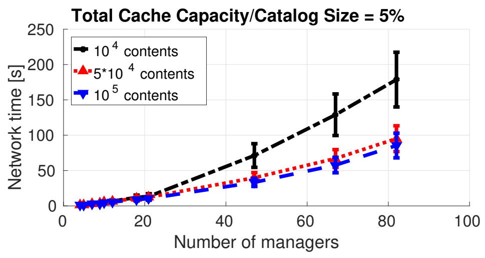

Fig. 9. Average Network Time with 95% confidence interval for a ratio between the total caching space and the catalogue size of 5%, for different values of the catalogue size, as a function of the number of managers.

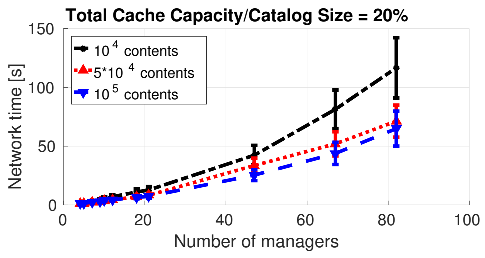

Fig. 10. Average Network Time with 95% confidence interval for a ratio between the total caching space and the catalogue size of 20%, for different values of the catalogue size, as a function of the number of managers.

considered, we observe that more signaling traffic is generated, which is due to how the local popularity is computed. As explained in Section VI-A1, we generate one hour of requests as a Poisson process. The total number of requests received during that period is then distributed between the contents in the catalogue according to the Zipf's Law, in order to compute the number of requests per content. The function proposed in [16] is used to compute the geographical distribution of the demand. The total number of requests per content is then divided by the number of geographical locations and assigned to a random set of managers, accordingly, if and only if the number of requests for that content at a specific location is greater than zero. Given that the total number of requests for each configuration cycle follows a stationary stochastic model and the number of requests per content is driven by Zipf's Law, an increase in the total number of content items leads to a decrease in the total number of requests per content. This causes more items to have zero requests at certain geographical locations. Therefore, the length of some of the messages exchanged in the first phase of the algorithm decreases, thus also diminishing the Network Time and the Network Traffic. In order to evaluate the effect of the function used to compute the number of requests per content, we also executed tests replacing the Zipf's law with a uniform distribution. This forces each content to be requested at least once and from at least one location. Since this is not the focus of this paper, we do not show the results here, but it is worth mentioning that, in this case, the value of network related metrics increases with the size of the catalogue.

Finally, we are interested in the effects of the degree of distribution of the management instances on the performance of the Cache Management application in terms of computational resources (CPU). Given that this comes with an increase in terms of traffic generated by the signaling framework, it is important to assess whether increasing the degree of distribution can result in better computational efficiency. Figure 11 shows the average CPU time spent by each MA instance to compute the cache reconfiguration algorithm for the next configuration cycle with a 95% confidence interval, as the number of manager increases and for different values of the catalogue size. Increasing the number of MA instances from 4 to 47 decreases the average CPU time of the cache reconfiguration algorithm by 78%, while incurring less than 15 MB of traffic on the whole network every hour. This amount of traffic can be considered negligible compared to the capacity available on modern networks. At the same time, the improvement in terms of CPU time could be important in a scenario where many MAs share computational resources through virtualization, and the selection of the appropriate number of LMs to be deployed in the network could be a valuable instrument to optimize their use.

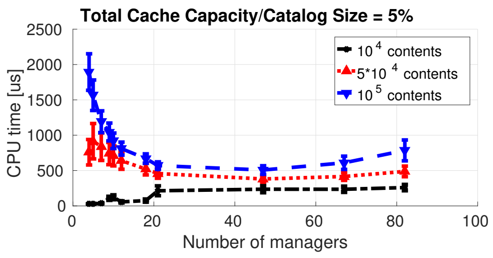

Fig. 11. Average Cache Management CPU time with 95% confidence interval for a ratio between the total caching space and the catalogue size of 5%, for different values of the catalogue size, as a function of the number of managers.

B. *Security Management*

The security management application aims at supporting DDoS defense mechanisms, by enabling alarms and attack profile distribution among management entities located at the edge nodes of a network. In this section we describe the models and settings we used for our experiments and analyze the results obtained with our simulations.

