International Journal of Electrical and Computer Engineering (IJECE)
Vol. 7, No. 4, August 2017, pp. 2261 – 2277
ISSN: 2088-8708, DOI: 10.11591/ijece.v7i4.pp2261-2277

A Proposal for End-to-End QoS Provisioning in Software-Defined Networks

Francesco Lucrezia[1], Guido Marchetto[2], Fulvio Risso[3], Michele Santuari[4], and Matteo Gerola[5]
[1,2,3]Polytechnic of Turin, Torino, Italy
[4,5]CREATE-NET, Trento, Italy

Article Info

Article history:
Received: Feb 17, 2017
Revised: Jun 2, 2017
Accepted: Jun 17, 2017

Keyword:
QoS
Provisioning
Networking
SDN
ONOS

ABSTRACT

This paper describes a framework application for the control plane of a network infrastructure; the objective is to feature end-user applications with the capability of requesting at any time a customised end-to-end Quality-of-Service profile in the context of dynamic Service-Level-Agreements. Our solution targets current and future real-time applications that require tight QoS parameters, such as a guaranteed end-to-end delay bound. These applications include, but are not limited to, health-care, mobility, education, manufacturing, smart grids, gaming and much more. We discuss the issues related to the previous Integrated Service and the reason why the RSVP protocol for guaranteed QoS did not take off. Then we present a new signaling and resource reservation framework based on the cutting-edge network controller ONOS. Moreover, the presented system foresees the need of considering the edges of the network, where terminal applications are connected to, to be piloted by distinct logically centralised controllers. We discuss a possible inter-domain communication mechanism to achieve the end-to-end QoS guarantee.

Corresponding Author:
Name: Francesco Lucrezia
Affiliation: Polytechnic of Turin, Italy
Address. Corso Duca degli Abruzzi, 24, 10129 Torino, Italy
Email: francesco.lucrezia@polito.it

1. INTRODUCTION

Internet service providers (ISPs) are striving to innovate their network infrastructures at the pace content providers do with their services. Digital contents are consumed by smart phones and sophisticated terminal stations that continuously evolve together with the applications they host. Interestingly enough, the evolution of the Over-The-Top (OTT) services is mainly happening without the aid of network service providers, within the best-effort data traffic channel in the access networks. Recently, a new plethora of applications requiring a RTT delay of around 1ms have been grouped under the hat of tactile-internet applications: a tactile sensor reads information and a connected system reacts with actuators seen by a human within 1 ms [1].

Although we are still far from achieving end-to-end RTT of around 1ms with wireless communications, ISPs need to be ready to re-architect their software control-plane in order to fully exploit the enormous potentials offered by their infrastructures.

The goal of this paper is to present the design and a prototype implementation of a control-plane network application for provisioning dynamic end-to-end QoS profiles to end-user applications. The current adoption of distributed control algorithms forces the use of the same signaling protocol (e.g. RSVP, BGP-LS) in all the data-path nodes, not taking into account the resistances inevitably present between device vendors and between administrative domains. For this reason the Service-Level-Agreements (SLAs) between a service provider and its customers or between providers are still mainly static. Moreover, the experience has shown that the scalability issue of the core network in maintaining per-flow state for resource reservation in each node along a path prevented the diffusion of RSVP and integrated services in general. As discussed in [2], per-flow service treatment does not scale in the Internet core; backbone routers must be fast and only an *aggregate behaviour* is feasible. Instead, it is important to enable such treatment at the edge of the network where mass of users enjoying a mixture of heterogeneous applications share indistinctly the same portion of the network as in the case of cellular access networks; performance degradation is likely to happen in the access links where an increasing number of traffic sources and sinks can introduce a significant amount of queueing delay. Given these considerations, we believe that an hybrid combination of flow-based and class-based traffic treatment, respectively at the edge and in the core of the network, could enable guaranteed services for current and future real-time applications. Since these applications could have terminals deployed all around the globe, the end-to-end provisioning will have to span a chain of administrative domains, requiring an east-west communication interface to convey data that vary from classic inter-domain routing information exchanged via BGP. If we make the assumption that QoS requirements requested by the customer applications are satisfied in the core network, at least for what concerns a delay bound, and up to a maximum bandwidth allocation, then we can think to overlay an integrated service scheme on top of the current deployments where class-based treatment is applied, as long as the resource admission control system is able to map the dynamic service requests to the statically allocated resources in the core. Our proposal is a signaling scheme for path reservation and configuration whose implementation does not require the involved data-path devices to be bound to a single control protocol. The aim is to solve the interoperability problem in provisioning end-to-end guaranteed services, in a multi-vendor, multi-technology and multi-domain environment by exploiting current software technologies advances; in particular, the decoupling between functional intents and the way they are accomplished is crucial.

The paper is organised as follows: Section 2. gives a high level description of the complete system workflow, the first part of Section 3. is dedicated to a general introduction to the QoS and to the Internet technologies adopted to achieve it. Here is where the most of the related works are considered. Then it focuses on our specific use-case scenario and on the RSVP critical issues. In Section 4. we discuss the communication interface between clusters or domains of networks required to achieve the end-to-end provisioning, while in Section 5. a brief contextualisation of our work into a policy management system is presented. Section 6. and all its subsections contain the specific details of our prototype system implementation and Section 7. presents some benchmark results on the service request computation time. Finally, Section 8. concludes the paper.

2. SYSTEM WORKFLOW

The overall process is activated upon the occurrence of some external event, either sensorial or caused by the human will, that triggers the dispatch of a request sent by the user application. The request contains a user authentication token, an application identifier, information about an endpoint to contact together with additional flow specifications, and a QoS profile containing delay and bandwidth requirements, plus an amount of time (or an estimate of it) for which the profile is required. A previous agreement between the user and the network operator is made in order to convey to a traffic plan based on its dynamism, amount of data, QoS parameters, number of requests and possibly other parameters. The request is sent to a manager application running on top of the local domain controller that is listening for incoming connections from registered users. The authentication token, previously generated in a hand-shake phase is verified and the content of the payload is parsed and elaborated as follows.

The endpoint information, either a destination address (L2 or L3, depending on the use-case) or the hostname of the machine to be contacted, is used as look-up key to get the collection of candidate paths existing between the end terminals of the user application. Then an *admission control* routine runs to check the availability of a suitable path where resources are to be reserved for the subject QoS profile. Once a path is selected, the network application has to instruct the core controller to setup a priority flow between the endpoints of the user application; for the sake of simplicity, we will refer to point-to-point paths, although the solution is equally applicable to paths with multiple destinations (> 2).

It may be necessary to establish connectivity between the endpoints, other than traffic control's rules. For example, in a pure Openflow network where none of the routing protocol suite runs in the data-path devices, the controller would prescribe a set of flow rules containing forwarding instructions, together with QoS constraints. If forwarding rules have been previously installed on the data-path devices, then only traffic classification and shaping is to be done through device-specific configurations.

