Network Slicing for Deterministic Latency

Sebastien Martin, Paolo Medagliani, Jeremie Leguay
Huawei Technologies, Paris Research Center, France.

Abstract—**Deterministic performance is a key enabler for 5G applications. While specific data-plane solutions have been proposed to reach a low deterministic end-to-end latency and jitter, legacy round-robin schedulers can already be used to guarantee bounds on the end-to-end latency, when associated with per-flow shapers. In this context, we propose a latency-guaranteed network slicing solution that trades-off between complexity and performance. We propose control plane algorithms to configure sub-channelized interfaces with an independent QoS scheduler at each physical port used by a slice. The algorithms allocate service rates and decide about queue assignments and routing inside each slice. Through numerical results on large network topologies, we demonstrate that our column-generation and two-steps algorithms can improve traffic acceptance while reducing the amount of reserved capacity.**

Index Terms—**Deterministic networks, Bounded delay, Deficit Round Robin, Network slicing.**

I. Introduction

The 5th generation of networks is paving the road for latency-sensitive network services to enable a wide-range of Internet applications like factory automation, connected vehicles and smart grids [1]. Traditional Internet Protocol (IP) services allow reliable data delivery, but they cannot provide strict Quality of Service (QoS) guarantees. Certain classes of service can get preferential treatment but performance remains statistical. To frame the development of new technologies for deterministic IP networks, the IETF [2] and the ITU-T [3] have specified target requirements for guaranteed bandwidth, bounded End-to-End (E2E) latency and bounded jitter.

In the industrial domain, technologies such as Deterministic IP (DIP) [4] and Time-Sensitive Networking (TSN) [5] are gaining importance as they can guarantee a very low deterministic end-to-end latency and almost 0 jitter to high priority (HP) flows. They rely on deterministic data plane solutions such as Cyclic Queuing and Forwarding (CQF) [6]. Resources are statically assigned at packet level, even when the network utilization is low. In this way, the delay experienced by each flow only depends on its characteristics (e.g., maximum burst size) and its allocated bandwidth in the different transmission cycles. The drawback of these solutions is that they require specific network interfaces to support deterministic forwarding (e.g., using CQF) in the data plane.

In more traditional IP networks, applications such as virtual reality or online games may ”only” require a strictly bounded end-to-end latency, instead of a ultra-low latency with 0 jitter. These applications may have different latency requirements and co-exist with best effort (BE) traffic. In this context, legacy schedulers such as weighted Fair Queuing (WFQ) [7] and Deficit Round Robin (DRR) [8], widely available in network chipsets, can be used to share the capacity between applications, satisfying the latency requirements of high priority flows and avoiding the starvation of the best effort traffic. In the case of round-robin schedulers (e.g., DRR), the latency experienced by a flow depends on his arrival curve (i.e., maximum burst size and rate), the arrival curves of the flows it interferes with, and the service rates of queues it traverses. Thanks to network calculus models [9], [10], worst case estimations can be derived for the end-to-end latency of each flow. Although these models can be complex when deriving tight latency bounds, they can simplify for more conservative estimations and become tractable for network optimization.

To support different latency requirements on top of the same physical network, different “slices” or virtual networks, can be created in order to ensure traffic isolation [11]. Technologies such as channelized sub-interfaces [12] can be used to allocate dedicated capacity to each slice on each port. In this case, each slice can have its own QoS scheduler at each port to decide how the overall allocated capacity is shared between its virtual services. The most straightforward way to ensure a low latency is to over-provision the reserved capacity for each slice. In this way, the network operates in low-load regime, where the latency performance is almost constant and known. However, to improve the utilization of bandwidth, a more accurate knowledge of the delay experienced is required. Leveraging on latency bounds given by network calculus, the network controller can assign resources to each slice, in order to respect the QoS requirement of each virtual service, while ensuring that a maximum of bandwidth is left for future slices.