1) Experiment Settings: We generated a traffic trace inspired by the real Distributed Denial of Service (DDoS) attack that happened in September 2015 and for which details were reported in [27]. In our attack scenario, pedestrian users are accessing web pages which contain advertisements through the browser of their smartphones. The advertisements are generated by an advertisement system and selected using an auction mechanism. We assume that the auction mechanism has been compromised by an attacker through the insertion of a malicious advertisement in the advertisements pool. The malicious item instructs the infected browser to generate a high volume of HTTP GET requests towards a specific target server [27] located inside the core network. As the attacker succeeds in compromising the auction system, the malevolent advertisement is selected to be included in an increasing number of web pages accessed by the mobile users, resulting in an increasing number of mobile devices flooding the target with HTTP requests.

We considered a set of 10,000 mobile users scattered around a square area and modeled their mobility pattern based on the approach described in [28]. The area is covered by a cellular network composed by 74 hexagonal cells, each connected to an edge access switch. The set of edge switches represents 80% of the Deltacom topology. The edge switches are themselves interconnected by a backbone of 18 core switches representing 20% of the topology. Each cell has a radius of 300 meters [29]. To model the attack spreading behavior, we assume that each user device can have two different statuses: a normal status in which a low number of requests is generated each second (*e.g.,* 1 request per second per user), and a compromised status in which the number of requests per second sent from each device is orders of magnitude greater (*e.g.,* 100 requests per second). The percentage of devices in the compromised status increases over time following a first order step response $N_{compr} = 1 - e^{\frac{t}{\tau}}$, that is uniquely defined by the time constant τ. As the value of τ decreases, the attack becomes faster. With this assumption, we can define the **attack speed** as the rise times (10% to 90%) of the aforementioned step response function. Moreover, users move around the area causing handovers from cell to cell so that the malevolent traffic moves and enters the network from different cells over time. We use the model proposed in [28] to evaluate the handover rate, allowing us to generate the ingress traffic coming from each cell (i.e. from each edge node of the Deltacom topology).

Based on this model, we generated one hour of traffic traces in which the malicious advertisement is fetched by users for one third of the time before it is stopped, and we evaluated the performance of the security management application using the following metrics:

- The **Network Traffic** generated by the signaling framework during the whole time window (*i.e.,* one hour). The measurements assume that a TCP socket is used by the MSP to transport messages.
- The **Alarm Propagation Time**, defined as the time needed to execute an alarm procedure, from the attack detection to the enforcement of traffic steering to the DPI engine.
- The **Countermeasure Propagation Time**, that is the time needed to distribute the Attack profile computed by the DPI engine.

2) Results: We evaluated these metrics using the same LM deployment described in Section VI-A1, with the number of local managers ranging from 4 to 82. As explained in Section V-B, the Security Management application runs only on LMs which control at least one edge node, so that the actual number of MA instances in the network ranges from 4 to 64. In this paper, we are interested in the performance of the proposed signaling framework (*i.e.,* how it can be used to support management applications with different requirements) rather than in the management substrate (defining the mode of communication between MA instances [5]). As such, we show the results as the total number of deployed LMs varies. In particular, this demonstrates how the framework can enable communication between the relevant set of LMs. In addition, to assess the ability of SigMA to respond to time constraints of different magnitudes, we considered different values of the attack speed, ranging from 44 seconds to almost 10 minutes.

Figure 12 shows the overall network traffic generated by the Security MA during the attack through both the LM-to-LM and LM-to-LC interfaces. As can be observed, a negligible volume of traffic is incurred by the signaling framework - 20 MByte only over the whole network in the worst case. Moreover, given that the management application instances only run where they are needed, the volume of generated traffic increases almost linearly with the number of LMs. Finally the results also show that the speed of the attack has almost no influence on the volume of signaling traffic incurred in the network.

Given the sensitivity of the Security Management application to time constraints (*i.e.,* event-driven), the performance obtained in terms of time-related metrics is crucial. Figure 13 shows the average alarm propagation time, with the relevant 95% confidence intervals. These values are obtained by averaging the measured alarm propagation times between all the alarms issued by each MA instance during the attack and between all the MA instances. The results demonstrate that even in the worst case, where the attack is moving from 10% to 90% of its power in 44 seconds and the access network is controlled by 83 LMs, SigMA enables the security MA to propagate the alarm and to instrument the necessary countermeasures in just 4 seconds. It is also important to highlight that the responsiveness of the proposed approach is independent of the attack speed. This shows that the proposed signaling solution can be used in scenarios with different time requirements. These observations are further confirmed by the results obtained for the countermeasures propagation time, which show the same performance in terms of responsiveness and robustness.

