Online Bandwidth Calendaring: On-the-Fly Admission, Scheduling, and Path Computation

Maxime Dufour, Stefano Paris, Jérémie Leguay, Moez Draief
Mathematical and Algorithmic Sciences Lab
France Research Center - Huawei Technologies Co. Ltd.
Boulogne-Billancourt, France
Email: {name.surname}@huawei.com

***Abstract*—The centralized control in Software Defined Networks paves the way for new services like Bandwidth Calendaring (BWC), where the possibility to shift temporally future bandwidth requests allows to efficiently use network resources. Assuming perfect knowledge of the calendar for all future bandwidth reservations is unrealistic. In this paper, we study the online version of the BWC problem presented in [1], where for unpredictable incoming demands an admission decision, scheduling and path allocation must be taken instantaneously. We design an algorithm for solving the online version of the BWC problem and proposes two heuristic approaches to exploit the scheduling flexibility of demands. Our numerical results reveal that the proposed solution approach outperforms state-of-the art methods by up to 70% in terms of accepted traffic.**

***Index Terms*—Software Defined Networking, Online Bandwidth Calendaring, Scheduling, Online Optimization.**

I. Introduction

Software-Defined Networking (SDN) [2] technologies have radically transformed the network architecture of data centers, network overlays, and carrier networks. By offloading the control plane to a remote platform, the control plane can now be implemented on top of commodity servers and benefit of their high computational power. Although the controller platform is usually distributed on multiple servers, it keeps a global view of the network status in real-time and pushes consistent configuration updates to network equipment. The resulting centralized control as well as the high flexibility and programmability of SDN enable service providers to exploit more efficiently their network resources. This eventually permits to quickly develop and offer new types of network services, whose implementation was more challenging with legacy network devices.

Bandwidth Calendaring (BWC) [1] is one of such emerging connectivity services that SDN enables. It helps enterprises or cloud providers in establishing connectivity at low cost for bulk data transfers with guaranteed bandwidth and quality of service. Its main goal is to optimize the execution of large batch processes (e.g., Hadoop jobs, database backups) which consume significant network resources. Indeed, in most of the cases, these resource-intensive tasks can tolerate delay as long as they are completed before a given deadline. For this type of services, service providers envisage new on-demand provisioning models where the customer is billed on a usage basis, similarly to what already exists for computational and storage resources in cloud platforms. A service provider can then decide to monetize the remaining capacity of its network to sell low cost connectivity services for delay-tolerant bulk data transfers. In this context, the goal for the service provider is to maximize the accepted volume of traffic and its revenue by leveraging the scheduling flexibility of delay-tolerant bandwidth reservations.

A challenging issue for the widespread adoption of the BWC service is the lack of an automation system for the creation of the calendar in highly dynamic scenarios (i.e., the set of demands with their traffic profile). Assuming perfect knowledge of future arrivals is often unrealistic. Indeed, traffic demands are usually revealed and notified when applications need network connectivity. In such a setting, Over-The-Top (OTT) operators cannot precisely predict important parameters such as the traffic profile, duration, and arrival time of data connections. Yet, these parameters are essential for building up the calendar of future bandwidth reservations, which is used by the BWC service to efficiently allocate network resources over time. We therefore advocate a system that learns and builds the calendar in an online fashion without any assumption on the knowledge of future arrivals.

To this end, we design an admission control mechanism with the objective of maximizing the volume of accepted traffic. The system decides acceptance, scheduling and routing of demands in an online fashion. Extending an algorithm for online routing [3], we propose two modifications to decide the scheduling of demands. These online algorithms take sequential decisions without knowing the future. To minimize the optimality gap with respect to the offline optimal, the general idea is to compute paths over a modified network graph, called the *oracle*, where weights depend exponentially on the link utilization. The oracle is used to proactively reject some demands and load balance network resources over time. The first algorithm we propose takes an immediate decision while the second can postpone it if the demand cannot start immediately. The online nature of the proposed algorithms permits to quickly provide an admission feedback to applications and commit resources to secure/ensure the admission decision. Numerical results show that our algorithms increase the accepted throughput by up to 70% and load balance the resources of the system with respect to other approaches with the same computational complexity.