To setup a priority flow, the manager application issues the setup of custom queues in the devices along the selected path and sends an intent request to the controller's core with the specification of the target flow. An intent is an abstraction used by the applications to specify their high-level desires in form of policies. The ONOS network controller [3], used in our prototype implementation, is the first open-source controller that provide such feature to

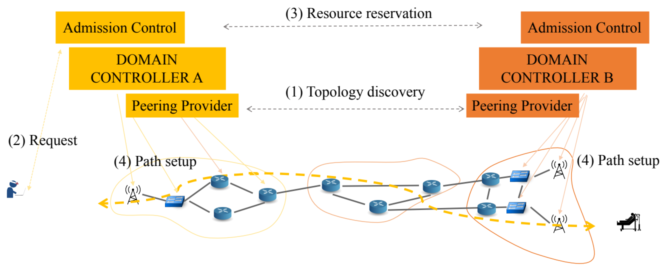


Figure 1. Reference scenario

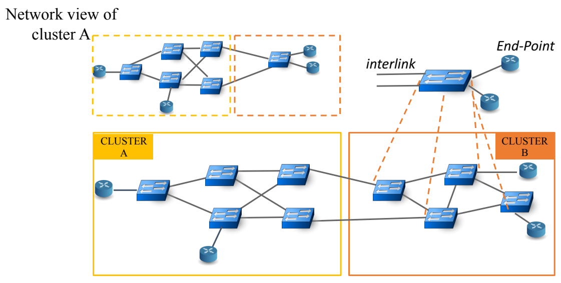


Figure 2. Domain topology abstraction

its applications. The intent is then split by the core into device-specific flow rule requests dispatched to the proper software drivers of the underlying devices. Queues are also configured by means of device-specific drivers.

As mentioned before, the endpoints can normally span multiple domains and, clearly, each domain has to take care of its portion of the network. A key point of the system is the topology abstraction (Figure 2): all forwarding devices of the local domain are exposed to the controller with the same abstraction model, as is an entire remote domain topology, view as a single device (a big switch) whose ports are connected to terminal endpoints or to other domain topologies. When the endpoint of the application requesting a priority path resides on a different domain, the admission control routine recognises the presence of a virtual device associated to the remote domain and as a consequence queries the network manager application of the remote domain in order to establish an end-to-end resource reservation along the path between the endpoints (Figure 1). Once the process converges, the domain controllers can simultaneously setup the path in their own portion of the network, the manager application that received the original request sends a response back to the user application that can start sending priority data over the network.

3. QOS PROVISIONING

3.1. General Discussion

QoS is introduced in packet-switched networks in order to apply a special treatment to specific data packet flows. At the device level, QoS is achieved through traffic classification, shaping, and scheduling at the egress ports; at the ingress ports packets are filtered for policing. Classification determines which treatment each data packet has to undergo, shaping is used to control customer input data rate to conform to the SLAs, while scheduling affects delay and throughput of the data packet flows. Given conformance to a SLA, scheduling at the network interfaces is the lowest level operation gearing QoS provisioning, also known as *service discipline*.

At the path level, QoS is achieved through resource reservation over the whole set of nodes along the path. The reservation must be secured end-to-end and the resource allocation in one node must be consistent with the others along the path. A path selection with the end-to-end delay constraint is subject to inaccurate information due to the dynamic nature of the delay and hence subject to theoretical intractability ([4] [5]). However, end-to-end delay and delay jitter bounds have been computed by means of network calculus applied to queueing systems modelling the networks ([6, 7, 8]). Academic and industrial communities have been very active in the last decades in investigating network models and algorithms to solve the *QoS routing* problem, also known as the *multi-constrained path computation* problem ([9, 10, 11, 12, 13, 14, 15]). In [2] they cover all the important components of the QoS provisioning in Internet as it has been conceived in the last decades: integrated services, RSVP ([16, 17]), DiffServ [18], Multi Protocol Label Switching (MPLS [19]) and constraint-based routing.

Today network operators employ MPLS mainly for layer 2 and Layer 3 Virtual-Private-Network (VPN) services ([20]), while constrain-based routing for Traffic Engineering (TE) operations is complex to achieve in a complete distributed control-plane. Moreover, TE is not useful in presence of congestion. This happens at the bottleneck links that typically reside in the last mile towards the customers. DiffServ model within a single AS is employed by ISPs for class-based treatment of the data packets; but the validity of the Type of Service (ToS) field in the packet IP header may lose completely meaning when traversing multiple administrative domains. In other words, within a single administrative domain the class-based QoS provisioning is theoretically easy to achieve and technology is not the hurdle, while policy and economic factors have the major impact on the fate of the QoS provisioning in the multi-domain scenario. IntService and the RSVP signaling protocol instead did not take off even within the single administrative domain. RSVP is used for labels distribution in G/MPLS but it failed in its primordial intent. In order to accomplish the process described in Section 2. an end-to-end guaranteed service must be provided. In the next section we discuss the critical issues of RSVP and in what our proposal differs from it.

3.2. Comparison with RSVP

Path Computation and Routing. RSVP uses a combination of Constrained Shrotest Path First (CSPF) algorithm and Explicit Route Objects (EROs) to determine how reserved traffic and signaling messages are routed over the network; RSVP is a distributed signaling protocol and it needs routing to work. EROs are a mean to explicit indicate some nodes that must belong to the reserved path; in order to force a specific path through a set of nodes you should enter and configure each node with specific EROs instructions. If the total bandwidth reservation exceeds the available bandwidth specified across the link for a particular path segment, the path must be recomputed through another route. If no segments can support the bandwidth reservation, path setup fails and the RSVP session is not established.

In our solution the path computation is independent from the routing protocol. Different routing and forwarding scheme can be used to build the path in the underlying network: flow-rules, labels, tunnels. But no routing of the signaling scheme is needed; centralised path computation is clearly much faster and protocol-independent. The controller only needs the view of the topology as a connected graph. To force the reservation in a specific path, you can directly reserve resources and install forwarding rules into the proper devices at once. The candidate paths are collected from the store and a suitable one is found before injecting resource reservation rules into the network. This occurs in the centralised controller within a single software process.

RSVP is simplex. In RSVP the reservation process is applied in a single direction of the path. To have full duplex reservation, the number of operations and the messages exchanged are doubled.

In a centralised network controller, you can equally provide simplex or duplex reservation via a single software entity.

Admission and Policy Control. RSVP Admission and policy control is applied to each node. So RSVP implementation must be integrated with each node's traffic and policy control module, thus increasing the chance of interoperability. RSVP must provide QoS service characterisation within opaque objects parsed by each network node.

In our proposal the QoS service characterisation is completely decoupled by the signaling protocol/scheme and must be embedded only into the front-end APIs consumed by the customer applications. The admission and policy control is applied only once, in each domain, in the central controller. An RPC-based mechanism is used for configuring the traffic control module given the possibility to fulfil the request. Our prototype relies on ONOS. ONOS provides common abstracted behaviours for traffic selection and treatments that are translated into device-specific rules.

First speaker. RSVP session initiator is the inbound router running RSVP in conjuction with other protocols (e.g. MPLS or GMPLS in case of label distribution). If RSVP is embedded in a host application, then the first network node should speak RSVP, otherwise a tunnel between the application and the first RSVP-capable device shall be created thus increasing the number of operation required for RSVP signaling to work.

In the presented solution the session initiator is a user application featured with proper APIs for contacting the controller. The idea is to keep the API consumer implementation as simple as a REST client that has the capability to create, remove, update and delete a priority path. Such client would be provided for different software environments.

