An Interactive Intent-based Negotiation Scheme for Application-Centric Networks

Antonio Marsico, Michele Santuari,
Marco Savi, Domenico Siracusa
FBK CREATE-NET, Trento, Italy

Abdul Ghafoor, Stéphane Junique,
Pontus Sköldström
RISE ICT/Acreo, Kista, Sweden

***Abstract*—The demonstration presents the first implementation of a resource negotiation scheme between users and a network for the provisioning of application-aware connectivity services. This active interaction enables the users, who request connectivity services with multiple application requirements, to select an alternative solution when the network does not have enough resources to satisfy the original requests.**

***Index Terms*—Software-Defined Networking; ONOS; Resource Negotiation.**

I. Introduction

The *application-centric networking* paradigm proposes a novel provisioning process to differentiate connectivity services based on application-specific requirements. Every application is characterized by a multitude of requirements, such as bandwidth, latency, availability, etc., defining the type of connectivity Service Request (SR) that the network should satisfy. These SRs can be communicated to an SDN orchestrator by exploiting an intent-based interface, where an *intent* is a human-readable abstracted policy used to specify the application requirements. The SDN orchestrator evaluates the application SR based on the current network state and tries to provision an *application-aware* path, satisfying all the application requirements.

In a previous work [1], we showed that considering multiple requirements during *service provisioning* may lead to a high SR blocking probability in the case of high network load conditions. The authors of [2] shows that lower SR blocking probabilities can be obtained by performing a degradation of the bandwidth requirement for the provisioned services. This solution, however, does not consider other application requirements, and the service degradation is enforced unilaterally by the network.

In this work, we introduce the concept of *negotiation* of application SRs, which offers the possibility to find a common agreement between users deploying applications and the underlying network for the provisioning of application-aware traffic flows. The proposed demonstration shows how a user can interact with the SDN orchestrator when an SR meeting all the specified requirements cannot be found. In this case, the SDN orchestrator offers several Alternative Solutions (AS) with downgraded/upgraded application requirements and the user can choose his/her preferred one, which will be finally installed by the SDN orchestrator.

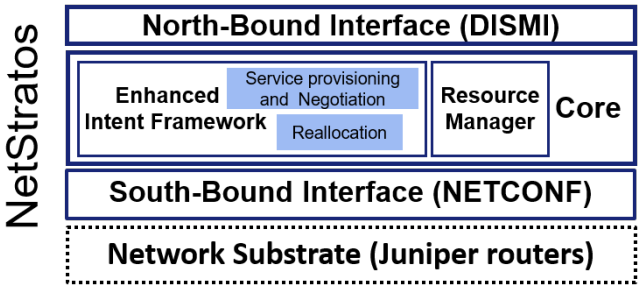


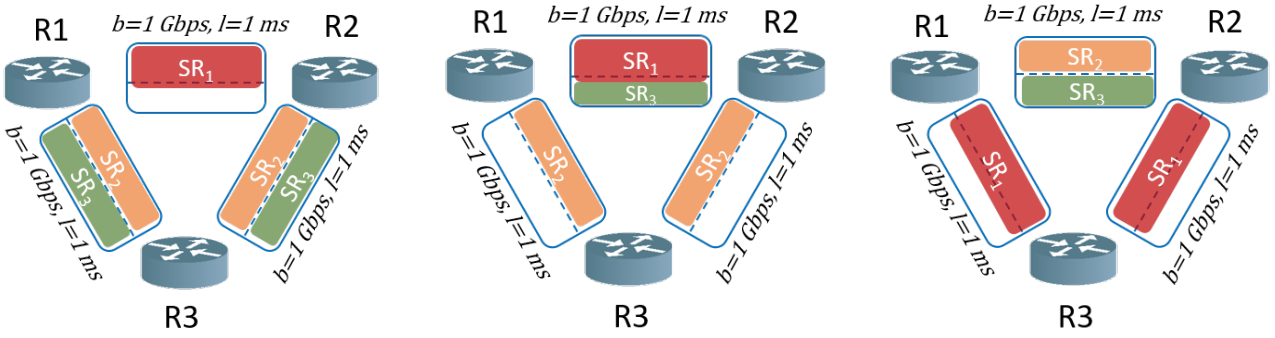
Fig. 1. SDN orchestrator architecture.

II. Architecture

The system architecture is depicted in Fig. 1. We proposed NetStratos as SDN orchestrator, which relies on the ONOS SDN controller [3].

The intent-based North-Bound Interface is provided by DISMI [4] to handle application SRs. A GUI client is provided to the users for interacting with DISMI via an HTTP RESTful API. The ONOS intent framework has been extended to implement the application-aware *service provisioning and negotiation* algorithm. This algorithm relies both (i) on an enhanced version of the ONOS Intent Framework and (ii) on the ONOS Resource Manager, which provides a database that maintains the information about network resources (e.g., the bandwidth of a link, latency, etc.) and their usage (i.e., which intent is allocated on which links and the respective amount of consumed resources). When the user communicates a new SR through DISMI, the ONOS Intent Framework tries to find a path that meets all the application requirements by executing the application-aware service provisioning algorithm and exploiting the ONOS Resource Manager. If a path with the given requirements cannot be found, instead of blocking the SR, the negotiation algorithm tries to find several ASs based on (i) looser application requirements (e.g. a lower bandwidth and/or a higher latency) or on (ii) a *reallocation* of already-provisioned SRs to satisfy the original SR.