The paper is structured as follows. Section II presents an overview of related research literature. In Section III we describe the system model and we formulate the onsline version of the BWC problem, while in Section IV we present the two algorithms to solve the problem. Section V compares the approaches we proposed for solving on-the-fly the BWC problem against other heuristic approaches proposed in the literature, showing the performance gain of our approaches. Finally, concluding remarks are discussed in Section VI.

II. Related Work

With the advent of SDN, the interest in dynamic routing and Traffic Engineering (TE) methods has been renewed. Designing TE over traditional distributed routing protocols is sub-optimal [4]. Instead, several works propose methods to exploit the global network view available at the SDN controller to optimize resource allocation [5]. If network reconfigurations could be calculated and applied instantaneously, the resulting bandwidth reservations and path allocations would be optimal. In reality, updating all the involved network devices with the new centrally-made decisions requires a significant amount of time and it might affect ongoing data transmissions. In this context, any knowledge about the future can be used to optimize resource allocations over time. Thus, the potential of bandwidth calendaring is explored in a series of works, e.g., [6]–[9].

The time dimension of TE over SDN is considered in [7], where the problem of optimal allocation of current and future bandwidth resources is studied. A max-min objective is pursued, regarding the minimum fraction over all demands that is satisfied by its deadline. The problem of fair bandwidth allocation to a set of demands over predetermined paths is investigated in [10]. An efficient method to schedule a batch of demands so that the overall throughput is maximized has been proposed in [1]. The market aspects of such bandwidth calendaring problems are considered in [6], where the utility of each user is assumed to be decreasing on the delay until the transfer has been completed. A pricing mechanism is proposed so that users reveal their true valuation of bandwidth.

All aforementioned works consider the inter-DC scenario where data transfers have to be delivered up to a deadline. Although this scenario captures the benefits of calendaring and scheduling in TE problems, as we demonstrate in Section IV applying directly this approach to our bandwidth reservation setting is highly suboptimal. To the best of our knowledge, [11] is the only work that considers time-varying bandwidth requirements, but in the setting of communicating virtual machines (VM). The authors address the resulting VM placement problem via dynamic programming.

III. Online Bandwidth Calendaring

In this section we present the system model and the assumptions we consider in the formulation of the online BWC problem.

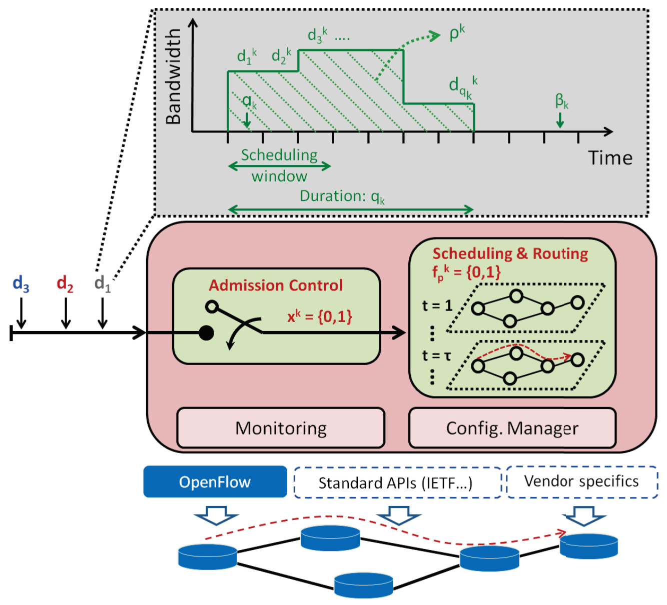


Fig. 1: BWC system to decide acceptance, scheduling and routing of demands having a time-varying bandwidth profile.