As a final observation, we stress that, although the actual logic of the security application is out of the scope of this work, the increased overhead produced by SigMA when the number of LMs increases is negligible in comparison to the resources available in modern networks. As such, it represents an affordable trade-off when compared to the advantages a distributed management application can bring. Recent literature has highlighted examples of how SDN [30], [31] and NFV [32] technologies can be effectively used to mitigate the effects of DDoS attacks [33], [34]. A representative example of these advantages in the context of a DDoS mitigation system can be found in [35], where a distributed set of classifiers, traffic throttles and packet markers is used, together with an alarm propagation system, to react to DDoS in a collaborative fashion. In this example the level of distribution of the entities positively affects the ability of the overall system to filter different attack types, thus balancing the cost of signaling among a larger number of agents. Another example can be found in [36], where the authors show that, in order to cope with large scale DDoS attacks, most filtering techniques require cooperation.

C. Online Traffic Engineering

In this subsection, we focus on the online traffic engineering application. As explained in Section V-C, to test scalability, we consider a worst case scenario where at each reconfiguration interval, application instances exchange their local raw

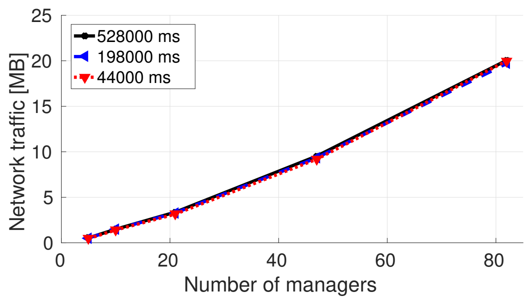

Fig. 12. Total network traffic for different values of the attack speed, as a function of the number of managers.

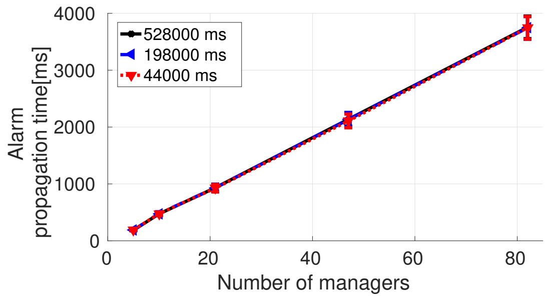

Fig. 13. Average Alarm propagation time with 95% confidence interval for different values of the attack speed, as a function of the number of managers.

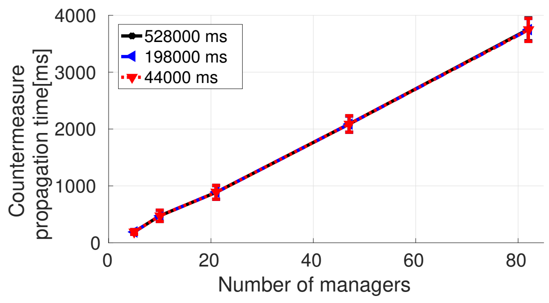

Fig. 14. Average Countermeasure propagation time with 95% confidence interval for different values of the attack speed, as a function of the number of managers.

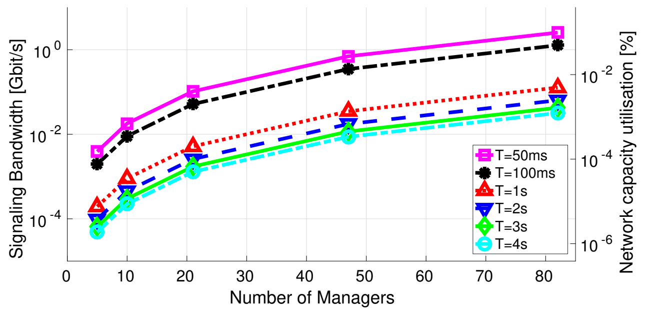

Fig. 15. Signaling bandwidth (left axis) and Network capacity utilisation (right axis) as functions of the number of managers, for different value of the monitoring interval

measurements so that they can all build a global view of the network state.