Scalability. As per [21], the scaling problems of RSVP are linked to the resource requirements (in terms of processing and memory) of running RSVP. The resource requirements increase proportionally with the number of sessions. Each session requires the generation, transmission, reception and processing of RSVP Path and Resv messages per refresh period. Supporting a large number of sessions, and the corresponding volume of refresh messages, presents a scaling problem.

A centralised control plane presents the same scalability issues concerning the state maintenance of an increasing number of sessions in the data-path nodes. But it only matters the traffic control and flow rules, while processing and singnaling overhead are significantly lowered.

Complexity. Finally, and maybe the most relevant obstacle to success, RSVP is complex because it was designed with IP multicast in mind, intermediate nodes have to merge resource reservation requests coming from the receiver nodes. Moreover, the basic RSVP reservation model is "one pass": a receiver sends a reservation request upstream, and each node in the path either accepts or rejects the request. This scheme provides no easy way for a receiver to find out the resulting end-to-end service. To solve this issue an enhancement was proposed [22], introducing further complexity in the concretization of RSVP and the integrated services in general.

Orchestrating the data-path nodes from within a central controller avoids the issues related to the exchange of asynchronous signaling messages. The decision process not being distributed decreases the complexity in maintaining a single state of the system.

3.3. End-to-End Behaviour

The end-to-end QoS profile model shall follow the one described in the Specification of Guaranteed Quality of Service [23]. As per [23], "*the end-to-end behaviour provided by a series of network elements is an assured level of bandwidth that, when used by a policed flow, produces a delay-bounded service with no queueing loss for all conforming datagrams*". We invite to refer to the specifications for further clarification about the QoS model taken into account. Each network node must provide a service that matches, with some error bounds, the fluid model ([24, 25]) through the token bucket scheme with paramters (b, r, p), respectively the bucket size, the token rate and the peak rate. The QoS request includes a maximum end-to-end delay bound, d_{req}, that shall be guaranteed between the application terminals.

In the centralised controller, the link providers are responsible for notifying the presence of the information on the links they are provider for (propagation delay, transmission capacity and the maximum transmission unit); these information are stored in the controller database upon discovery of the link itself. Likewise, the device providers in the southbound must export other relevant information, such as the delay error terms representing how the device's implementation of the guaranteed service deviates from the fluid model in each network interface (the C_{tot} and D_{tot} in the formula 2 below).

On a link l with capacity c_l, we define a minimum bandwidth reserved to the best effort traffic, R_{be_l}. Let R_i be the allocated bandwidth for a flow i. On each link, the total number of accepted profiles N is subject to:

$$N : \; c_l \geq R_{be_l} + \sum_{i}^{N} R_i \tag{1}$$

The end-to-end delay bound as defined in [23] is:

$$[(b - M)/R * (p - R)/(p - r)] + (M + C_{tot})/R + D_{tot} + \sum_{l} d_{p_l} \tag{2}$$

With $r <= p <= R$, M being the path Maximum-Transmission-Unit and $\sum_l d_{p_l}$ the propagation delay sum.

Statement 1 imposes that the sum of the allocated bandwidths for N flows must not exceed the capacity of the link. Flows requesting a maximum delay bound are assigned to higher priority queues w.r.t. to the best effort traffic. It is possible to assign the same high priority queue to more distinct flows, as long as statement 1 holds. R shall be chosen such that d_{req} is greater or equal to the value computed in equation 2, provided that d_{req} is greater than the fixed delay terms D_{tot} and $\sum_l d_{p_l}$. These constraints must apply on all links of a candidate path between the end terminals of the customer application; a new queue is created for a new flow if they are satisfied. As mentioned in the previous section, the *admission control* routine occurs only once per domain (more details in Sec. 4.), in the central controller. Network elements must export the proper information, while the drivers have to translate a service request into device-specific traffic control rules.

The provisioning of a guaranteed service along a path of several devices and links is possible only through a cross-vendor and cross-technology solution. This leads to the adoption of software driver modules installed into the centralised controller. These drivers converts the protocol-agnostic rules into device-specific instructions and are essential to solve the interoperability problem derived by the presence of devices from multiple vendors and technologies. For example, in a LTE cellular network, the high-level profile is mapped to a standardised QoS Class Identifier (QCI) by the proper software driver; the mapping would be followed by the setup of the packet data network gateway and the mobile station with some scheduling rules applied to the target data flow [26]. The same high-level profile has to be translated into a specific setup on the backhaul that provides the connectivity towards the core network consisting of all the required switches to aggregate the traffic from the access cellular network [27]. These switches could be, for instance, pure Linux devices in which case the driver would execute a remote procedure call configuring the involved interfaces with the well-known commands suite *tc qdisc*, *tc filter* and *tc class* for setting up the queues. If instead a switch is an Openflow-enabled device, *classification* and *priority* come within the forwarding rules, while the queue configuration for the *service discipline* must be supplied on a separate communication channel, for example, through OVSDB protocol in Open-vSwitch [28] (Fig 3).
Together with the local domain (or local cluster in the single administrative domain) our framework adds the reflection of such operations into the remote domain (cluster) where the endpoint of the customer application requesting the service resides.

Note that we control the edge devices on each side of the communication while leaving aside the backbone routers where per-flow service treatment does not scale. While rfc-2212 states that all the nodes of a path should take part of the resource reservation process for equation 2 to hold, we argue that the dynamic resource reservation at the edge of the network can occur transparently w.r.t. the statically allocated resources of the core where the QoS exists only for classes of traffic, rather than flows, in form of virtual circuits created with protocols such as MPLS or GMPLS (Fig. 4). If the backbone is treated as a composition of these circuits rather than a composition of nodes and links, then a resource mapping between the dynamic and the static portions of the network resolves in representing these circuits as aggregate elements into the topology view of the controller. Path-Computation-Element (PCE) describes a model to address the problem of constrain-based path computation in conjunction with a label switched protocol ([29, 30]). An all in one orchestration framework for the complete set of the network elements is left as a future work.

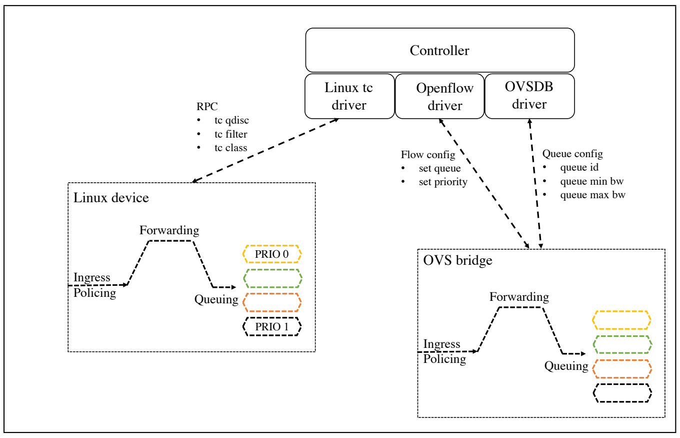


Figure 3. Driver modules in the controller are the means for device-level configuration of the queues