A. System Model

We consider a Wide Area Network (WAN) represented by an undirected graph $G\{\mathcal{V},\mathcal{E}\}$, where each edge e corresponds to a network link and is characterized by its capacity b_e and cost c_e per unit of traffic. Let $\mathcal{K}$ denote the set of demands that are expected to arrive in any order. For the sake of clarity, we discretize time into a set $\mathcal{T}$ of epochs of equal duration. Each demand $k \in \mathcal{K}$ is defined as a tuple $\langle s^k, t^k, \alpha^k, \beta^k, q^k, \mathbf{d}_k, \rho^k \rangle$. The parameters s^k and t^k represent the *source* and *destination* nodes of the demand, whereas α^k, β^k, q^k define the *earliest starting epoch*, the *latest ending epoch* (i.e., the *deadline*), and the *duration* of a demand (in number of epochs), respectively. Finally, the vector $\mathbf{d}_k = \left[d_1^k, d_2^k, ..., d_\tau^k, ..., d_{q^k}^k\right]$ describes the *traffic profile* of demand k where d_τ^k corresponds to the size of the demand k in its τ-th *epoch*, as illustrated in Fig. 1. The demand profit ρ^k corresponds to the traffic volume, namely $\rho^k = \sum_{t=1}^{q^k} d_t^k$. Note that demands can be started at any epoch within the scheduling window $\left\{\alpha^k, \cdots, \beta^k - q^k + 1\right\}$.

In contrast to the offline version of the BWC problem where the network operator knows the calendar of future bandwidth reservations, the goal here is to rapidly decide and provide feedback to applications. The challenge in this context comes from the online nature of the optimization problem: new variables are revealed sequentially, as soon as an arrival of a flow occurs in the system. Even in the case that arrival rates can be estimated from past observations, the exact sequence of future requests is not known in advance. In this setting, we consider that the earliest starting epoch corresponds to the epoch where the demand arrives α^k and that a decision has to be made as quickly as possible. To this end, for each arrival the network operator faces the decision whether to accept the demand, when to set up the data connection and where to accommodate the traffic in the network (i.e., which path to use to route the traffic from the origin to the destination). The main objective is to maximize the volume of data traffic accepted over time. To this aim, an operator may reject low-profit demands over highly utilized paths as they may prevent the future acceptance of high-profit demands.

We observe that admission decisions for the BWC service are non-preemptive to prevent any service interruption. Therefore, once a demand is accepted and network resources are reserved, the operator cannot halt the data connection in order to admit larger demands that may arrive in the future. Table I summarizes the notation used throughout the paper.

Parameter	Description
$\mathcal{V}$	Nodes (network devices).
$\mathcal{E}$	Edges (network links).
c_e	edge cost (in cost units) $e \in \mathcal{E}$.
b_e	edge capacity (in capacity units) $e \in \mathcal{E}$.
$\mathcal{K}$	set of demands (i.e., commodities).
$\alpha^k \in \mathcal{T}$	Arrival time of demand $k \in \mathcal{K}$.
$q^k \in \mathcal{T}$	Duration for demand $k \in \mathcal{K}$.
$\beta^k \in \mathcal{T}$	Latest ending time for demand $k \in \mathcal{K}$ $\left(\beta^k \geq \alpha^k + q^k - 1\right)$.
d^k_τ	Bandwidth request for demand $k \in \mathcal{K}$ at time epoch $\tau\{1, 2, \dots, q^k\}$.

Variable	Description
$x^k \in \{0, 1\}$	Whether demand k is admitted in the system.
$f_{pt} \in \{0, 1\}$	Whether path $p \in \mathcal{P}_k$ is selected to route demand k starting from epoch t ($f_{pt} = 1$).

TABLE I: Input parameters and variables of the problem.

B. Problem Formulation

In the following we present the *path-schedule*-based formulation of the offline BWC problem. Before introducing the model, let us define the set of paths and decision variables used in our formulation. $\mathcal{P}$ denotes the set of all network paths, whereas $\mathcal{P}_k \subseteq \mathcal{P}$ defines the subset of paths that connect the source s^k to the destination t^k of demand k (i.e., set of paths over which the demand can be routed). Similarly, $\mathcal{P}_e \subseteq \mathcal{P}$ identifies the set of paths that traverse edge $e \in \mathcal{E}$.

Let x^k be the binary decision variable used for the admission decision of a demand ($x^k = 1$ if demand k is accepted). Furthermore, we use binary decision variables f_{pt} with $p \in \mathcal{P}_k$ and $t \in \mathcal{T}$ to decide the starting epoch t at which path p is used to route demand k. We refer to variable f_{pt} as *path-schedule* (p, t), since it jointly provides the scheduling and routing of a demand. The offline BWC problem can be formulated as (1)-(5):

$$\max \sum_{k\in\mathcal{K}} \sum_{t\in\mathcal{T}} d^k_t x^k \tag{1}$$

$$s.t. \sum_{p\in\mathcal{P}_k} \sum_{t=\alpha^k}^{\beta^k - q^k + 1} f_{pt} = x^k \qquad \forall k \in \mathcal{K} \tag{2}$$

