Hard-isolation for Network Slicing

Nicolas Huin, Paolo Medagliani, Sébastien Martin, Jérémie Leguay,
Lei Shi, Shengming Cai, Jinchun Xu, Hao Shi
Huawei Technologies Co. Ltd.
Email: {firstname.lastname}@huawei.com

***Abstract*—The fifth generation of mobile networks (5G) promises to improve network utilization by allowing operators to create virtual networks for their tenants with different Quality of Service (QoS) requirements. This technology is also referred to as *network slicing*. To guarantee physical isolation for virtual networks and no interference between different slices, it is possible to rely on the Flex Ethernet (FlexE) technology. It isolates different slices by allocating resources in a TDMA-fashion.**

In this demo, we will present our algorithmic framework, based on a Column Generation routine, and we will showcase how the configuration of FlexE interfaces to guarantee hard isolation between slices.

Index Terms*—*Resource Allocation, Network Slicing, Network Virtualization, Combinatorial Optimization.

I. Introduction

"Slicing" a network means creating virtual networks with different SLA requirements, operated by slice tenants, on top of a common physical network [1]. Virtual links and virtual nodes can be easily established by an Software Defined Network (SDN) controller or network orchestrator [2].

In order to guarantee data isolation between virtual networks, it is possible to provide either *soft* or *hard* slicing. While the former approach, presented in [3] is easier to deploy, it provides no guarantees on the status of the network when one of the slices experiences a heavy load. The latter fixes this problem by providing a stricter subdivision of the network. One of the emerging standards for hard slice isolation is Flex Ethernet, that leverages on a TDMA-like resource subdivision between different slices [4].

Hard isolation is expected to become a key enabler for 5G, especially for uRLLC use case, as it can guarantee almost deterministic performance in terms of E2E latency due to the absence of collisions between services of different slices [5].

In this demo, we show how we can reserve resources for a network slice requesting hard isolation on top of a physical network. The algorithmic framework, based on column generation [6], allows to configure the selected FlexE interfaces to guarantee hard slicing, while respecting QoS constraints.

II. Slicing constraints

Technology constraint The FlexE standard introduces a new constraint on the different resource reservation policy that must be taken into account in the modelization of the optimization problem. According to [4], the bandwidth of a link is reserved in slots with a granularity of 5 Gb. This means that when a given amount of bandwidth is required on a link, enough slots must be activated to provide at least the given capacity. However, if some bandwidth remains available in the same slot, it can be used by other demands of the same slice. For instance, two services of 7 Gb (S1) and 3 Gb (S2) require to use the same link. For S1, 2 slots of 5Gb must be activated for a total of 10 Gb. As 3 Gb are available service S2 can be routed through the same link, without the need to allocated more resources.

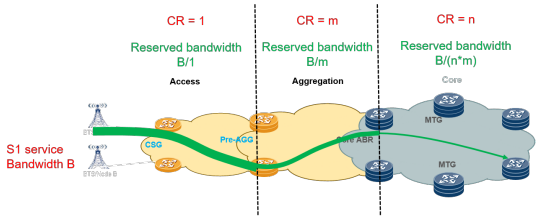

Fig. 1: Convergence ratio (CR) in IPRAN networks.

Network constraint The FlexE technology is expected to be used in IPRAN networks, which are composed by an access network, an aggregation network, and a core network. In this type of network, we assume that some services passing through the aggregation and the core network will not be active at the same time. For this reason, it is possible to statistical multiplex the reserved resources, i.e., we scale down the reserved bandwidth by a multiplicative factor, as shown in Fig 1, referred to as *convergence ratio* (CR). However, we must also guarantee that when a single service is active enough bandwidth is reserved on each used link.

III. Algorithm

In order to efficiently solve the resource reservation problem, we developed an algorithm based on a Column Generation (CG) routine, as shown in Fig. 2. The CG routine can be warmstarted by providing a first initial feasible solution, for instance, by using a greedy algorithm. In the CG routine, the master and the pricing problem are solved iteratively until an optimal, but relaxed, solution is found. The FlexE constraints are implemented into the master problem, while in the pricing we consider QoS constraints (i.e., latency and 1+1 protection). In the rounding step, we run in parallel several randomized rounding routines to ensure that the final solution will be integer and feasible. Finally, the best solution among those provided in the rounding step is selected.

This algorithm can be implemented inside an SDN controller to reserve FlexE resources over physical links in order to satisfy the QoS requirements of a slice.

IV. Automatic Slice Creation

The slice creation and deployment steps that follow the issue of a slice request, as shown in Fig. 3, are the following:

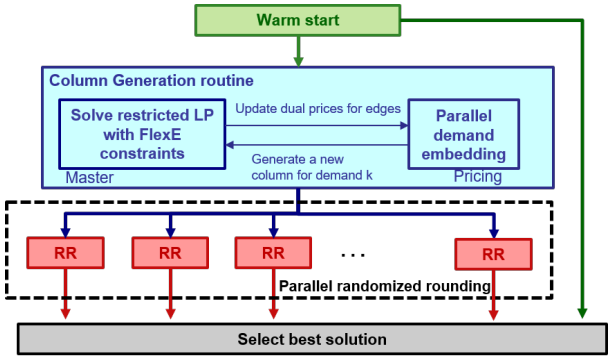

Fig. 2: Algorithmic framework to solve the resource reservation problem

1) **Slice Planning**: given inputs on slice demands and QoS requirements, the virtual topology necessary for the slice is computed by the algorithm presented in Section III.