In this paper, we focus on the design of network slices for deterministic latency guarantees. Our architecture is based on legacy round-robin schedulers and per-flow shapers inside channelized sub-interfaces allocated to each slice at each port. We show that this solution well trades-off between complexity and the strict respect of end-to-end latency. In order to optimize bandwidth utilization, we introduce centralized control plane algorithms that embed a network calculus model to decide the minimum amount of capacity that needs to be allocated to each slice. From a traffic matrix with arrival curves of virtual services and the available network capacity, the algorithm decides about the service rate of each queue of the sub-interfaces and the routing inside the slice. We study two variants of the slicing problem, a simpler one where a bounded delay is given and must be respected for each queue (called *bounded delay*), and a more complex one where the maximum delay of queues is decided (called *variable delay*). For the first case, we present a column generation algorithm that solves the underlying multi-commodity flow problem. In the variable delay case, as the model is non-linear, we propose a two-steps algorithm that alternatively decides about routing and rate allocation. We show through numerical results on a large IPRAN network that the column generation improves traffic acceptance and decreases the total reserved capacity compared to a greedy algorithm. We also show that when the maximum queuing delay is decided by the two-steps algorithm, we further increase acceptance and decrease bandwidth utilization.

The rest of this paper is structured as follows. Sec. II reviews the state of the art. The system architecture and the slicing problem are introduced in Sec. III, while Sec. IV presents the mathematical formulation of the problem. Sec. V introduces heuristics algorithms for the control plane. Sec. VI shows the performance evaluation and Sec. VII concludes this paper.

II. Related work

To guarantee traffic isolation between slices, different data plane technology can be used: from *soft* slicing [13] with traditional QoS and VPN technologies, where resources are returned to the network after their utilization, to *hard* slicing with technologies like channelized sub-interfaces [12] or Flex Ethernet [14] that leverage on dedicated capacity assignment to different tenants for strict isolation. In Sec. III, we will present an architecture based on channelized sub-interfaces that can actually work also for Flex-Ethernet simply by considering a different slotted allocation of bandwidth [15].

Approximate latency models have already been embedded into network design or routing problems. Fortz *et al* considered the piecewise linear unsplittable multi-commodity flow problem [16], [17] where the cost of links is inversely proportional to the link utilization. Ben Ameur *et al* [18] considered the Kleinrock function [19] and gave a convex relaxation to compute a lower bound of the (fractional) routing problem with minimum linear cost. While these works can be used to decrease latency, they do not intend to accurately capture the end-to-end delay and provide strict guarantees.

In the past decade, a collection of IEEE 802.1 Ethernet standards, known as Time-Sensitive Networking (TSN) [5], has been developed to support professional applications over Local Area Networks (LAN) with layer-2 mechanisms such as priority queuing, preemption, traffic shaping, and time-based opening of gates at output ports. To improve the scalability of TSN solutions, the IETF DetNet (Deterministic Networking) [20] group has been working on the Large-scale Deterministic Network (LDN) [21] architecture. While these solutions can ensure a small and bounded latency with almost 0 jitter, they require specific network interfaces to support deterministic forwarding (e.g., using CQF) in the data plane. Advanced control plane algorithms have been proposed to decide about routing and scheduling at each node using column generation [4], [22].

Some works [23] have already studied how network calculus can be used to compute the optimal route for a flow in the presence of cross-traffic. However, due to tractability issues of latency models, flows are processed one by one.

Our paper goes beyond state of the art by considering a pragmatic architecture that trades-off between complexity and performance thanks to the use of a tractable latency model for round-robin schedulers. It presents efficient control plane algorithms to jointly optimize the routing of a set of flows in order to meet deterministic end-to-end latency requirements.

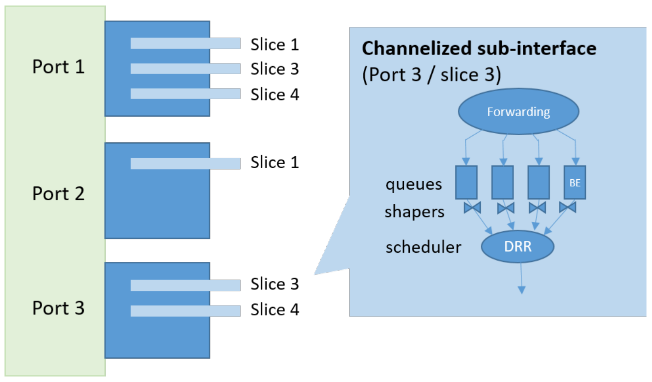

Fig. 1: Slicing architecture with channelized sub-interfaces.