$$\sum_{p\in\mathcal{P}_e} \sum_{k:p\in\mathcal{P}_k} \sum_{\tau=1}^{\min\{q^k,t\}} d^k_\tau f_{p(t-\tau+1)} \leq b_e \qquad \forall e \in \mathcal{E}, t \in \mathcal{T} \tag{3}$$

$$f_{pt} \in \{0, 1\} \qquad \forall p \in \mathcal{P}, t \in \mathcal{T} \tag{4}$$

$$x^k \in \{0, 1\} \qquad \forall k \in \mathcal{K} \tag{5}$$

The objective function (1) maximizes the accepted volume of traffic. Constraints (2) impose that an accepted demand starts its transmission at a unique starting epoch and uses a single path, while the set of constraints (3) ensures that the allocation stay within the capacity region during each epoch.

In the *online* setting presented in this paper, the parameters and decision variables of the BWC problem (1)-(5) are discovered in a unknown sequence that depends on the demands arrival. The goal is still to maximize the accepted volume of traffic over the time horizon $\mathcal{T}$. However decisions for any new demand must be made before the end of the scheduling window without knowing the sequence of arrivals (i.e., the calendar) and without the possibility of changing past decisions. In the next Section we present two algorithms for solving the online BWC problem that minimize the optimality gap with respect to the offline version, which instead assumes perfect knowledge of the whole sequence of demands.

IV. On-line Algorithms for Admission, Scheduling, and Path Allocation

In this section we present two algorithms that solve the joint admission, scheduling, and routing decisions of the online BWC problem. These algorithms extends the online routing algorithm proposed by Awerbuck in [3], where only admission and routing decisions were considered. We modify this algorithm to decide the scheduling of all demands in addition to the routing once they are accepted. Finally, we show that the solution obtained over a time horizon has provable performance guarantees.

An online algorithm takes sequential decisions without knowing the future. To minimize the optimality gap with respect to the offline optimal that can be computed knowing exactly the sequence of arrivals like in [1], we need to proactively reject some demands and load balance network resources over time. Small demands that do not contribute to the maximization of the accepted traffic can be rejected. At the same time, scheduling and routing decisions have the twofold objective of avoiding the utilization of bottleneck links and the creation of traffic spikes. To achieve this goal, the general idea is to compute paths over a modified network graph, called the *oracle*, where weights depend exponentially on the link utilization. This weight can be interpreted as the urgency of avoiding the resource: high traffic load results into a steep increase of the weight of congested resources in order to restrain their use in the near future. In addition, they are used to define an admission criteria which compares the price of accepting a request to its profit (i.e., the traffic volume of the demand). If the admission price is higher than the expected profit, the connection is rejected.

Before detailing the two algorithms, let us define the relative load of and edge e seen by demand k as follows:

$$\lambda_{et}(k) = \frac{\sum_{p\in\mathcal{P}_e} \sum_{m:p\in\mathcal{P}_k \wedge m<k} \sum_{\tau=1}^{\min\{q^m,t\}} d_\tau^k f_{p(t-\tau+1)}}{b_e} \tag{6}$$

The relative load seen by demand k depends on all decisions $\langle x^k, f_{pt}\rangle$ taken for all demands m arrived before k (recall that for demands with the same arrival time we can break ties arbitrary). In both algorithms, the edge cost function given by Equation (7) is coupled with a shortest path computation method (e.g., Dijkstra) to select the least congested links.

$$w_e(k) = \sum_{t=\iota}^{\iota+q^k} \lambda_{et}(k)\, b_e \left(\mu^{\lambda_{et}(k)} - 1\right) \tag{7}$$

Such a function increases exponentially with the relative load, thus decreasing the possibility of selecting a link as its load increases. We refer to this function as admission price of the link. Note that both $\lambda_{et}(k)$ and $w_e(k)$ are defined only within the interval between the earliest starting time and the deadline of the demand k.

