SDN Application-Centric Orchestration for Multi-Layer Transport Networks

F. Pederzolli[1], D. Siracusa[1], P. Sköldström[2], S. Junique[2], Ć. Rožić[3], D. Klonidis[3], T. Szyrkowiec[4], M. Chamania[4], V. Uceda[5], V. Lopez[5], Y. Shikhmanter[6], O. Gerstel[6]
[1] CREATE-NET, Trento, Italy, [2] ACREO, Kista, Sweden, [3] AIT, Athens, Greece
[4] ADVA, Germany, [5] Telefónica I+D/GCTO, Madrid, Spain, [6] Sedona Systems, Raanana, Israel
fpederzolli@create-net.org

ABSTRACT

Modern IP/Optical transport networks are seldom jointly operated and optimized, and do not cater to the usually implicit requirements of applications, which ultimately drive network traffic. In this concept paper we propose a Software Defined Networking (SDN) based Network Orchestrator to manage multi-layer transport networks while taking explicit application requirements into account. We discuss its architecture and requirements, an interface to allow applications to explicitly specify their requirements in a network-agnostic manner, and possible strategies to optimize the network taking these requirements into account.

Keywords: Network Orchestration, Multi-Layer Networks, Intent-based Networking, Software-Defined Networking.

1. INTRODUCTION

The applications that drive Internet traffic have evolved beyond simple requirements that can be easily and cheaply met with a best-effort network and a "reliable" end-to-end transport protocol. Today's applications can have stringent requirements in terms of latency and jitter (e.g. financial transactions), bandwidth (e.g., Video over IP), reliability, security, etc. Despite these diverse requirements, their traffic is ultimately routed over the same optical connections, barring some clever Traffic Engineering (TE) efforts in the Multi Protocol Label Switching (MPLS) layer typically used in transport networks. This "blind" approach makes it difficult to provide adequate services to all applications, whose exact requirements are implicit and not known in detail to the control plane of the transport network, which limits the effectiveness of pure TE, and results in an allocation of traffic to MPLS and optical connections which is not guaranteed to actually satisfy the service requirements for that traffic. While offering the best possible service characteristics all the way down to the optical layer (e.g. using a fully meshed optical virtual topology) to all traffic is theoretically achievable, it is also prohibitively expensive. A smarter approach is therefore needed.

In order to tackle this problem, the H2020 European Project ACINO proposes to exploit the logically centralized approach of Software Defined Networking (SDN) to realize a network orchestrator capable of: (*i*) interfacing directly with demanding client applications, exposing a North-Bound Interface (NBI) to allow them to submit service requests with explicit requirements, expressed as high-level, application-friendly "intents" rather directly using low-level network configuration commands (other than strictly necessary ingress/egress point and traffic selection parameters), (*ii*) learning the state of and controlling both the packet and optical layers of a transport network domain through an appropriate South-Bound Interface (SBI), and (*iii*) translating, through appropriate dynamic and planning algorithms, such high-level intents into optimized implementable network configuration.

This paper describes a possible high-level architecture for such an application-centric SDN network orchestrator, discusses possible languages to express intents in the NBI, gives an overview of potential strategies to compile them into optimized network configuration. This work paves the way to the implementation of the open-source ACINO network orchestrator, but also sets the ground for the debate on the usage of SDN to provide a programmable multi-layer infrastructure that really caters to the needs of applications.

The paper has the following structure: Section 2 gives a high-level overview of the architecture of the proposed Orchestrator and its main internal modules. Then, Section 3 describes the intent-based interface between the orchestrator and applications, while Section 4 gives an overview of the possible multi-layer strategies that could be employed. Lastly, Section 5 concludes and summarizes the work.

2. OVERVIEW OF THE APPLICATION-CENTRIC MULTI-LAYER NETWORK ORCHESTRATOR

The realization of the application-centric network vision relies, at a high level, on a logically centralized Orchestrator, whose primary function is translating application-level service requirements into appropriate configuration requests for both the underlying IP/MPLS and Optical network layers. Performing this functions involves several sub-processes: maintaining a multi-layer model of the controlled network, determining which resources are available to serve new connections, and selecting resources that satisfy application-level constraints, which may include instantiating additional connections in the supporting optical layer or new MPLS tunnels if needed.