1) Experiment Settings: In this use case, we evaluated the overall **Signaling Bandwidth** needed by the MA instances to exchange link utilization updates as the number of deployed LMs varies from 4 to 82. The required bandwidth was evaluated for different values of the signaling interval, which is defined as the time each MA instance waits before sending its peers an update of the local link utilization. We let this parameter vary from the extreme case of 50 milliseconds to the more relaxed scenario where an update is sent every 4 seconds. To get an estimation of the network capacity used by signaling, we also compute the **Network Capacity Utilization** as the ratio between the signaling bandwidth to the total available bandwidth in the network. In a similar fashion to the two previous use cases, a TCP socket is used by the MSP to transport messages.

2) Results: Figure 15 shows the results obtained in terms of signaling bandwidth and network capacity utilization on a logarithmic scale. The left y-axis presents the value of the signaling bandwidth in Gbit/s while the right y-axis indicates the percentage of the available bandwidth used by SigMA in each experiment setting (*i.e.,* for different number of LMs). As can be observed, the required bandwidth increases quadratically as the number of deployed MA instances increases. In a similar fashion to the Cache Management application (Section VI-A2), this behavior can be explained by the scheme of communication used by the Online Traffic Engineering MA. More specifically, after each signaling interval, a global view of the resource utilization is built at each MA instance location. To build this global view, each MA instance shares with all its peers, the status of the links under the responsibility of its associated LMs. This causes the exchange of a quadratic number of messages and, as such, the footprint profile depicted in Figure 15. The results demonstrate that SigMA is able to support the requirements of the MA without introducing further complexity or scalability limitations in terms of generated traffic. Finally, the results obtained in terms of network capacity utilization show that in the worst case where 82 managers are deployed, 0.0988% and 0.0012% of the bandwidth globally available on the network is consumed with a signaling interval of 50ms and 4s, respectively. This shows that SigMA allows to increase the level of distribution of TE instances, which is required to support online decisions (*e.g.,* [19] [20]), with negligible effects on the bandwidth usage of the operated network.

VII. Related Work

It is common in networking to use the terms *northbound* and *southbound* when defining the interfaces of communication between the different components of a system. In the recent years, the notion of *northbound* and *southbound* interfaces has become central in the literature related to SDN where the SDN controller is presented as the focal point from which communication between a low-level data plane on one hand and a high-level user application plane on the other is defined. An example of northbound interface design based on an extensible REST API is presented in [37], where the use of REST is justified by the rapid evolution of SDN northbound APIs. In [38], the authors present a new architecture for SDN network management called Recursive InterNetwork Architecture (RINA) and propose to replace the northbound and southbound APIs with a unified and recursive RINA API.

This provides a high level interface for both administrators and users and allows management (and user) applications to be programmed recursively over different scopes at different levels. A review of existing network operating systems for SDN and relevant interfaces is presented in [39].

While early SDN solutions rely on a physically centralized control infrastructure, more recent research efforts have been investigated distributed control plane approaches (*e.g.*,[40][2][3]), introducing the notion of *east-westbound* interface to refer to as the communication between peer controllers. Different communication models have been considered in the literature. In [2], Yeganeh et al. propose a hierarchical approach by which communication between control entities is only permitted vertically between a root controller and a set of distributed local controllers. In contrast, a horizontal communication design based on a pub/sub system is used by Tootoonchian et al. in [40] to achieve the synchronization of network-wide view between distributed SDN controllers. A peer-to-peer approach was also considered in [3] where the authors focus on the design of a new controller architecture adapted to distributed settings and that integrates a messaging component to enable inter-controller communication. The development of *east-west* interfaces is in particular essential to support the exchange of information between multiple SDN domains. A protocol to support such communication is SDNi [41] that is designed to facilitate coordination between SDN controllers and the exchange of control information across multiple SDN domains. In [42], the authors present a signaling approach for distributed SDN controllers to exchange information regarding the routing rules and flow processing actions that need to be executed and propose a model to represent this information. In [11], Levin et al. focus on the problem of interaction between distributed controllers from the perspective of the impact of distribution on the performance of management applications. In particular, two trade-offs which should be taken into account when designing applications for distributed control planes are discussed: between performance optimality and state distribution overhead, and, between complexity of the application logic and robustness to inconsistency. The primary objective of most of the communication mechanisms proposed in the SDN literature is to enable distributed controllers to build a global network view. While this is also one of the objectives of the signaling framework proposed in this paper, our solution provides additional functionality for the purpose of coordinating the decisions of distributed management entities.