A. *One-Shot Scheduling and Routing*

Algorithm 1 illustrates the steps of what we call the one-shot scheduling and routing approach, in short *one-shot*, which decides simultaneously whether to accept the demand, its starting time within the scheduling window and the path to connect the origin to the destination. When demand k arrives, the algorithm selects among all possible epochs of its scheduling window the one that produces a path p with the lowest admission price $\sum_{e\in p} w_e$. Among the paths generated for each possible epoch, only those with a total admission price lower than the profit ρ^k of the demand are retained as candidates for routing the traffic of k from its origin to its destination. The profit is $\sum_{t\in\mathcal{T}} d_t^k$, the total volume of data traffic that the demand will contribute to the throughput maximization objective. If any such path exists for any possible epoch of the scheduling window, the demand is rejected. Upon arrival of a demand, the *one-shot* algorithm takes a decision and provides an immediate answer to the application. We would like to underline that the paths computed for each possible

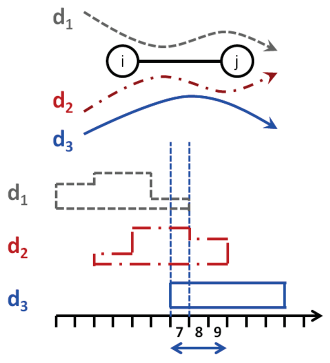


Fig. 2: Computation of relative load for edge $(i;j)$ seen by demand 3. Demand 3 does not see the load of 1 after epoch 7.

Algorithm 1: One-Shot Online BWC

```
  Input: Network topology: $G\{\mathcal{V},\mathcal{E}\}$, $\mathbf{b}$, $\langle s^k, t^k, \alpha^k, \beta^k, q^k, \mathbf{d}_k\rangle$
  Output: Admission, Scheduling, Routing
  $\eta^k \leftarrow \rho^k$
  $x^k \leftarrow 0$
1 for $\iota \in \{\alpha^k, ...\beta^k - q^k + 1\}$ do
      /* Select unique starting time for $k$ */
2     Update $\lambda_{et}(k)$, $\forall t \in \{\iota, ..., \iota + q^k\}, e \in \mathcal{E}$
      Update $w_e(k)$, $e \in \mathcal{E}$
3     Compute shortest path $p$ over $G\{\mathcal{V},\mathcal{E},w_e(\cdot)\}$
4     if $\sum_{e\in p} w_e(k) \le \rho^k$ and $\sum_{e\in p} w_e \le \eta^k$ then
          $\eta^k \leftarrow \sum_{e\in p} w_e(k)$
          $x^k \leftarrow 1$
          $f_{p\iota} \leftarrow 1$
5 if $x^k = 0$ then
      reject demand $k$.
```

epoch may differ, since when we shift the demand towards its deadline β^k some demands started in an earliest epoch might terminate and disappear from the computation of the relative load. As illustrated in Figure 2, demand 1 terminates after epoch 7 and it is not considered anymore after such an epoch in the computation of the relative load and admission price of edge $(i;j)$ seen by demand 3. Therefore, *one-shot* selects either epoch 8 or 9 as starting time of demand 3, since link $(i;j)$ is less congested.

B. *Postpone Scheduling and Routing*

The second approach we propose, called postpone scheduling and routing (in short *postpone*), may not take an immediate decision. Algorithm 2 details the operations performed by *postpone* when a new demand k arrives. At each epoch, as in the one-shot algorithm, we first update the relative load and the admission price within the time window of the demand. However, we evaluate only the first starting epoch of the

Algorithm 2: Postpone Online BWC

```
  Input: Network topology: $G\{\mathcal{V},\mathcal{E}\}$, $\mathbf{b}$, $\langle s^k, t^k, \alpha^k, \beta^k, q^k, \mathbf{d}_k\rangle$
  Output: Admission, Scheduling, Routing
  $x^k \leftarrow 0$
  /* Test starting time for $k$ */
1 Update $\lambda_{et}(k)$, $\forall t \in \{\alpha^k, ..., \alpha^k + q^k\}, e \in \mathcal{E}$
  Update $w_e(k)$, $e \in \mathcal{E}$
2 Compute shortest path $p$ over $G\{\mathcal{V},\mathcal{E},w_e(\cdot)\}$
3 if $\sum_{e\in p} w_e(k) \le \rho^k$ then
      $x^k \leftarrow 1$
      $f_{p\iota} \leftarrow 1$
  else if $\alpha^k < \beta^k - q^k + 1$ then
      /* Postpone $k$ to the next timeslot */
      $\alpha^k \leftarrow \alpha^k + 1$
  else
4     reject demand $k$.
```

scheduling window and we compute the shortest path in the oracle. If there exists no path with an admission price lower than the demand profit in the current epoch, the decision is postponed to the next epoch. Therefore the demand is put in the pool of arrivals of the next epoch. If in the latest epoch of the scheduling window (i.e., $\beta^k - q^k + 1$) the demand profit does not compensate the admission price, we definitely reject the demand.