However, the reallocation of SRs may imply a disruption of already-provisioned services that do not have high availability constraints, while the others must always be migrated in a hitless manner. This operation thus comes with a cost for the network, since some users will experience some connectivity disruption and the hitless reallocation requires additional resources to be temporarily allocated in the network (i.e., by following the make-before-break paradigm). Therefore, we



(a) AS_1: Higher latency (b) AS_2: Lower bandwidth (c) AS_3: Upgraded availability (reallocation)

Fig. 2. Set of ASs proposed by the negotiation algorithm for the provisioning of SR_3.

extend the ONOS Intent Framework with a non-disruptive reallocation mechanism to avoid any connectivity downtime of a SR under a high-availability constraint.

As substrate network, we consider a real testbed composed of 3 Juniper routers MX240, in a triangular topology configuration. The South-Bound Interface uses NETCONF as communication protocol, and implements an extension to support Juniper devices. The testbed links have a fixed capacity of 1 Gbit/s and a latency of ∼1 ms.

III. Demonstration

In our scenario, the user can create via the client GUI an application SR. For the sake of clarity, we refer to a DISMI SR as a tuple $SR = \{s, d, b, l, ha\}$, where s and d represent the connection endpoints, b the requested bandwidth, l the maximum allowed latency, and ha the high-availability flag (i.e., it informs the network whether a SR requires high availability, $ha = true$, or not). Two SRs have already been provisioned in the network. They are defined as $SR_1 = \{R1, R2, 0.6\ Gbps, \infty, true\}$ and $SR_2 = \{R1, R2, 0.5\ Gbps, \infty, false\}$, where ∞ means that the latency is not constrained. SR_1 has been provisioned on the shortest path between $R1$ and $R2$, while SR_2 on the longest path.

The demonstration is performed in three steps as follows:

1) The user requests a new SR via the provided GUI to the SDN orchestrator. The new service request is defined as $SR_3 = \{R1, R2, 0.5\ Gbps, 1\ ms, false\}$. However, the current network status does not offer the possibility to allocate the new SR, because between R1 and R2 (i) the longest path offers 2 ms of latency, thus, it does not meet the latency requirement of SR_3 and (ii) the shortest path satisfies the latency requirement but not the bandwidth one, since the available capacity is $0.4\ Gbps$ only.
2) The service provisioning and negotiation algorithm provides two ASs based on the degradation of application requirements, as shown in Fig. 2a-2b. The former offers $AS_1 = \{0.5\ Gbps, 2\ ms, false\}$, while the latter $AS_2 = \{0.4\ Gbps, 1\ ms, false\}$. In addition, the algorithm provides a third AS (AS_3) that does not require any requirement degradation, since a reallocation of SR_1 and SR_2 would allow the provisioning of the original SR. Unlike SR_1, SR_2 does not have a high-availability constraint, so the algorithm can swap the paths between SR_1 and SR_2 (Fig. 2c) with a hitless migration of SR_1 and a temporary disruption of SR_2. SR_1 cannot be migrated in a hitless way without disrupting SR_2 because on the longest path there is not enough spare capacity to accommodate it. However, once SR_2 is disrupted and SR_1 is migrated on the longest path, enough resources are freed on the shortest path to both reallocate SR_2 and provision SR_3. AS_3 is offered to the user with an *upgrade* in the high-availability constraint: $AS_3 = \{0.5\ Gbps, 1\ ms, true\}$. This choice is made by the network as an additional reward for the reallocation effort (i.e., SR_2 disruption), since stricter requirements imply a higher price for users.
3) The user evaluates on the GUI the new proposed solutions and he/she interactively chooses the one that better suits its needs. Finally, the SDN orchestrator modifies the network configuration accordingly.

IV. Conclusion

This work demonstrated a bi-directional negotiation scheme between users and an SDN orchestrator for the provisioning of application-aware service requests. In the case of lack of network resources, the SDN orchestrator offers several alternative solutions, based on a downgrade/upgrade of the specified requirements, to the users. They, they can choose the preferred solution and avoid the blocking of the request.

Acknowledgments

This work has received funding from the EU H2020 Research and Innovation program, ACINO project, grant agreement no. 645127.

References

[1] M. Savi *et al.*, "An Application-Aware Multi-Layer Service Provisioning Algorithm based on Auxiliary Graphs," in *Proceedings of the OFC Conference*, Los Angeles, USA, 19-23 Mar. 2017.
[2] Z. Zhong *et al.*, "On QoS-Assured Degraded Provisioning in Service-Differentiated Multi-Layer Elastic Optical Networks," in *Proceedings of IEEE GLOBECOM Conference*, Seoul, South Korea, 6-10 Jun. 2016.
[3] P. Berde *et al.*, "ONOS: Towards an Open, Distributed SDN OS," in *ACM HotSDN*, Chicago, USA, 22 Aug. 2014.
[4] P. Sköldström *et al.*, "DISMI - An intent interface for application-centric transport network services," *Proceedings of the ICTON 2017 Conference*, 2017, to appear.