III. Network slicing architecture

To prevent interferences between slices, we consider a network slicing architecture based on the channelized sub-interface technology [12], as currently supported by most vendors. The traffic from the different slices is forwarded through specific channelized sub-interfaces with 802.1Q encapsulation. Each channelized sub-interface implements an independent QoS scheduler, such as HQoS or DiffServ, to serve traffic with different requirements and priorities. Given that channelized sub-interfaces are provided with a dedicated resource assignment, it ensures hard isolation as it prevents a slice to interfere with the others. Indeed, as shown in Fig. 1, a sub-interface is assigned to each slice on physical links with a dedicated allocated capacity. Inside each sub-interface, traffic is scheduled using a round robin scheduler and per-flow shapers to share the slice capacity among the different flows (i.e., virtual services). Some queues are dedicated to HP flows, while others are dedicated to BE traffic to avoid starvation. Depending on the amount of traffic to serve, 2 or 4 queues can be assigned to HP traffic, leaving the rest to handle QoS and priorities for BE traffic.

Traditional data plane scheduling methods mainly include round robin schedulers, such as Deficit Round Robin (DRR) [8], in which queues are scheduled one after another. At each round, an amount of data based on each queue's weight or deficit is scheduled. DRR is among the most used scheduler at it exhibits a complexity in $O(1)$. It is a practical and efficient implementation of Generalized Processor Sharing (GPS) [24].

Besides being useful to prioritize traffic, round-robin methods can be used to provide deterministic guarantees as latency bounds can be computed with network calculus [25]. While several research works have proposed tight latency bounds (see [9], [26] for DRR for instance), a loose but tractable estimation of the worst-case latency can be derived considering that the scheduler is modeled as a latency-rate server [10]. According to this model, the upper bound of the delay D_e^q experienced by flows in each queue q on sub-interface e (or physical link, to simplify) can be expressed as

$$D_{eq} \leq T_{eq} + \frac{B_e^q}{C_{eq}} \tag{1}$$

where B_e^q is the sum of the bursts of all flows going through queue q of link e, C_{eq} is service rate associated to queue q over link e and T^{eq} is a constant factor that represents the worst-case delay seen by the first packet of a busy period and depends on the resources (i.e., to *quanta*) assigned to each queue in the scheduler.

According to network calculus terminology, each flow can be described by an *arrival curve* in the form of $\alpha_k(t) = b_k + r_k t$, where b_k is the maximum allowed burst for the k-th flow and r_k is the average allowed rate. As derived in [9], the service rate of a DRR scheduler guaranteed by queue q to flow k can be expressed as $\beta_k(t) = C_{eq}[t-T_{eq}]^+$. The queuing policy of each queue can be modeled as FIFO, resulting that the overall latency introduced by a queue is $T_{eq} + \frac{B_e^q}{C_{eq}}$. Considering only one flow in the network, at the next node, the data to be transmitted would be $B_e^q + C_{eq}T_{eq}$. However, if a shaper is used at the output of the queue, the resulting data to be transmitted is B_e^q. Using [27], if a shaper is applied after each queue as shown in Fig. 1, there is no additional delay to be considered and the arrival curve is preserved at each intermediate hop. It results that model (1) can be applied safely to each sub-interface in the network as we can consider that inside a queue there is only one flow resulting from the aggregation of the flows traversing the queue. In practice, the real latency of flows can be much smaller than the worst case given by the analytical model as (i) bursts of different flows may arrive at different time instants and (ii) the actual flow size can be smaller than the maximum burst. But this conservative model can be used to ensure that end-to-end latency requirements are met. The capacity reserved for a slice and unused by HP traffic can be exploited by BE traffic.

When a slice needs to be created, the network controller receives a list of virtual services described each by a pair of source and destination nodes and an arrival curve for each HP flow. Based on the characteristics of services and the updated information about the remaining network capacity, the controller decides the following: 1) how much capacity has to be allocated to each sub-interface, i.e., $\sum_q C_{eq}$ for $q \in Q$, $e \in E$, 2) which routing paths must be used for each service, and 3) to which queues on the different ports that are traversed (i.e., channelized sub-interfaces that are created) the services must be assigned. The controller should also take into account the amount of BE traffic in the slice to provide at least partial acceptance. However, for the sake of simplicity, we neglect this term for the slice planning optimization in Sec. IV and we assume that BE traffic will partially be able to use the resources of HP traffic as the network calculus bound is loose and yields to some over-provisioning. Using 4 to 6 queues can also allow prioritizing BE traffic inside the slice.

IV. Problem formulation

In this section, we formulate the slice planning problem where we maximize the number of accepted HP virtual services, also called *demands*, and we minimize the total capacity allocated to the slice. The maximization of traffic acceptance is the first objective, whereas the minimization of the allocated capacity is the second one. As explained in Sec. III, each slice is deployed using channelized sub-interfaces at physical ports. Each sub-interface has a number of round-robin queues through which demands can be assigned. The network controller decides about routing and bandwidth allocation so as to meet the latency requirement of each demand, giving in output a path and an assignment to queues and a service rate allocation for each queue. The latency-rate model is used to calculate the worst queuing delay in each queue. It is used to ensure that end-to-end latency requirements are feasible for each accepted demand.