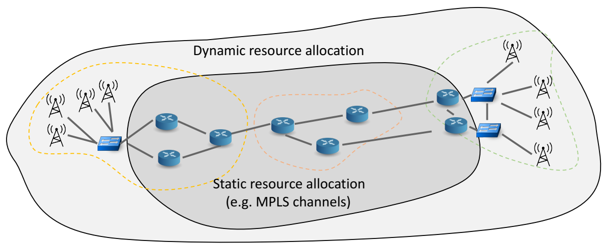


Figure 4. Two allocation schemes for two layers of the network

4. EAST-WEST COMMUNICATION INTERFACE

When the terminals of the application requiring a priority path are located in two portion of the network piloted by distinct controllers, a communication mechanism between these controllers is necessary in order to exchange the proper information during the reservation process. From hereafter, we will use the term domain and cluster interchangeably to indicate distinct portions of the network.
At the origin of the communication, there is the route discovery phase to share the endpoints of each domain. A design choice is to be made on how and when to expose the network element parameters related to the topology itself. There are two options: **a)** sharing these information during the route discovery phase, or **b)** avoid to pre-share the parameters and collect them during the resource reservation process on-demand, that is, every time a new request arrives.
Suppose we have a portion of the network under domain *A*, and another portion under domain *B*, then suppose that at some point in time an application connected to *A* asks for a priority path that includes an endpoint in domain *B*. The two different approaches are described in the following sections.

4.1. Pre-Shared Network Parameters and Bandwidth Resource

With option **(a)**, the controller *A* has already all the information required for the end-to-end admission control routine when the request arrives. This means that the topology exposed by *B* to *A* shall include all the necessary network infrastructure parameters, $(c_l, MTU, C_{tot}, D_{tot}, d_{tot}^{prop})_p$ for all p in the set of paths that *B* is willing to expose to *A*, which in turns implies that the exposed topology shall be detailed enough for *A* to select a suitable path. This requires a more complex topology abstraction than the single big switch as depicted in Fig. 2, because clearly with the single node abstraction you cannot have as many path properties as you would have with a network of nodes. You can always achieve a certain level of aggregation by hiding elements of the *B* topology, by computing the aggregate parameters for some paths towards the destinations and then exposing a virtual topology composed by only these paths. In this case each domain controller should maintain a mapping between the local physical devices and links and the virtual ones exposed to the remote domains. Other than the topology abstraction, there is one major issue with this approach: the computation of the aggregate, rate-related delay error term, C_{tot}, which is theoretically not feasible when the computation occurs, for instance, in the domain controller *A* for some devices of domain *B*, because C depends on the parameter r, the rate requested by a user application. So either C, expressed as a function of r for each device, is shared during the topology discovery phase, which means exposing the entire topology, or it must be computed on-demand, by the domain controller *B* as described in the next section.

4.2. On-demand Network Parameters and Bandwidth Resource

In this case, the path parameters are collected and exchanged during the admission control routine. The domain *B* is requested to run the resource reservation process in its own domain. Controller *A* forwards the application request parameters (i.e. d_{req}, the tuple (b, r, p), the domain ingress point and the target destination) to *B*, which replies with a tuple $(M, C_{tot}, D_{tot}, d_p)^B$ and an upper bound on R, chosen based on a proper selected path, if available, so that *A* can compute the end-to-end delay bound before proceeding with the actual reservation and path setup. This way the topology abstraction can be kept as simple as a single big switch whose function is to merely offer a point of connection to the remote destinations to any domain controllers with which there is a peering. The network parameters and the resource selection comes directly from an up-to-date decision process made within the concerned domain. The computation of C_{tot} can be actually computed while masking the details of the local topology. This approach is the choice of our implementation prototype described in the rest of the article.

4.3. Resource Management

The network parameters used for the delay bound computation in equation 2 are mainly static, except for the bandwidth R, the dynamic network resource under consideration. We assume unlimited buffer space for the queues, or at least enough to fill the entire bandwidth resource in any link of the network. Within each domain a resource management system should track the allocated and the available bandwidth in each link of the underlying network. From the inter-domain communication perspective, the bandwidth resource management unfolds three cases depending on the use-case scenario:

- **Full share**. The bandwidth is completely shared between applications, regardless of the domain they reside. This can be the case where the control plane of a single administrative domain is split into multiple regions for performance reason or because of different underlying physical networks. In this scenario, it is necessary to update the exposed resource each time an application obtains or releases a portion of the bandwidth. Upon a new allocation or release in one domain, a message is sent in broadcast to all the other domains with which there is a peering. The message is read by the remote controllers and their local resource stores are updated accordingly. In the on-demand mode of communication, Section 4.2., each domain manages its bandwidth resource independently and only during a reservation process the resource availability in a remote domain is determined.
- **Partial share**. This case is equal to the previous one but only a portion of the bandwidth is shared with the remote domains.
- **Partial static share**. A domain controller advertises to each remote controllers a virtual value of the bandwidth resource during the route discovery phase and no further update messages are exchanged between controllers. This value is the static portion of the bandwidth allocated to each remote domain. In this case, also with the on-demand mode of communication a controller can determine the availability of bandwidth even before contacting a remote domain. This is the case of the multi administrative domains scenario.

5. POLICY ENFORCEMENT

Policy-based QoS management is of primary importance for network operators. If the *service discipline* at the network interface is the lowest level operation gearing QoS, at the top level we have the SLAs expressed in terms of policies. The SLAs consist of a set of specifications that are translated by the network manager into device level primitives (e.g., forwarding rules, queue configurations, traffic shaping policies, etc.). In [31] [32] the authors propose automatic policy based management system in the Internet DiffServ architectures. They present a framework for policy management that reacts to network state changes or customer users requests to dynamically re-adapt the policy enforcement. In [33] a management framework for automatic policy enforcement is introduced in a network controller based on Openflow; they describe all the necessary functional components of the system without entering in the implementation details of any of them thus avoiding to discuss how do they actually interact between each other.

The focus of the present article is on the QoS provisioning in the economic context of dynamic SLAs; this framework foresees the possibility to be integrated with an existing policy-based management system. The concerned SLAs are between a service provider and its customers and between service providers who cooperate to provide an overall service that can span multiple administrative domains. Between the customer and the provider, a set of APIs can be embedded directly into the customer applications and layered on top of an existing policy management system. The network operator could also provide ready-to-use applications for specific services (e.g. a remote health control system). Here we limit the discussion by listing the additional information needed by the policy manager in order to conform the ingress traffic to the dynamic SLAs.