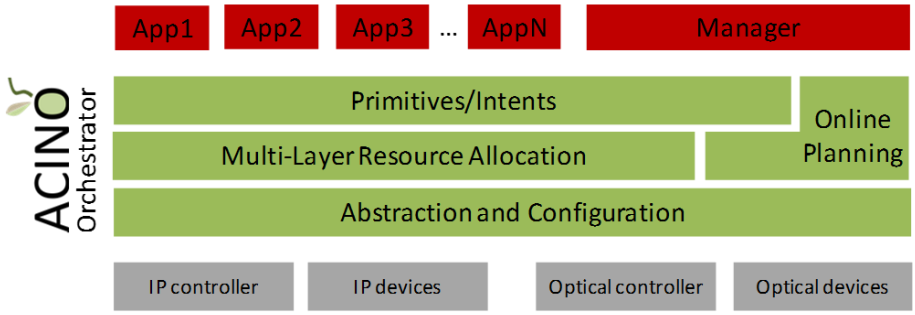

Figure 1. Main architectural elements of the ACINO orchestrator.

The main elements of the proposed application-centric Network Orchestrator are outlined in Figure 1.

At the top, the orchestrator interfaces with multiple client applications, which may include a Network Management System (NMS), by exposing a high level "Intent-based" North-Bound Interface (NBI), described in greater detail in Section 3. This interface allows to specify target service parameters, such as desired bandwidth (static or variable), maximum latency, transport security, reliability (as e.g. survivable downtime), calendaring requirements, etc., without specifying how to achieve these goals; the NBI must also expose some limited topological information (the network ingress/egress points), to allow applications to specify needed services, and basic management functions to check, update and remove requested services. Observe that these requirements are not necessarily orthogonal: e.g. max latency and survivability are interwoven, as an hard bound on total latency cannot depend on a best-effort, unknown backup path survivability scheme. It is the task of the orchestrator, through its internal Multi-Layer Resource Allocation logic, to determine how to satisfy these requirements in the underlying network environment, possibly grooming together traffic with similar needs to achieve high utilization of the underlying resources, using dynamic multi-layer provisioning algorithms. Additionally, an Online Planning module, accessible by select applications (such as an NMS) through the NBI, implements longer-term, planning algorithms for re-optimization and the computation of what-if scenarios. Some possible strategies for these algorithms are outlined in Section 4.

Finally, at the bottom of the structure of the Orchestrator an Abstraction and Configuration layer isolates the internal logic from the specifics of the underlying devices and controller, implementing a multi-layer network model and its translation to and from multiple individual (possibly standard) models and protocols (e.g. COP [1][2]) for specific devices and SDN controllers. Observe that the proposed orchestrator can, in principle, work with multiple distinct lower-level controllers, at least one per layer. These are used by the orchestrator to perform service provisioning at all layers, which entails the ability to discover the underlying topologies and network ports, their capabilities, and obtaining administrative access to their configuration. Furthermore, for circuit-oriented technologies, like MPLS or transparent Dense Wavelength Division Multiplexing (DWDM), the ability to provision specific circuits (e.g. using explicit or constrained paths) is required.

3. INTENT-BASED NORTH-BOUND INTERFACE

Applications need a flexible and expressive interface to communicate what they need from the network, together with any constraints and priorities they may have. To keep this interface as simple and portable as possible, the application should not have to understand the technical details of the underlying network domain(s), but only worry about the measurable characteristics of the service it requires. It is then up to the network, specifically the orchestrator, to decide how to implement the request in a manner that both satisfies the explicit requirements of the requesting applications and the goals of the service provider. An interface that provides a way for the application to describe *what* it wants from the network, rather than *how* it wants to configure the network can be described as *Intent*-based. This division of responsibilities gives the orchestrator full freedom to decide *how* to provide the requested service, which makes the application both more portable and simpler, and especially preserves the separation between the network and applications running at its edge. In existing SDN environments both *how* and *what* have to be specified by an application, which has to specify how to implement a particular service based on a few basic network capabilities which may even depend on specific underlying hardware support. Conversely, using intents an application and the orchestrator are only loosely coupled, removing the need for the application to have knowledge of the internals of the orchestrator, the network topology and technology.