Let $G = (V, E)$ be a graph, where V is the set of network devices (i.e., routers) and E is the set of directed links between devices (i.e., physical outgoing ports). For each link $e \in E$, we define its available capacity C_e and the set of queues Q_e associated with the sub-interface that is assigned to the slice under consideration. In the rest of the paper, we call the *link/queue eq* the queue $q \in Q_e$ associated with the sub-interface on link e. For each link/queue eq, the propagation delay is denoted l_e, and the scheduling delay at each queue is T_{eq}. Let K be the set of demands (i.e., virtual services of the slice). For each demand $k \in K$, we denote by s_k its source, by t_k its destination, and its arrival curve is defined by a maximum burst size b_k and an average rate r_k. Each demand k has an end-to-end latency requirement dl_k. Let $\bar{G} = (V, EQ)$ be an extended graph of G where each link $e \in E$ is duplicated $|Q_e|$ times to represent each link/queue $eq \in EQ$ where $q \in Q_e$.

We now propose a mixed integer non-linear programming with a polynomial number of variables and constraints. The variables are the following:

- $x_k^{eq} \in \{0,1\}$: equals to 1 if the demand k is assigned to the queue q on the link e, 0 otherwise.
- $C_{eq} \in \mathbb{R}$: equals to the rate allocated to the queue q on the link e.
- $x_k \in \{0,1\}$: equals to 1 if the demand k is accepted, 0 otherwise.
- $d_{eq} \in \mathbb{R}$: equals to the maximum delay of the queue.

The mathematical model (MP) is as follows:

$$\max \quad \sum_{k\in K} M x_k - \sum_{e\in E}\sum_{q\in Q} C_{eq} \tag{2}$$

$$\sum_{eq\in\delta^+(v)} x_k^{eq} - \sum_{eq\in\delta^-(v)} x_k^{eq} = v(k)x_k \qquad v \in V, k \in K, \tag{3}$$

$$\sum_{q\in Q_e} x_k^{eq} \le 1 \qquad k \in K, e \in E \tag{4}$$

$$T_{eq} + \frac{\sum_{k\in K} b_k x_k^{eq}}{C_{eq}} \le d_{eq} \qquad e \in E, q \in Q_e \tag{5}$$

$$\sum_{e\in E}\sum_{q\in Q_e} (d_{eq} + l_e) x_k^{eq} \le dl_k \qquad k \in K \tag{6}$$

$$\sum_{k\in K} r_k x_k^{eq} \le C_{eq} \qquad e \in E, q \in Q_e \tag{7}$$

$$\sum_{q\in Q_e} C_{eq} \le C_e \qquad e \in E \tag{8}$$

where M is a big value ensuring that the number of accepted demands is the main objective and $v(k)$ equals to 1 if $v = s_k$, -1 if $v = t_k$ and 0 otherwise. Inequalities (3) are the flow conservation constraints. Inequalities (4) ensure that a demand is assigned to at most one queue of a link.

Constraints (5) consider the model to compute the worst-case delay experienced in a queue. Inequalities (6) are the end-to-end latency constraints. Inequalities (7) ensure that for each queue the average rate is respected. Finally, inequalities (8) guarantee that the service rate allocated to each queue do not exceed the capacity of each link.

In the following we consider two variants of the problem. The first one where a delay bound is given in input for each queue and is used to compute the end-to-end delay and a second one, more accurate but more complex, where the delay of each queue is determined by the algorithm.

a) Bounded delay (BD): A given delay bound D_{eq} can be considered for each link/queue. In this case, it is used to compute the end-to-end delay ($d_{eq} = D_{eq}$) and the (MP) model that still guarantees that the bound is satisfied can be easily linearized. Indeed, the non-linearity comes from the inequality (5), but if d_{eq} is a constant (replaced by D_{eq} and not a variable), then we can replace (5) by

$$C_{eq}T_{eq} + \sum_{k \in K} b_k x_k^{eq} \leq C_{eq}D_{eq} \quad e \in E, q \in Q_e \qquad (9)$$

(MP) becomes similar to a classical multi-commodity flow with additional constraints. The main challenge in this case for the network manager is to set the delay bounds for each queue. In Sec. VI, we explain a simple method based on the clustering of demands.

b) Variable delay (VD): In this case, the delay d_{eq} experienced in each queue is decided by the algorithm. The model (MP) can be solved via the two-steps heuristic presented in Sec. V or using a non-linear solver like SCIP [28] only on small networks.