In general, the implementation of any distributed system come with a set of traditional challenges such as the design of communication and interaction models between distributed entities [4], synchronization issues [43] *etc.*. In the context of distributed network management application, efforts have focused on various issues such as the definition of the interaction type [44], the orchestration of distributed decisions [45], or the development of frameworks and protocols for the exchange of information between distributed decision points [46][5][47]. In this paper, we also design a communication protocol for distributed management entities. However, in contrast to previous work which mainly focused on specific use cases, we propose a general and extensible signaling framework that can be used to support any type of application.

The issue of a generic signaling framework, flexible enough to support virtually every kind of signaling application, has also been addressed by the IETF in [48], and further extended in [7] as NSIS. However, in spite of its generality, the complexity of the framework prevented it from becoming widely used. NSIS was initially designed to provide support for quality of service (similarly to RSVP) and to go further by providing an extensible mechanism for peer-to-peer oriented signaling applications. To support both these approaches, NSIS includes functions like peer discovery, peer-to-peer session control, and packet interception which introduce overhead and complicate the design of the signaling logic for applications. A generic messaging system (RELOAD) able to support peering messaging and to provide a messaging substrate service to overlaying applications has been proposed in [6]. Although some similarities with our approach exist, the main focus of RELOAD is on resource location and on creating and maintaining a distributed storage overlay. As such, it cannot be used to synchronize distributed decision-making algorithms.

VIII. Discussion and Conclusions

In this paper we significantly extend our previous work in [8] and present the design and in-depth performance evaluation of a novel signaling framework that enables communication in a distributed management system. In particular, compared to our previous work, we further elaborated on and formalized the operation involving the LC component, providing abstraction models and translation functionality to implement the relevant interface toward the LM component and toward resources. It is worth to mention that the proposed abstraction focuses on packet processing and storage as two illustrative examples of resources types. This does not preclude the introduction of other types of resources, which would apply to other environments and that would be simply represented by new sub-modules. Furthermore, we evaluated the performance of our solution on a wider spectrum of use cases, exploiting the characteristics of SigMA to meet different application requirements. Some key observations can be drawn from the results obtained for each of the three use cases.

The cache management application can be regarded as an extreme case for the signaling footprint, since it requires each distributed application instance to build a global view of both the input condition (*i.e.,* the content demand) and the system status (*i.e.* cache status during the cache reconfiguration algorithm). In addition, the synchronization of the algorithmic steps is required throughout the decision-making process. On one hand, these requirements directly influence the footprint generated by the signaling framework during the execution of the application, *i.e.,* it quadratically grows with the number of Cache Management instances. On the other hand, however, the distribution of the management entities provides non-negligible benefits in optimizing the decision-making process in terms of computational resources. As shown in the evaluation, the cost of distribution in terms of traffic is very limited. In the extreme case of highly distributed deployments, the volume of traffic generated by the content placement application over a time window of one hour is just 150 MB. We believe that this represents a reasonable trade-off with respect to the aforementioned benefits.

In the security management application a different set of requirements has been tested. Although infrequent, the dissemination of alarms and countermeasures to security threats should be fast and reliable to enable the distributed system to react in a coherent way to different attack aggressiveness but also to cope with the mobility of the attackers. Despite the simplicity of the logic of the security management application considered in this paper, the results show that in an event-driven, time-constrained scenario, SigMA is able to distribute the needed information within the expected time, introducing negligible traffic and without being itself affected by the aggressiveness of the attack. It should be noted here that in our scenario, possible issues associated with network congestion are not taken into consideration. In our opinion, these are not deciding factors in our context. First, most DDoS attacks aim at overloading the server capacity rather than the network capacity itself. In addition, our design is based on off-band signaling, which means that the signaling messages constitute a separate flow of traffic (*e.g.,* in terms of addressing points, protocols *etc.*). It is also worth mentioning that, in the worst case, the signaling protocol generates only 20 MB of traffic throughout the network in a time span of 4 seconds. In general, the obtained results (in terms of responsiveness and independence of the attack speed) provide evidence of the effectiveness of the design.