Policy to regulate the interaction with customer applications:

- List of user and application IDs that are allowed to request a service
- Upper bound on the amount of bandwidth each user could request
- Maximum amount of time each user is allowed to retain a priority path
- Amount of bandwidth reserved to the best effort traffic
- Set of destinations for which a user could request a priority path
- Set of network elements that cannot be part of the reservation process
- Pre-configured queues for selected customers or applications

Policy to regulate the interaction between providers:

- List of peer domains that are allowed to interact with the local domain
- List of destinations to expose to the remote domains
- Topology abstraction to expose to the remote domains
- Virtual bandwidth resource associated to the exposed destinations
- Aggregate parameters selection, $MTU, C_{tot}, D_{tot}, d_p$
- Number of total service requests that a remote cluster is able to perform
- Pre-configured queues for selected peers

6. ARCHITECTURE

6.1. High-level System Components

The main functional modules in the control plane are *protocol agnostic* thanks to the separation of concerns given by the network controller architecture of ONOS (see Section 6.2.) that is partitioned into:

- Protocol-aware network-facing modules that interact with the network
- Protocol-agnostic system core that tracks and serves information about network state
- Applications that consume and act upon the information provided by the core

At the application layer resides the *admission control* and *resource allocation* routine. It guarantees the correctness of the QoS provisioning to the end-user applications; it has to dynamically setup and teardown multiple and concurrent QoS profile sessions and verify that everything in the underlying network is up-to-date and in a consistent state. In the core controller there are several components required to accomplish the complete reservation process, see Figure 5, while in the network-facing layer we have as many drivers as the number of different devices in the underlying network and a communication interface used to exchange data between the domain network controllers. Such interface has to take into account several aspects of the system: routes to destinations discovery, network topology elements exposition, resource reservation parameters and a policy-driven mechanism to abide to the SLA made between the involved administrative domains. We leverage on a project called ICONA to fulfil the remote topology and destinations discovery function, see Section 6.4.. The resource reservation parameters are currently exchanged during the *admission control* routine at the application layer, using a prototype REST channel interface. The integration with a policy manager is left as a future work.

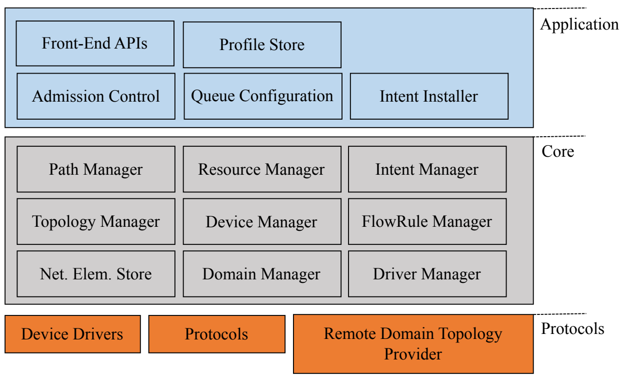


Figure 5. Controller's Main Components

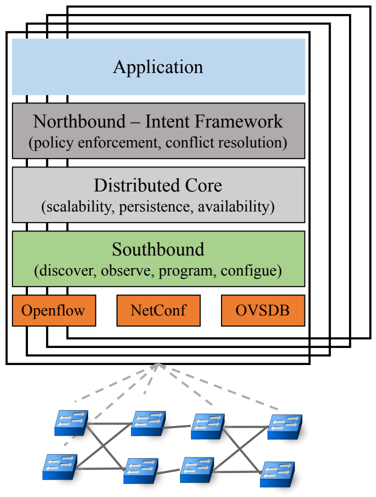

Application
Northbound – Intent Framework
(policy enforcement, conflict resolution)
Distributed Core
(scalability, persistence, availability)
Southbound
(discover, observe, program, configue)
Openflow
NetConf
OVSDB

Figure 6. ONOS distributed architecture

6.2. An overview of ONOS

The Open Network Operating System (ONOS) is a software defined networking (SDN) OS for Service Providers, that is targeting scalability, high availability, high performance and abstractions to make it easy to create apps and services [34].

ONOS implements a distributed architecture in which multiple controller instances share multiple distributed data stores with different level of consistency. The entire data plane is managed simultaneously by the whole cluster. However, for each device a single controller acts as a master, while the others are ready to step in if a failure occurs. With these mechanisms in place, ONOS achieves scalability and resiliency. Figure 6 shows the ONOS internal architecture within a cluster of four instances. ONOS is based on software modules managed by the Apache Karaf suite [35], a set of java OSGi based runtime and applications. It provides a container into which various component can be deployed, installed, upgraded, started and stopped at runtime, without interfering other components. The southbound modules manage the physical topology, react to network events and program/configure the devices leveraging on different protocols. The distributed core is responsible to maintain coherent information, to elect the master controller for each network portion and to share information with the adjacent layers. In case of a failure in the data path (switch, link or port down), an ONOS instance becomes aware of the event through the southbound modules, computes alternative paths for all the traffic crossing the failed element, and notifies them to the distributed core; then, each master controller configures accordingly its portion of the network. The northbound subsystem offers an abstraction of the network and the interface for applications to interact and program the NOS. Finally, the Application layer offers a container in which third-party applications can be deployed. Applications on top of ONOS can benefit of the Intent Framework. The ONOS core accepts the intent specifications and translates them into actionable operations on the network environment. These actions are carried out by the intent installation process, such flow rules being installed on a switch, or optical lambdas (wavelengths) being reserved.

6.3. The Manager Application

The manager application is a standard, on-platform ONOS application. Its main function is the *admission control* routine. We separate the routing and resource assignment process into two steps: the collection of candidate paths between the application terminals through the ONOS *PathManager* and the selection of the one that satisfies the constraints imposed by the basic *admission control* scheme described in Section 3.. The state of the underlying network resources is tracked by the *ResourceManager*, backed by a distributed store, which is queried during this process to reserve the bandwidth resource. The local path parameters are collected by the internal store populated with the information of the underlying network elements, while in presence of virtual domain devices, the parameters are queried to the remote controllers and then merged with the local ones. If a suitable path is found using algorithm 1, the bandwidth in each link is temporary reserved (locked), priorities queues are setup in each device through the *QueueManager* module and an intent is submitted by the *IntentInstaller* (Fig. 7). If the installation is successful, the *ResourceManager* is requested to allocate the previously locked resource, otherwise a rollback is performed on all the previous actions, bandwidth reservation and queues configuration. The collection of the candidate paths is subject to the constraint imposed by the eq. 1; other criterions may be applied, taken, for instance, by a configured policy. A REST applet that implements the APIs consumed by the end-user applications and by the remote domain controllers to request and terminate the setup of priority paths is also part of the manager bundle. All these components exploit the service-based OSGi model to communicate each other within the platform.

Algorithm 1 Pseudo-code of the admission control routine. *Assuming $p >> r, R$

```
1:  $(b, r, p, mtu, d_{req})$: user app request

2:  $R \leftarrow r$
3:  collect all paths whose spare capacity on each link is $\geq R$
4:  **for all** p in paths **do**
5:    **if** p contains a domain device **then**
6:      collect $(R, MTU, C, D, d_p rop)_p$ from the remote domain
7:      merge local and remote path parameters
8:    **end if**
9:    $R \leftarrow \max\{r,\ (M + C_{tot})/(d_{req} - D_{tot} - d_{tot}^{prop})\}$*
10:   **if** R can be allocated along the path **then**
11:     allocate R and setup queues
12:   **else**
13:     rollback and try next path
14:   **end if**
15: **end for**
```

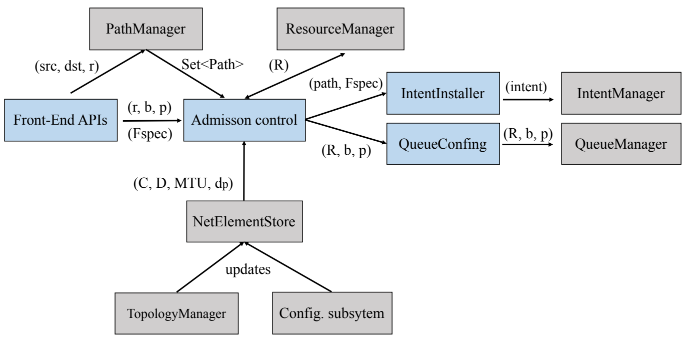


Figure 7. Admission control workflow.