Based on what can be realistically provided by networking hardware, we propose the definition of an application-centric intent-based NBI. This interface allows the application to discover a number of *primitives* and a *grammar* describing their valid combinations. A bare minimum of three types of *primitives* are required for an *intent* to be valid, as shown in Figure 2: *ConnectionPoints*, an Action, and a Selector. Connection Points are names describing network locations that are defined per application to fit its particular context, removing the

need for applications to be aware of e.g. router ports and their IP addresses. Actions describe connectivity patterns such as Mesh, Tree, or Path. Selectors allow the application to define which part of the traffic arriving to a *ConnectionPoint* belongs to it and should have an application-specific behaviour applied.

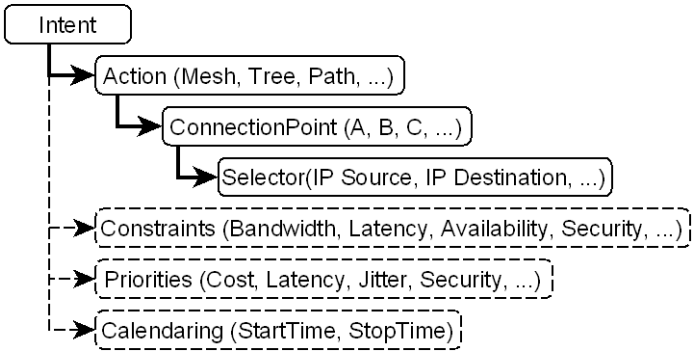

Figure 2. High-level view of primitives in an Intent

The application-specific behaviour can be described by additional *constraint primitives*, such as *Bandwidth*, *Latency*, *Security*, *Cost*, and *Availability*. To describe *when* the service is required, a *Calendaring* primitive can also be inserted. If there are multiple implementations for an intent, the main priority of an application can be indicated using a *Priority primitive*, instructing the orchestrator to choose e.g. the implementation with the lowest latency or cost. All these *primitives* are agnostic of the underlying network technology, not restricting the NBI to be used only on top of e.g. optical networks. A valid intent could be e.g. "Mesh ConnectionPoints A, B, and C, for traffic on IP-subnet 1.2.3.0/24 with bandwidth 10Gb, prioritize latency, from Monday 21st of February 02:00 GMT". A schema detailing an initial definition of the REST interfaces, and hierarchical data structures that make up the *primitives* and the *grammar* is available at [3].

Once an intent has been sent to the orchestrator, it compiles it into possible implementations. It does this by validating and resolving the different *primitives* in the intent, for example checking that the *ConnectionPoints* are defined and mapping them to actual nodes in the network graph. In the example of Mesh connectivity there are likely multiple solutions that satisfy the intent, e.g.: using only the IP network, combining both IP and Optical layers, etc. Possible solutions are derived and fed to the Multi-Layer Resource Allocation module which maps them to the network topology and calculates their costs and performance (in terms of e.g. latency). If the *Intent* contains *Priority primitives*, solutions can be ranked according to them and the best is forwarded to the Abstraction and Configuration layer for installation in the network. If no solution satisfying all the constraints of the incoming intent is found, then a negotiation with the application may begin, offering one or more solutions that partially satisfy the request and letting it decide which, if any, best suits its needs.

4. STRATEGIES FOR APPLICATION-CENTRIC MULTI-LAYER RESOURCE ALLOCATION

The Multi-Layer Resource Allocation and the Online Planning modules are the core of the application-centric Orchestrator. In concert with the other modules, they ensure that the network resources needed for the applications are available and the network is run efficiently. To that end, the modules perform algorithmic network optimization that relies on multi-layer strategies. In addition to application service requests, the optimization process must take into account a particular internal objective function (such as minimum wavelength or power usage), the features of the available equipment and the current network configuration. The Orchestrator views the underlying network as two-layered: optical- and packet-based. Both layers have their own requirements and unique features that, if exploited, allow for a more optimized network. While optimizing each layer separately is simpler, jointly optimizing both layers can achieve better overall performance, to the ultimate benefit of the both applications and network operators.