Finally, experimentation with the online traffic engineering MA shows the performance of our solution in terms of scalability when a smaller time scale is considered and the distribution of link status information becomes very frequent. The proposed solution supports this frequent exchange, scaling both as the number of distributed entities increases and the required refresh interval becomes smaller. We highlight that these results could be used as an instrument for designing the level of MA distribution; the proposed ASP is generic enough to provide an estimation tool for the bandwidth consumption of any application exchanging key-value information, in scenarios with different responsiveness requirements.

In conclusion, in this paper we show that the proposed signaling framework provides both flexibility and extensibility, enabling different types of management applications to interact and control a heterogeneous set of network resources. As evidenced by the evaluation results based on three different use cases, SigMA does not introduce scalability limitations or significant complexity: in all the presented use cases, the evolution of the network related metrics with the number of managers follows the expected trends while satisfying the constraints imposed by the relevant application. We believe that these results are important as they demonstrate that the proposed signaling solution can support advanced distributed management algorithms. In addition, due to its modular design, it also allows the management and control planes to evolve without needing to modify the signaling stack. Performance enhancements could also be achieved by further exploiting the decentralized nature of the proposed management framework. An example is to rely on a gossip-based ASP to spread information in an epidemic fashion, tolerating partial knowledge. Furthermore, the scope of peering between managers can be limited by partitioning the substrate into clusters, or introducing optimized substrate structure, so as to further adapt signaling mechanisms to the application requirements.

Acknowledgment

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.

References

[1] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, “Adaptive Resource Management and Control in Software Defined Networks,” *Network and Service Management, IEEE Transactions on*, vol. 12, no. 1, pp. 18–33, March 2015.

[2] S. Hassas Yeganeh and Y. Ganjali, “Kandoo: A framework for efficient and scalable offloading of control applications,” in *Proc. of HotSDN’12*, 2012, pp. 19–24.

[3] K. Phemius, M. Bouet, and J. Leguay, “Disco: Distributed multi-domain sdn controllers,” in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–4.

[4] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, “A Clean Slate 4D Approach to Network Control and Management,” *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 41–54, Oct. 2005.

[5] D. Tuncer, M. Charalambides, H. El-Ezhabi, and G. Pavlou, “A Hybrid Management Substrate Structure for Adaptive Network Resource Management,” in *Proc. of ManFI’14*, Krakow, Poland, May 2014, pp. 1–7.

[6] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, “REsource LOcation And Discovery (RELOAD) Base Protocol,” RFC 6940 (Proposed Standard), Tech. Rep. 6940, Jan. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc6940.txt

[7] M. Femminella, R. Francescangeli, G. Reali, and H. Schulzrinne, “Gossip-based signaling dissemination extension for next steps in signaling,” in *Proc. of NOMS’12*, April 2012, pp. 1022–1028.

[8] D. Valocchi, D. Tuncer, M. Charalambides, M. Femminella, G. Reali, and G. Pavlou, “Extensible signaling framework for decentralized network management applications,” in *Proc. of IEEE/IFIP Network Operations and Management Symposium (NOMS’16)*, April 2016, pp. 153–161.

[9] “Network Functions Virtualisation (NFV); Management and Orchestration,” 2014, http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf.

[10] M. Charalambides, G. Pavlou, P. Flegkas, N. Wang, and D. Tuncer, “Managing the future internet through intelligent in-network substrates,” *Network, IEEE*, vol. 25, no. 6, pp. 34–40, 2011.

[11] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, “Logically Centralized?: State Distribution Trade-offs in Software Defined Networks,” in *Proc. of HotSDN’12*, Helsinki, Finland, 2012, pp. 1–6.

[12] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, “Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks,” *Network and Service Management, IEEE Transactions on*, vol. 12, no. 1, pp. 4–17, March 2015.

[13] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, “On the placement of management and control functionality in software defined networks,” in *Proc. of 2nd International Workshop on Management of SDN and NFV Systems*, Nov. 2015.