6.4. ICONA: Inter-Cluster ONOS Network Application

ICONA [36] extends ONOS to enable:

- a single administrative domain network to be divided into multiple regions controlled by a cluster of controllers to decrease event-to-response delays, increase the robustness to geographical location faults and distribute the load, not only among a single cluster, but also among multiple clusters.
- the communication with other administrative domains leveraging on an East-West interface to ensure the full control of services and events between domains and enforce configuration policies between domains

ICONA divides the SDN control plane in clusters, each one taking care of a portion of the entire geographical network and provides an east-west communication interface to enable programmability of the entire network. It shares an aggregated form of the network topology between two or more clusters of ONOS (Fig. 2) and offers a transparent, end-to-end, programmatic interface to the NB applications to accommodate connectivity requests between two or more end-points crossing multiple clusters. It is completely transparent to the northbound applications; the remote topologies are exposed to the core as normal devices (Figure 8) so that applications and users can interact with them

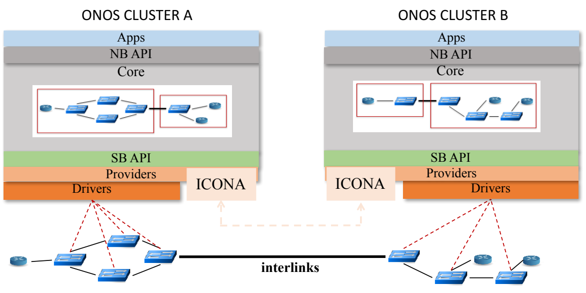


Figure 8. ICONA as a peering provider

through the ONOS GUI, CLI, the REST and Java APIs. It leverages on ONOS to take advantage of its native fault tolerance management and scalability.

ICONA is composed by two main logical components: the *Provider* containing the main logic which is agnostic to the communication protocol and a set of pluggable and independent mechanisms for the communication with remote clusters representing the *Southbound Interface* (SI). The decoupling between functionalities and implementation is achieved through the use of java interfaces, the modularity and the dynamic activation/deactivation of multiple components is achieved thanks to the OSGi framework. Each peer is associated to a communication mechanism by configuration. The *Provider* component receives from the southbound mechanisms the topology of the remote clusters in form of virtual devices with ports connecting endpoints and/or other cluster topologies. The device provider module exposes those devices to the core; the same happens for the endpoints attached to the virtual device. These endpoints are represented in ONOS by a L2 address and a location attribute, plus a number of metadata, such as ipv4/ipv6 addresses, hostname etc. The inter-cluster links are retrieved partially by configuration (link ID and local physical connect point) and partially by the remote controller that notifies a virtual connect point whose metadata contains the same link ID. As the reservation process described in the article targets the access network providers, the inter-domain links represents indirect connections traversing the backbone transit domains. A centralised and automatic inter-link discovery and fault recovery mechanism is matter of future investigation. A southbound mechanism is an implementation of the communication system between clusters. Basically, it is a software component in charge of translating the provider's requests into protocol-specific, network operations and the remote clusters messages into abstracted notifications via the SI. This component performs the exchange of the information with message encoding and decoding, and does not retain any system state except the status of the remote clusters. In our prototype implementation we use a REST client/server peer-to-peer architecture as the communication mechanism. The client is in charge of sending local topology elements, while the server is responsible to receive remote topology elements from the other clusters.

6.5. Routing and Scalability

With a complete view of the network topology, routing becomes a problem of graph searching; the ONOS *PathManager* exports the proper APIs to the northbound applications. Destinations addressing is a matter of use-case scenario; in our prototype implementation the endpoints are identified by L2 addresses and the infrastructure devices are Openflow devices, but this does not affect the generality of the system because, as mentioned in 6.1., the main functional modules like the admission control routine are protocol-agnostic. Currently the topology discovery is implemented by a full-mesh communication between clusters of ONOS and each cluster only advertises the destinations that are directly connected to the local devices, thus avoiding to implement a distance vector or link state routing protocol. Every instance of a cluster handles the peering with a subset of all the other clusters through the *LeadershipService* of ONOS in order to load-balance the number of peering connections among the ONOS instances.

7. ALGORITHM COMPUTATION TIME EVALUATION

The purpose of this section is to report some benchmark results on the scalability performance of the control plane of our prototype implementation.

Table 1. Single-request computation time

	$N = 50, p_l = 0.75$	$N = 100, p_l = 0.75$	$N = 450, p_l = 0.05$	$N = 500, p_l = 0.05$
Response time (ms)	6.392	9.109	201.842	331.055

The resource reservation process overhead within a single domain is proportional to:

$$D_{app,ctl} + T_{algo} + \max_{n \in N} \{D_{ctl,n}\}$$

where:

- $D_{app,ctl}$: latency between the application and the controller
- T_{algo}: admission control computation time
- $D_{ctl,n}$: latency between the controller and the n_{th} device

The overhead when considering the endpoints placed in M domains is proportional to:

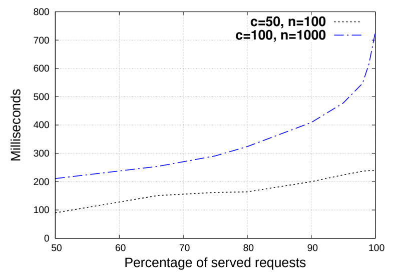
$$D^i_{app,ctl} + T^i_{algo} + \max_{j \in M \setminus i} \{D_{ij} + T^j_{algo}\} + \max_{n \in N, j \in M} \{D^j_{ctl,n}\}$$

where D_{ij} is the latency between domain controller i and j. In [36] we show that splitting the control plane into multiple clusters improves the event-to-response reactivity by decreasing the $D_{ctl,n}$ term at the expense of adding communication overhead between clusters. Splitting the control-plane into multiple regions also decrease the algorithm computation time that here we benchmark within a single region.