V. Heuristic algorithms

We introduce heuristics for the bounded and variable variants, respectively denoted as *BD* and *VD*.

A. Bounded delay (BD)

For the BD variant, we now present a Column Generation (CG) based heuristic and a greedy procedure.

a) CG_BD algorithm: To solve the problem on large scale networks, a classical method to efficiently solve multi-commodity flow problems is to use column generation [29] on an extended model where the arc variables x_k^{eq} are replaced by path variables x_k^p for each demand $k \in K$ and each path $p \in P_k$ where P_k is the set of all possible paths from s_k to t_k that respect the end-to-end delay constraint dl_k. The following extended model (EP) is equivalent to (MP)

$$\max \quad \sum_{k \in K} M x_k - \sum_{e \in E} \sum_{q \in Q} C_{eq} \qquad (10)$$

$$\alpha_k : \; x_k \leq \sum_{p \in P_k} x_k^p \qquad k \in K \qquad (11)$$

$$\alpha_{eq}^1 : C_{eq}T_{e,q} + \sum_{k \in K} b_k \sum_{p \in P_k : e \in p} x_k^p \leq C_{eq}D_{eq} \quad q \in Q, e \in E \qquad (12)$$

$$\alpha_{eq}^2 : \sum_{k \in K} \sum_{p \in P_k : e \in p} r_k x_k^p \leq C_{eq} \qquad e \in E, q \in Q \qquad (13)$$

$$\sum_{q \in Q_e} C_{eq} \leq C_e \qquad e \in E \qquad (14)$$

where α_k (resp. α_{eq}^1, α_{eq}^2) are the dual variables associated with the inequalities (11) (resp. (12), (13)). To solve the extended model (EP), we use a column generation algorithm. The associated pricing problem, for each demand $k \in K$ consists in finding a constrained shortest path in the extended graph $\bar{G}$ where the cost of each link/queue arc is equal to $b_k\alpha_{eq}^1 + r_k\alpha_{eq}^2$ and its associated delay is equal to the bound D_{eq} plus l_e. If the cost of this path is smaller than α_k then the column associated with this path (i.e., path variable x_k^p) for the demand k is added to the model. Each time the restricted (EP) model is updated, it is solved using a linear solver and dual variables are updated. When no new columns can be found, the algorithm terminates and returns the optimal relaxed solution of the extended model. The pseudo-code is presented in Alg 1.

Algorithm 1 CG_BD algorithm

```
Require: Set of demands $K$, graph $G$.
while Columns added at the last iteration or First iteration do
    Solve the restricted (EP)
    for all demand $k \in K$ do
        Solve pricing problem for $k$
        if path $p$ found such that $\sum_{eq \in p} b_k\alpha_{eq}^1 + \sum_{eq \in p} r_k\alpha_{eq}^2 < \alpha_k$ then
            add the variable $x_k^p$ to the restricted (EP)
        end if
    end for
end while
while max number of rounding steps not reached do
    current solution ← empty set of paths
    for $k \in K$ do
        select randomly a path $p$ with a probability proportional to the linear relaxation $x_k^{p*}$ and add it to the solution
    end for
    if the current solution is feasible and better then
        best solution = current solution.
    end if
end while
return Best solution
```

To heuristically derive an integer solution we then consider a randomized rounding algorithm where we generate 100 solutions by drawing a path at random for each demand based on the probability given by the linear relaxation.

b) Greedy_BD algorithm: For each demand $k \in K$, this algorithm computes a path and accepts this demand if the end-to-end delay is respected and the resources are enough for each link of the path. This algorithm aims at minimizing the total bandwidth allocated to accepted demands. More formally, the algorithm computes for each demand $k \in K$ a constrained shortest path in the extended graph $\bar{G}$ where the cost for each queue/link arc is equal to add_{eq}^k, the additional service rate needed if the demand uses this queue so that the queuing delay equals to D_{eq}.

Algorithm 2 *2S-VD* algorithm

```
Require: Set of demands $K$, graph $G$.
  while Solution can be improved do
    (Step 1) calculate the routing using CG_BD algorithm.
    (Step 2) allocate a minimum service rate to each queue.
    Update the delay bound of each queue.
  end while
  return Solution
```

Considering the delay model, we have $add_{eq}^{k} = \frac{b_k}{D_{eq} - T_{eq}}$ for all link/queue $eq \in EQ$.