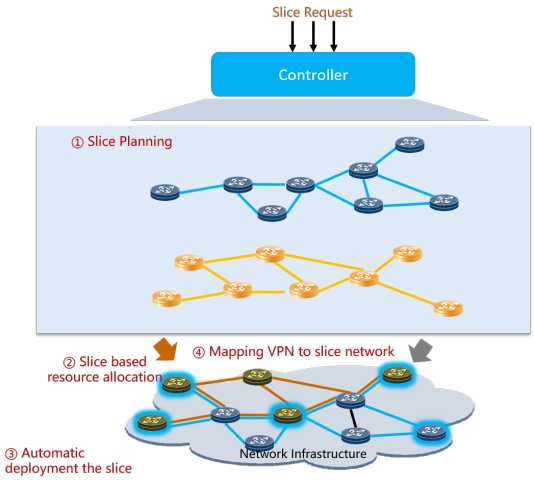

Fig. 3: SDN controller creating and deploying a slice.

2) **Resource Allocation**: Once the virtual slice has been created, the physical nodes used by the slice are instructed by the SDN controller to reserve the dedicated FlexE resources to the slice. The operations done by the controller in this phase are: (i) enabling of the FlexE interface, (ii) allocation of a FlexE group number, (iii) creation of the FlexE subinterface, and (iv) configuration of the bandwidth, allocation of the clientId, and association of the subinterface to the FlexE group.
3) **Automatic Slice Deployment**: After the resources have been reserved, an IP address is assigned to each interface, the IGP metrics and FlexE interfaces are configured. The SDN controller collects this information and distributes the SR policies to perform in-slice routing and to guarantee isolation between slices.
4) **Mapping VPN to slice network**: Once the virtual network is configured, the service paths calculated by the algorithmic framework are deployed to the network and the desired traffic policies are applied (i.e., the SR labels are associated to each service).

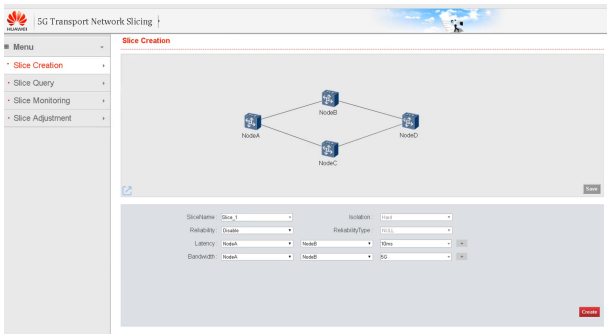

Fig. 4: Interface of the slice controller.

V. Demo

In this demo, we will first show how two slices can be created and monitored by using the controller interface presented in Fig. 4. Then, we will show how the hard isolation between two slices allows to guarantee that the performance of one slice are not impacted by the other misbehaving slice.

The network is composed by 4 Huawei NE40 routers interconnected via 10 GB links. Traffic is generates using the Tesgine 2.0 traffic generator.

The demonstrated steps are the following:
1) Two slices are created: Slice 1 requires to reserve 5GB links from Node A to Node D with protection and Slice 2 requires to reserve 5GB from Node A to Node D without protection. The virtual networks are automatically deployed on top of the physical network.
2) For Slice 1, a 3 GB service is created. The requested bandwidth is below the capacity of the links reserved for the slice. For Slice 2, a 5 GB service is created. Considering the overhead introduced by SR, this second service exceeds the capacity of the link reserved for Slice 2.
3) The slice monitoring interface allows to see that for Slice 1, the network performance is good as the average latency is 0.4 ms and there is no packet loss. For Slice 2, instead, 5% of the packet are lost and the average delay is 21.57 ms. The fact that Slice 1 is not affected by the poor behavior of Slice 2 proves that FlexE manages to guarantee hard isolation between slices. The video of the demo is available at https://drive.google.com/open?id=1kPH5AAMOQWeAwfIKNxiAiiMXTI5Zruuv

References

[1] 5G Service-Guaranteed Network Slicing White Paper. Huaweis whitepaper, February 2017.
[2] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, Jan 2015.
[3] A. Destounis, G. Paschos, S. Paris, J. Leguay, L. Gkatzikis, S. Vassilaras, M. Leconte, and P. Medagliani. Slice-based column generation for network slicing. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–2, April 2018.
[4] Flex Ethernet 2.0 Implementation Agreement, url=https://www.oiforum.com/wp-content/uploads/OIF-FLEXE-02.0.pdf, year=2018, month=June.
[5] Network Slicing Architecture, url=https://tools.ietf.org/id/draft-geng-netslices-architecture-02.html, year=2018, month=January.
[6] G. Desaulniers, J. Desrosiers, and M.M. Solomon. *Column Generation*. GERAD 25th anniversary series. Springer US, 2006.