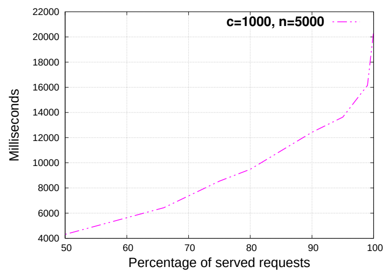
The overhead is reported against increasing number of concurrent HTTP requests and increasing topology dimension for a single instance of an ONOS cluster running on a bare metal HP EliteDesk 800 G1 SFF with Intel Core i7-4770 CPU @ 3.40GHz, 16 GB RAM. Random topologies are generated by assigning a probability p_l of link existence between any two nodes (Bernoulli model) and injected into the ONOS core database. N is the number of infrastructure devices, c the concurrency level and n the total number of requests per experiment. For $N = 450$ and $N = 500$ we exploit the limit theorem according to which the probability that a Bernoulli random graph is fully connected is distributed as $1 - N(1 - p_l)^{(N-1)}$. The thread pool is configured to use up to a maximum of 300 threads. The computation time T_{algo} is assumed to start as soon as the request arrives to the REST applet of the manager application until the drivers for setting up the queues are called. We use a modified version of algorithm 1 in which the set of candidate paths is chosen with the Dijkstra algorithm for a maximum of five shortest paths, using a link cost function that forces to infinite the cost of the direct link between two endpoints, if present, while the cost of all the other links is uniformly distributed between zero and one. The endpoints for each single request are also randomly generated so are the network parameters for the admission control formula. The T_{algo} overhead even for networks with hundreds of nodes (and links and hosts) is of the order of milliseconds (Table 1); in Figure 9(e) 50% of the requests, at the origin point, are served within one second. However, we encountered serious problems in processing requests with a concurrency level of 1000 connections with N equal to 450 and 500 so much so that we decided to not report the numbers. This inefficiency is intrinsic to the ONOS controller that showed a greedy cpu usage of some hundreds percentage during the tests, due to possibly unnecessary operations on the simulated network elements. Nevertheless, note that T_{algo} includes the collection of candidate paths, the transactional allocation process on the bandwidth resource on each link of any scanned path, the collection and the access to the device drivers for setting-up the queues. The high values reported in 9(e) 9(f) with a concurrency level of one hundred are given by the failure in the transactional operation for bandwidth allocation due to possible collisions among the requests. This is certainly a point of investigation and the prototype application and control framework would need an engineering effort in terms of scalability performance for a production-ready application.

8. CONCLUSION

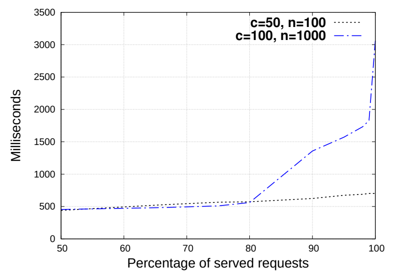
In this article we presented a resource reservation scheme for end-to-end QoS provisioning. We analysed all the essentials aspects of the framework application running on top of a centralised network controller: the admission control, the inter-domain communication required to achieve the end-to-end guarantee, the interaction with the core controller components and the employment of software drivers to decouple the functional intents from the



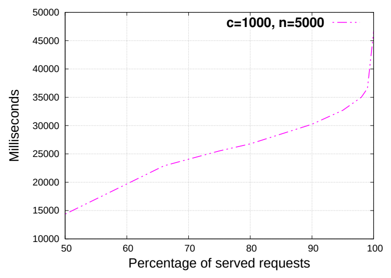
(a) $N = 50, p_l = 0.75$



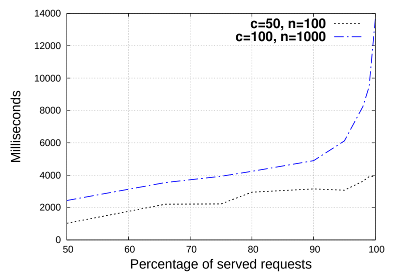
(b) $N = 50, p_l = 0.75$



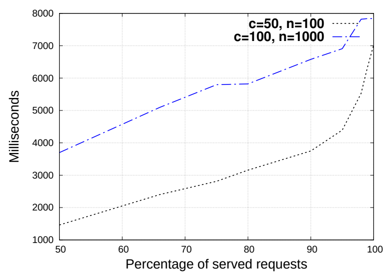
(c) $N = 100, p_l = 0.75$



(d) $N = 100, p_l = 0.75$



(e) $N = 450, p_l = 0.05$



(f) $N = 500, p_l = 0.05$

Figure 9. Percentage of requests served within a certain amount of time

device-specific traffic control rules. A lot remains to be explore though: the integration between the flow-based and class-based QoS within the controller, the automation of a policy-driven mechanism to enable dynamic SLA and a investigation focusing exclusively on the communication design between domains. We believe that the adoption of centralised controllers orchestrating the infrastructure devices could enable guaranteed services to span multiple domains and wide-area-networks, opening the doors to the fruition of yet-to-be-seen real-time and tactile applications over the next-generation communication networks.

REFERENCES

[1] G. P. Fettweis, "The tactile internet: Applications and challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, March 2014.

[2] X. Xiao and L. M. Ni, "Internet qos: A big picture," *Netwrk. Mag. of Global Internetwkg.*, vol. 13, no. 2, pp. 8–18, Mar. 1999. [Online]. Available: http://dx.doi.org/10.1109/65.768484

[3] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "Onos: Towards an open, distributed sdn os," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14. New York, NY, USA: ACM, 2014, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/2620728.2620744

[4] D. H. Lorenz and A. Orda, "Qos routing in networks with uncertain parameters," *IEEE/ACM Transactions on Networking*, vol. 6, no. 6, pp. 768–778, Dec 1998.

[5] R. Guerin and A. Orda, "Qos based routing in networks with inaccurate information: theory and algorithms," in *INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., Proceedings IEEE*, vol. 1, Apr 1997, pp. 75–83 vol.1.

[6] P. Goyal, S. S. Lam, and H. M. Vin, "Determining end-to-end delay bounds in heterogeneous networks," in *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, ser. NOSSDAV '95. London, UK, UK: Springer-Verlag, 1995, pp. 273–284. [Online]. Available: http://dl.acm.org/citation.cfm?id=646982.712176

[7] D. C. Verma, H. Zhang, and D. Ferrari, "Delay jitter control for real-time communication in a packet switching network," in *Proceedings of TRICOMM '91: IEEE Conference on Communications Software: Communications for Distributed Applications and Systems*, Apr 1991, pp. 35–43.

[8] H. Jia and Z. Jinhe, "The design of finegrained network qos controller and performance research with network calculus," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 12, no. 6, pp. 4468–4474, 2014. [Online]. Available: http://iaesjournal.com/online/index.php/TELKOMNIKA/article/view/5484

[9] A. R. Bashandy, E. K. P. Chong, and A. Ghafoor, "Generalized quality-of-service routing with resource allocation," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 450–463, Feb 2005.

[10] X. Yuan and X. Liu, "Heuristic algorithms for multi-constrained quality of service routing," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, vol. 2, 2001, pp. 844–853 vol.2.

[11] R. G. Garroppo, S. Giordano, and L. Tavanti, "A survey on multi-constrained optimal path computation: Exact and approximate algorithms," *Comput. Netw.*, vol. 54, no. 17, pp. 3081–3107, Dec. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2010.05.017

[12] F. Kuipers, P. V. Mieghem, T. Korkmaz, and M. Krunz, "An overview of constraint-based path selection algorithms for qos routing," *IEEE Communications Magazine*, vol. 40, no. 12, pp. 50–55, Dec 2002.

[13] A. Juttner, B. Szviatovski, I. Mecs, and Z. Rajko, "Lagrange relaxation based method for the qos routing problem," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, vol. 2, 2001, pp. 859–868 vol.2.

[14] H. Zang, J. P. Jue, B. Mukherjee *et al.*, "A review of routing and wavelength assignment approaches for wavelength-routed optical wdm networks," *Optical Networks Magazine*, vol. 1, no. 1, pp. 47–60, 2000.