B. Variable delay (VD)

We now present a two-steps heuristic, called *2S_VD*, for the VD case. As the problem is non-linear and much harder to solve than *BD*, we split the problem into two sub-problems. The first part computes a path for each demand, while the second part determines the rate allocation of each queue such that the end-to-end delay is respected for all the accepted demands and the capacity constraints are met on all links.

As presented in Alg. 2, the algorithm consists in solving the problem by iterating between two algorithms until no improvement is achieved. The first algorithm (*Step 1*) solves the BD problem using *CG_BD* to provide a routing P_K. The second algorithm (*Step 2*) consists in a greedy procedure to minimize the service rate allocation. Based on the routing P_K and the rates C_{eq}^{*} decided after the two steps, a new delay bound is given for each queue such that $D_{eq}^{*} = T_{eq} + \frac{\sum_{k \in K_{eq}} b_k}{C_{eq}^{*}}$ (if a queue/link is not used then $D_{eq}^{*} = D_{eq}$). This new delay becomes the input of the first step at the next iteration.

(Step 2) Given a routing P_K associated with a set of demands K (one path p_k per demand), the goal of the algorithm at *Step 2* is to provide for each link/queue $eq \in EQ$ a feasible and minimum quantity of service rate. A rate allocation is feasible if all paths in P_K meet their end-to-end latency requirement and if the sum of allocated rates to all the queues of Q_e is smaller than the link capacity C_e for each link $e \in E$. Remark that the best rate allocation (i.e., the minimum) for a link/queue $eq \in EQ$ is $BC_{eq} = \frac{\sum_{k' \in K'} b_{k'}}{d_{eq} - T_{eq}}$ where K' is its set of demands crossing eq and d_{eq} is its delay. The algorithm begins by fixing the rate allocation of each link/queue to C_e. It then iterates over all links/queues in EQ until an improvement is obtained. If at least one link $e \in E$ does not meet the link capacity constraint ($\sum_{q \in Q_e} C_{eq} \leq C_e$) then the algorithm considers only the violated links/queues. Otherwise, the algorithm considers all of them. At each iteration, for each link/queue the algorithm performs a sort of dichotomy search by updating the service rate as follows $C_{eq} = BC_{eq} + \frac{C_{eq} - BC_{eq}}{2}$, where BC_{eq} is the best rate allocation when the delay d_{eq} is equal to $\max_{k \in K'} \frac{dl_k - \sum_{e \in P_k} l_e}{|P_k|}$ with K' the set of demands crossing the link/queue eq.

VI. Experimental results

We now compare our slice planning algorithms in terms of traffic acceptance, reserved bandwidth and execution time.

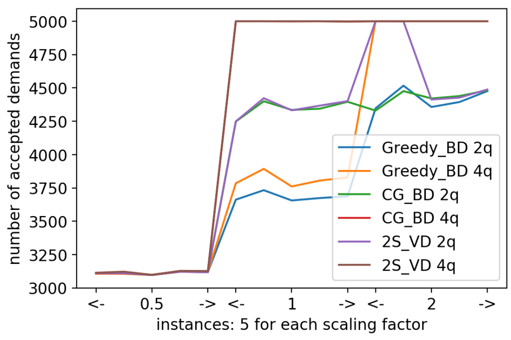

Fig. 2: Accepted demands for each instance. On the x-axis, 5 instances are considered for each scaling factor, i.e., 0.5, 1, 2.

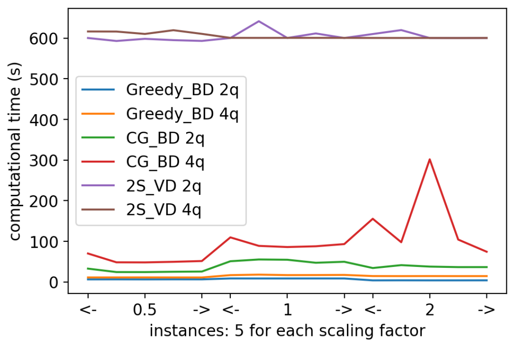

Fig. 3: Computational time for each instance. On the x-axis, 5 instances are considered for each scaling factor, i.e., 0.5, 1, 2.