[14] X. Zhou, R. Li, T. Chen, and H. Zhang, “Network slicing as a service: enabling enterprises’ own software-defined cellular networks,” *IEEE Communications Magazine*, vol. 54, no. 7, pp. 146–153, July 2016.

[15] R. Riggio, M. K. Marina, J. Schulz-Zander, S. Kuklinski, and T. Rasheed, “Programming abstractions for software-defined wireless networks,” *IEEE Transactions on Network and Service Management*, vol. 12, no. 2, pp. 146–162, June 2015.

[16] D. Tuncer, M. Charalambides, R. Landa, and G. Pavlou, “More Control Over Network Resources: An ISP Caching Perspective,” in *Proc. of CNSM’13*, 2013.

[17] D. Tuncer, V. Sourlas, M. Charalambides, M. Claeys, J. Famaey, G. Pavlou, and F. De Turck, "Scalable cache management for isp-operated content delivery services," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2063–2076, 2016.

[18] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: Responsive yet stable traffic engineering," in *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4. ACM, 2005, pp. 253–264.

[19] I. Gojmerac, P. Reichl, and L. Jansen, "Towards low-complexity internet traffic engineering: the adaptive multi-path algorithm," *Computer Networks*, vol. 52, no. 15, pp. 2894–2907, 2008.

[20] S. Fischer, N. Kammenhuber, and A. Feldmann, "Replex: dynamic traffic engineering based on wardrop routing policies," in *Proceedings of the 2006 ACM CoNEXT conference*. ACM, 2006, p. 1.

[21] "The Deltacom topology," 2010, http://www.topology-zoo.org/maps/Deltacom.jpg/.

[22] "The GEANT topology," 2014, http://geant3plus.archive.geant.net/Resources/Media_Library/Documents/GEANT_Project_TopologyMAR14_Web.pdf.

[23] Y. Zhu, C. Dovrolis, and M. Ammar, "Combining Multihoming with Overlay Routing (or, How to Be a Better ISP without Owning a Network)," in *Proc. of INFOCOM'07*, may 2007, pp. 839 –847.

[24] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding User Behavior in Large-scale Video-on-demand Systems," in *Proc. of EuroSys'06*, 2006, pp. 333–344.

[25] Y. Sun, S. K. Fayaz, Y. Guo, V. Sekar, Y. Jin, M. A. Kaafar, and S. Uhlig, "Trace-Driven Analysis of ICN Caching Algorithms on Video-on-Demand Workloads," in *Proc. of CoNEXT'14*, 2014, pp. 363–376.

[26] G. R. D. Rossi, "Caching performance of content centric networks under multi-path routing (and more)," 2011, Telecom ParisTech, Paris, France.

[27] "Mobile Ad Networks as DDoS Vectors: A Case Study," 2015, https://blog.cloudflare.com/mobile-ad-networks-as-ddos-vectors/.

[28] X. Lin, R. K. Ganti, P. J. Fleming, and J. G. Andrews, "Towards understanding the fundamentals of mobility in cellular networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 4, pp. 1686–1698, April 2013.

[29] S. Sesia, I. Toufik, and M. Baker, *LTE - The UMTS Long Term Evolution: From Theory to Practice*. Wiley, 2011.

[30] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "Ddos attack protection in the era of cloud computing and software-defined networking," in *2014 IEEE 22nd International Conference on Network Protocols*, Oct 2014, pp. 624–629.

[31] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 602–622, Firstquarter 2016.

[32] A. H. M. Jakaria, W. Yang, B. Rashidi, C. Fung, and M. A. Rahman, "Vfence: A defense against distributed denial of service attacks using network function virtualization," in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, June 2016, pp. 431–436.

[33] A. S. Pimpalkar and A. R. B. Patil, "Detection and defense mechanisms against ddos attacks: A review," in *Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015 International Conference on*, March 2015, pp. 1–6.

[34] C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms: a classification," in *Signal Processing and Information Technology, 2003. ISSPIT 2003. Proceedings of the 3rd IEEE International Symposium on*, Dec 2003, pp. 190–193.

[35] G. Oikonomou, J. Mirkovic, P. Reiher, and M. Robinson, "A framework for a collaborative ddos defense," in *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, Dec 2006, pp. 33–42.