The general strategy for multi-layer resource allocation is to reuse existing routes in the network whenever possible. Such a strategy allows for faster accommodation of intents, i.e. it minimizes connection set-up time. In addition, reusing the existing packet routes and lightpaths minimizes the number of active router ports, which in turn means reduced network power consumption [4]. Clearly, whenever existing routes do not satisfy the requirements of an incoming service request the orchestrator will resort to setting up new routes. The two network layers and the two main strategies (reuse a route vs. set up new route) in each layer thus define four basic actions (Figure 3) that the resource allocation and planning modules can take. Similarly to [5] and [6], new traffic will be routed on (*i*) an existing direct packet link from the ingress to egress points (*ii*), an existing packet layer route (*iii*), a newly setup path in the optical layer, which is subsequently shown in the packet layer as a new direct link, or (*iv*) a newly setup packet layer link (and a corresponding path in the optical layer) that allows a

partial packet layer route to reach the egress point. Since the general strategy is to reuse the routes that are already available, network optimization algorithms should prefer actions (*i*) and (*ii*) whenever possible.
An inherent difficulty for the Multi-Layer Resource Allocation module, which must perform network optimization in real time, is the lack of advance knowledge of service requests. The orchestrator has a number of means to cope with this issue. Firstly, it can preserve some "high-quality" routes in the network for highly constrained traffic, where strict requirements specify e.g. low latency. This implies the counter-intuitive action of routing less sensitive traffic on paths that are longer, thus using more resources. A request for a low-latency connection can then be accommodated on the unused shorter path. Secondly, in order to be operational the network should have a few links in the packet layer, with corresponding lightpaths, set up before any constrained requests are made to the orchestrator. In real network deployments this is likely to be true by default, since the network is already carrying traffic before the orchestrator is made operational. Thirdly, since some applications require traffic reliability in the form of backup routes in case of failure, this backup bandwidth can be used to accommodate service requests that cannot wait for new optical connections to be set up (such as in actions (*iii*) and (*iv*) above). After the network has used the backup path for a new service request, the orchestrator must find and/or set up an additional path to protect the traffic from failures (i.e., either an alternative backup path or a new path for the new incoming traffic).

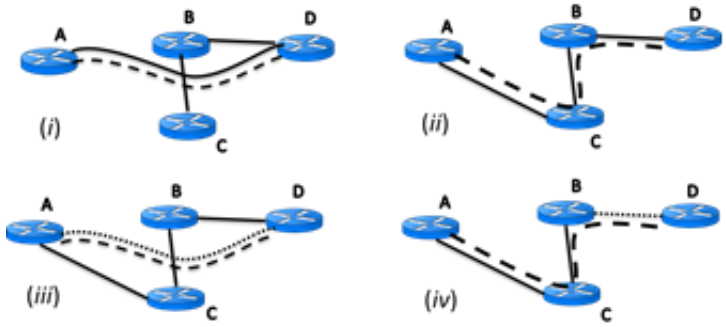

Figure 3. The four possible actions to set up a packet layer route from node A to node D. Solid lines show established, and dotted lines show new packet layer links. Dashed lines show the chosen route for each case.

Lastly, the Online Planning module re-optimizes the network. Periodically, if the state of the network is such that it could benefit from moving some traffic or lightpaths, the module calculates which to move where and how. A further function of the module is to perform what-if calculations, useful to the network operator. Specifically, it calculates the necessary network resources needed for a future set of requests, and the effects of network upgrades or failures.

5. CONCLUSIONS

This paper outlines the problem of application-centric multi-layer transport network orchestration, and proposes an SDN network orchestrator to tackle it. It gives an overview of its architecture and requirements, intent-based north-bound interface, and application-centric multi-layer resource allocation algorithms. As a future work, the authors will realize the proposed orchestrator software and test its efficiency and efficacy in both emulated and real scenarios.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Commission within the H2020 Research and Innovation program, ACINO project, Grant Number 645127, www.acino.eu.

REFERENCES

[1] R. Vilalta et al., "The need for a Control Orchestration Protocol in research projects on optical networking," in 2015 European Conference on Networks and Communications (EuCNC), 2015.
[2] Common Orchestration Protocol (COP): https://github.com/ict-strauss/COP
[3] P. Skoldstrom, "Initial release of DISMI", in Zenodo. 2016, DOI:10.5281/zenodo.46586
[4] W. Hou, L. Guo and X. Wei, "Robust and Integrated Grooming for Power and Port-Cost-Efficient Design in IP Over WDM Networks," Journal of Lightwave Technology, vol. 29, no. 20, pp. 3035 - 3047, 2011.
[5] H. Zhu, H. Zang, K. Zhu and B. Mukherjee, "A Novel Generic Graph Model for Traffic Grooming in Heterogeneous WDM Mesh Networks," IEEE/ACM Transactions on Networking, vol. 11, no. 2, pp. 285 - 299, 2003.
[6] S. Martinez et al., "Assessing the Performance of Multi-Layer Path Computation Algorithms for different PCE Architectures," in Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), 2013.