A. Simulation settings

We test our algorithms on an IP-RAN (Radio Access Network) topology with a hierarchical structure: aggregation layers are connected to core, being themselves connected to access layers. The topology is composed of 1000 nodes and 2498 links, and we generate 5 random sets of 5000 demands. Although in practice, each sub-interface can handle up to 8 queues, we consider either 2 or 4 queues to analyze the impact of the number of queues. Furthermore, we also consider 3 scaling factors (respectively, 0.5, 1 and 2) that multiply the end-to-end delay of each demand to strength or relax the latency constraints, implying a decreased amount of allocated capacity to satisfy latency requirements. In total, we have $5 \times 2 \times 3 = 30$ test instances and the numerical results were realized on an Intel Xeon CPU E5-4627 v2 of 3.30GHz with 504GB RAM and 32 cores, running under Linux 64 bits. We used 10 parallel threads and a time limit of 600 seconds.

To initialize the delay bound for each queue, we partition the demands in $|Q|$ sets according to their end-to-end delay requirements divided by the length of the shortest path length plus a small constant (i.e., 2). Then, for each set, representing fast to slow queues, we set the bound to the minimum end-to-end delay requirements divided by the same value as above. This partitioning method is used to determine the delay bound in BD and an initial solution in VD.

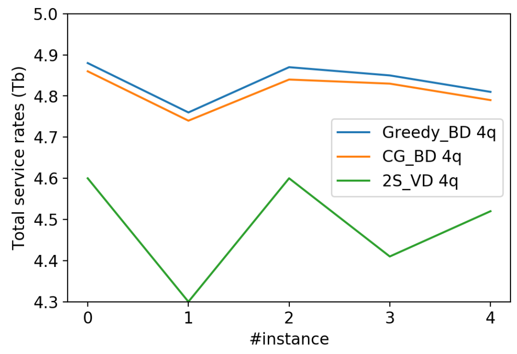

Fig. 4: Reserved capacity (all demands accepted).

B. Results

Fig. 2 and Fig. 3 present the number of accepted demands and the execution time for each instance, while Fig. 4 presents the total allocated service rates for instances where all demands are accepted. Remark that *Greedy_BD* accepts all demands only when 4 queues (i.e., *4q* in figures) are available and the scaling factor is equals to 2 thus Fig. 4 reports only results for 4 queues and a scaling factor of 2.

We first compare the results for the two algorithms solving the BD variant, namely *Greedy_BD* and *CG_BD*. We can observe that the column generation algorithm performs much better in terms of traffic acceptance, up to 8% of improvement, at the cost of a higher running time. On the instances where all algorithms accept all demands, *CG_BD* allows saving and extra 0.4% in average of the total allocated rate. We can also observe that increasing the number of queues give more flexibility to the algorithms and improves significantly traffic acceptance, also at the cost of higher running times as the extended graph is larger.

Second, we analyze the gain when a variable delay is decided (i.e., VD) compared to when a bounded delay given as input (i.e., BD). We can observe in the case with 4 queues that the accepted traffic is increased by 10% in average while the total reserved capacity is decreased by 7% (when all demands are accepted). As *2S_VD* requires more iterations compared to *CG_BD*, its execution time is higher while it remains below the time limit by design.

VII. Conclusion and Perspectives

We presented in this paper a network slicing architecture for deterministic latency guarantees. Based on legacy round-robin schedulers and per-flow shapers, the solution provides a good trade-off between complexity and performance. We presented two efficient algorithms for the planning of a single slice to maximize traffic acceptance and minimize the total reserved capacity. Future work along these lines may include the embedding of tighter, although more complex, delay models to further improve the utilization of resources.

References

[1] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, 2018.

[2] E. Grossman, "Deterministic Networking Use Cases," RFC 8578, May 2019. [Online]. Available: https://rfc-editor.org/rfc/rfc8578.txt

[3] R. Li, "Towards a New Internet for the Year 2030 and Beyond," *Proceedings 3rd Annual ITU IMT-2020/5G Workshop Demo Day*, 07 2018.

[4] B. Liu, S. Ren, C. Wang, V. Angilella, P. Medagliani, S. Martin, and J. Leguay, "Towards Large-Scale Deterministic IP Networks," in *IFIP Networking*, 2021.

[5] A. Nasrallah, V. Balasubramanian, A. S. Thyagaturu, M. Reisslein, and H. Elbakoury, "Cyclic Queuing and Forwarding for Large Scale Deterministic Networks: A Survey," *ArXiv*, vol. abs/1905.08478, 2019.