[36] K. Kalkan, G. Gr, and F. Alagz, "Filtering-based defense mechanisms against ddos attacks: A survey," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–13, 2016.

[37] L. Li, W. Chou, W. Zhou, and M. Luo, "Design patterns and extensibility of rest api for networking applications," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 154–167, March 2016.

[38] Y. Wang and I. Matta, "Sdn management layer: Design requirements and future direction," in *2014 IEEE 22nd International Conference on Network Protocols*, Oct 2014, pp. 555–562.

[39] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[40] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proc. of INM/WREN'10*, 2010, pp. 3–3.

[41] "SDNi: A Message Exchange Protocol for Software Defined Networks (SDNs) across Multiple Domains," Jun. 2012, https://tools.ietf.org/html/draft-yin-sdn-sdni-00. [Online; accessed 08-July-2015].

[42] F. Salvestrini, G. Carrozzo, and N. Ciulli, "Towards a distributed sdn control: Inter-platform signaling among flow processing platforms," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*. IEEE, 2013, pp. 1–7.

[43] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Commun. ACM*, vol. 21, no. 7, pp. 558–565, Jul. 1978.

[44] S. White, J. Hanson, I. Whalley, D. Chess, and J. Kephart, "An architectural approach to autonomic computing," in *Proc. of International Conference on Autonomic Computing*, May 2004, pp. 2–9.

[45] E. Lavinal, T. Desprats, and Y. Raynaud, "A generic multi-agent conceptual framework towards self-management," in *Proc. of NOMS'06*, April 2006, pp. 394–403.

[46] D. Tuncer, M. Charalambides, G. Pavlou, and N. Wang, "DACoRM: A coordinated, decentralized and adaptive network resource management scheme," in *Proc. of NOMS'12*, Maui, Hawaii, Apr. 2012, pp. 417–425.

[47] F. Wuhib, R. Stadler, and M. Spreitzer, "Gossip-based resource management for cloud environments," in *Proc. of CNSM'10*, Oct 2010, pp. 1–8.

[48] R. Hancock, G. Karagiannis, J. Loughney, and S. V. den Bosch, "Next Steps in Signaling (NSIS): Framework," IETF, RFC 4080, Jun. 2005.

Dario Valocchi received its Ph.D. in Engineering from University of Perugia (IT) in 2016. The same year he joined University College London as research associate at the Department of Electronic and Electrical Engineering. His research interests focus on signaling protocols, network management, network function virtualization.

Daphne Tuncer is a postdoctoral researcher in the Department of Electronic and Electrical Engineering at University College London, UK. She received her Ph.D. from the same department in November 2013. Before joining UCL, she studied in France, where she obtained a "Diplôme d'ingénieur de Télécom SudParis" in 2009. Her research interests are in the areas of software-defined networking, adaptive network resource management and cache management.

Marinos Charalambides is a senior researcher at University College London. He received a BEng in Electronic and Electrical Engineering, a MSc in Communications Networks and Software, and a Ph.D. in Policy-based Network Management, all from the University of Surrey, UK, in 2001, 2002 and 2009, respectively. His current research interests include network programmability, adaptive resource management, content distribution,and network monitoring.

Mauro Femminella (M'01) received both the master degree and the Ph.D. in Electronic Engineering from University of Perugia in 1999 and 2003, respectively. Since November 2006, he is assistant professor at the Department of Electronic and Information Engineering, University of Perugia. His current research interests focus on nanoscale networking and communications, location and navigation systems, network and service management architectures, and big data.

Gianluca Reali is an associate professor at the University of Perugia, Department of Information and Electronic Engineering (DIEI), Italy, since January 2005. He received the Ph.D. degree in Telecommunications from the University of Perugia in 1997. From 1997 to 2004 he was researcher at DIEI. In 1999 he visited the Computer Science Department at UCLA. His research activities include resource allocation over packet networks, wireless networking, network management, and multimedia services.

George Pavlou is Professor of Communication Networks in the Department of Electronic and Electrical Engineering, University College London, UK. He received a Diploma in Engineering from the National Technical University of Athens, Greece and MSc and PhD degrees in Computer Science from University College London, UK. His research interests focus on networking and network management, including aspects such as traffic engineering, quality of service management,autonomic networking, information-centric networking, and software-defined networks. 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.