[15] L. Hui, "A novel qos routing algorithm in wireless mesh networks," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 11, no. 3, pp. 1652–1664, 2013. [Online]. Available: http://iaesjournal.com/online/index.php/TELKOMNIKA/article/view/2321

[16] B. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol (rsvp) – version 1 functional specification," Internet Requests for Comments, RFC Editor, RFC 2205, September 1997, http://www.rfc-editor.org/rfc/rfc2205.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2205.txt

[17] J. Wroclawski, "The use of rsvp with ietf integrated services," Internet Requests for Comments, RFC Editor, RFC 2210, September 1997, http://www.rfc-editor.org/rfc/rfc2210.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2210.txt

[18] S. Blake, D. L. Black, M. A. Carlson, E. Davies, Z. Wang, and W. Weiss, “An architecture for differentiated services,” Internet Requests for Comments, RFC Editor, RFC 2475, December 1998, http://www.rfc-editor.org/rfc/rfc2475.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2475.txt

[19] Tech. Rep.

[20] E. Rosen and Y. Rekhter, “Bgp/mpls ip virtual private networks (vpns),” Internet Requests for Comments, RFC Editor, RFC 4364, February 2006.

[21] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, and S. Molendini, “Rsvp refresh overhead reduction extensions,” Internet Requests for Comments, RFC Editor, RFC 2961, April 2001.

[22] S. Shenker and L. Breslau, “Two issues in reservation establishment,” *SIGCOMM Comput. Commun. Rev.*, vol. 25, no. 4, pp. 14–26, Oct. 1995. [Online]. Available: http://doi.acm.org/10.1145/217391.217403

[23] S. Shenker, C. Partridge, and R. Guerin, “Specification of guaranteed quality of service,” Internet Requests for Comments, RFC Editor, RFC 2212, September 1997, http://www.rfc-editor.org/rfc/rfc2212.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2212.txt

[24] D. Mitra, “Stochastic theory of a fluid model of producers and consumers coupled by a buffer,” *Advances in Applied Probability*, vol. 20, no. 3, pp. 646–676, 1988. [Online]. Available: http://www.jstor.org/stable/1427040

[25] S. Ahn and V. Ramaswami, “Fluid flow models and queuesâĂŤa connection by stochastic coupling,” *Stochastic Models*, vol. 19, no. 3, pp. 325–348, 2003. [Online]. Available: http://dx.doi.org/10.1081/STM-120023564

[26] M. Alasti, B. Neekzad, J. Hui, and R. Vannithamby, “Quality of service in wimax and lte networks [topics in wireless communications],” *IEEE Communications Magazine*, vol. 48, no. 5, pp. 104–111, May 2010.

[27] J. Costa-Requena, “Sdn integration in lte mobile backhaul networks,” in *The International Conference on Information Networking 2014 (ICOIN2014)*, Feb 2014, pp. 264–269.

[28] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, “The design and implementation of open vswitch,” in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. Oakland, CA: USENIX Association, May 2015, pp. 117–130. [Online]. Available: https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/pfaff

[29] A. Farrel, J.-P. Vasseur, and J. Ash, “A path computation element (pce)-based architecture,” Internet Requests for Comments, RFC Editor, RFC 4655, August 2006, http://www.rfc-editor.org/rfc/rfc4655.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4655.txt

[30] J. Vasseur and J. L. Roux, “Path computation element (pce) communication protocol (pcep),” Internet Requests for Comments, RFC Editor, RFC 5440, March 2009, http://www.rfc-editor.org/rfc/rfc5440.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5440.txt

[31] K. Yoshihara, M. Isomura, and H. Horiuchi, “Distributed policy-based management enabling policy adaptation on monitoring using active network technology,” 2001. [Online]. Available: http://proceedings.utwente.nl/23/

[32] L. Lymberopoulos, E. Lupu, and M. Sloman, “An adaptive policy-based framework for network services management,” *J. Netw. Syst. Manage.*, vol. 11, no. 3, pp. 277–303, Sep. 2003. [Online]. Available: http://dx.doi.org/10.1023/A:1025719407427

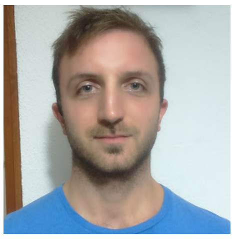
[33] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, “Policycop: An autonomic qos policy enforcement framework for software defined networks,” in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Nov 2013, pp. 1–7.

[34] “Onos website - http://onosproject.org/.”

[35] “Apache karaf - http://karaf.apache.org/.”

[36] M. Gerola, F. Lucrezia, M. Santuari, E. Salvadori, S. S. P. L. Ventre, and M. Campanella, “Icona: a peer-to-peer approach for software defined wide area networks using onos,” in *European Workshop on Software Defined Networks (EWSDN)*, October 2016.

BIOGRAPHIES OF AUTHORS

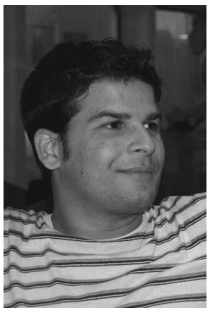


Francesco Lucrezia is a PhD student at Politecnico di Torino, Italy. He received his Bachelor Degree at the University of Pavia in Computer Science (20111) and his M.S. at Politecnico di Torino in 2013 in Computer and Communication Networks. His research interests include QoS and QoE models for the Internet, traffic characterization, Software-Defined-Networking and network control-plane architectures.

Guido Marchetto is an assistant professor at the Department of Control and Computer Engineering of Politecnico di Torino. He got his Ph.D. in Computer Engineering in April 2008 from Politecnico di Torino. His research topics cover innovative network protocols and network architectures.

Fulvio Risso (Ph.D. in Computer Engineering) is Assistant Professor at at Politecnico di Torino, Italy. His research interests focus on high-speed and flexible network processing, software defined networks, network functions virtualization. He started and led several open-source software projects including WinPcap, the de-facto library for capturing and analysing traffic on Windows, and NetBee, a novel packet library for high speed and flexible traffic processing. Fulvio is author of more than 60 papers, mostly focused on high speed and flexible network processing.

Michele Santuari Michele Santuari received the Master degree on Telecommunications Engineering in 2014 from the University of Trento, with the thesis: "An OpenFlow architecture to improve traffic management in enterprise edge networks". In 2013, he joined CREATE- NET as a junior research engineer and software developer within the Smart Infrastructure application area. In 2016, he joined the Future Network area as a research engineer. He is focused mainly on development actives related to Software Defined Network, in particular on the Control Plane and multi-layer and multi-domain orchestration. He contribute to open source communities e.g., Open Network Operating System (ONOS), OpenStack.



Matteo Gerola Matteo Gerola is a software architect and senior research engineer at Future Networks unit at CREATE-NET research center. His main research interests focus on SDN, Network Virtualization, OpenFlow and Optical Networks. Within CREATE-NET he has been involved in several European projects on SDN, optical technologies, and Future Internet test-beds. He has published in more than 20 International refereed journals and conferences. Along the years, he has participated to events, conferences and workshops as TPC member, author and invited speaker.