[6] "IEEE Standard for Local and Metropolitan Area Networks: Cyclic Queuing and Forwarding," *IEEE 802.1Qch-2017*, pp. 1–30, June 2017.

[7] J. C. Bennett and H. Zhang, "WF/sup 2/Q: worst-case fair weighted fair queueing," in *Proc. IEEE INFOCOM*, 1996.

[8] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Trans. on Net.*, vol. 4, no. 3, 1996.

[9] M. Boyer, G. Stea, and W. M. Sofack, "Deficit Round Robin with network calculus," in *6th International ICST Conference on Performance Evaluation Methodologies and Tools*. IEEE, 2012, pp. 138–147.

[10] D. Stiliadis and A. Varma, "Latency-rate servers: a general model for analysis of traffic scheduling algorithms," *IEEE/ACM Transactions on networking*, vol. 6, no. 5, pp. 611–624, 1998.

[11] China Mobile Communications Corporation, Huawei Technologies Co., Ltd., Deutsche Telekom AG, and Volkswagen, "5G Service-Guaranteed Network Slicing White Paper," Tech. Rep., Feb. 2017. [Online]. Available: https://www-file.huawei.com/-/media/corporate/pdf/whitepaper/ 5g-service-guaranteed-network-slicing-whitepaper.pdf

[12] Huawei, "Application Scenarios for VLAN Channelized Sub-Interfaces," March 2021. [Online]. Available: https://support.huawei.com/enterprise/en/doc/EDOC1100093973/ a85fddf5/application-scenarios-for-vlan-channelized-sub-interfaces

[13] 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 demo*, 2018.

[14] OIF, "Flex Ethernet 2.0 Implementation Agreement," June 2018. [Online]. Available: https://www.oiforum.com/wp-content/uploads/OIF-FLEXE-02.0.pdf

[15] N. Huin, J. Leguay, S. Martin, P. Medagliani, and S. Cai, "Routing and Slot Allocation in 5G Hard Slicing." in *INOC*, 2019, pp. 72–77.

[16] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc.IEEE INFOCOM*, 2000.

[17] B. Fortz, L. Gouveia, and M. Joyce-Moniz, "Models for the piecewise linear unsplittable multicommodity flow problems," *European Journal of Operational Research*, vol. 261, no. 1, pp. 30–42, Aug. 2017.

[18] W. Ben-Ameur and A. Ouorou, "Mathematical models of the delay constrained routing problem," *Algorithmic OR*, vol. 1, no. 2, 2006.

[19] L. Kleinrock, *Communication nets: Stochastic message flow and delay*. Courier Corporation, 2007.

[20] "Deterministic Networking Architecture," RFC 8655, Oct. 2019.

[21] L. Qiang, X. Geng, B. Liu, T. Eckert, L. Geng, and G. Li, "Large-Scale Deterministic IP Network," IETF Draft draft-qiang-detnet-large-scale-detnet-05, Sep. 2019.

[22] J. Krolikowski, S. Martin, P. Medagliani, J. Leguay, S. Chen, X. Chang, and X. Geng, "Joint Routing and Scheduling for Large-Scale Deterministic IP Networks," *Elsevier Computer Communication*, 2020.

[23] A. Bouillard, B. Gaujal, S. Lagrange, and É. Thierry, "Optimal routing for end-to-end guarantees using network calculus," *Performance Evaluation*, vol. 65, no. 11-12, pp. 883–906, 2008.

[24] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM trans. on Net.*, vol. 1, no. 3, 1993.

[25] M. Fidler, "Survey of deterministic and stochastic service curve models in the network calculus," *IEEE Communications surveys & tutorials*, vol. 12, no. 1, pp. 59–86, 2010.

[26] A. Bouillard, "Individual Service Curves for Bandwidth-Sharing Policies using Network Calculus," *IEEE Networking Letters*, pp. 1–1, 2021.

[27] J. Le Boudec, "A Theory of Traffic Regulators for Deterministic Networks With Application to Interleaved Regulators," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2721–2733, 2018.

[28] G. Gamrath, D. Anderson, K. Bestuzheva, W.-K. Chen, L. Eifler, M. Gasse, P. Gemander, A. Gleixner, L. Gottwald, K. Halbig *et al.*, "The SCIP optimization suite 7.0," 2020.

[29] G. Desaulniers, J. Desrosiers, and M. M. Solomon, *Column generation*. Springer Science & Business Media, 2006, vol. 5.