V. Numerical Results

In this section we compare the performance of our admission control algorithms, referred to as *Postpone* and *OneShot*. We first describe the experimental methodology of our simulations. Then, we discuss the comparative evaluation of our methods against other heuristic approaches based on the same scheduling policies.

A. Experimental Methodology

To evaluate our solutions, we consider two different scenarios: i) a *random* scenario where we compare our algorithms in terms of accepted traffic, and ii) a *realistic* scenario that permits to validate our solutions in a realistic setting.

In the *random* scenario, we generated networks according to the small-world model, where few nodes called hubs permits to reach every other node in the network by a small number of hops. The typical distance L between any two randomly chosen nodes grows proportionally to the logarithm of the number of nodes N in the network ($L \propto \log N$). We consider 4 different instances with increasing number of nodes, links, and demands as illustrated in Table II. In the *realistic* scenario we used the GEANT topology [12], the high bandwidth pan-European research and education backbone composed of 22 nodes and 36 high capacity 40G bidirectional links.

In both scenarios demands are generated randomly over a period of 24 hours using distributions that closely model the real traffic. More specifically, we have used the Zipf distribution with exponent equal to 2.5 to select the $|\mathcal{K}|$ origin-destination pairs to model the skew distribution of pairwise connections. Furthermore, we increase the size of demands around noon to simulate the classical diurnal pattern of real traffic. The 24 hours period has been split into 96 consecutive epochs corresponding to a granularity of 15 minutes. In all simulations we performed 100 independent measurements computing very narrow 95% confidence intervals.

We compare our OneShot and Postpone algorithms against a greedy approach that routes demands using the cheapest path over the residual graph. For a fair comparison, we kept in the greedy approach the scheduling philosophy of testing all starting epochs or postponing the demand in case on insufficient resources. We replace the computation of the admission price in Algorithms 1 and 2 with the computation of the shortest path using as distance function the original edge cost c_e after pruning from the original graph the edges that have not enough residual capacity during the duration of a demand. In other words, if $\exists \tau \in \{\iota + \alpha^k, ..., \iota + \alpha^k + q^k\} : r_{e\tau} + d^k_\tau > b_e$, where ι is the epoch within the scheduling window,we prune edge e, and we compute the shortest path. If after the pruning the origin and destination are disconnected, the demand is rejected.

B. Random Networks

The performance of any admission control algorithm depends mainly on the strategy used to select network resources. In this section, we show that using the admission price increases the volume of accepted traffic with respect to a greedy policy (called *Greedy*) that tends to quickly saturate network hubs.

	Small	**Medium**	**Large**	**Very-Large**
Nodes	100	100	200	1000
Links	500	500	4000	15000
Demands	100	1000	10000	10000

TABLE II: Parameters used in the *random scenario*.

Figure 3 shows the gain in terms of volume of accepted traffic of our approaches with admission price against Greedy. We observe that in all instances and independently of the scheduling mechanism the algorithms with *admission price* always outperforms Greedy. As illustrated in the figure, our approaches (either OneShot or Postpone) accept 25% to 70% more traffic than Greedy. Greedy tends to quickly saturate the cheapest edges and it does not discriminate between the demands according to their profit. Therefore, small volume demands can consume resources that are later needed for the transfer of huge volume of traffic. Note that rejecting small demands in order to leave room for future demands does not solve the problem, since we may wait forever for the arrival of large demands (recall that we do not know the future). In contrast, the admission price permits to distribute the traffic over the network resources and anticipate the arrival of huge demands in the future. The figure depicts also the gain of Postpone over OneShot, which varies between 5% and 20% depending on the size of the instance we consider. This is due to the different approach used to select the starting time of the demand. OneShot delays the demands by choosing the latest epochs of the scheduling window as starting time, since for a generic demand k the network is generally seen as less

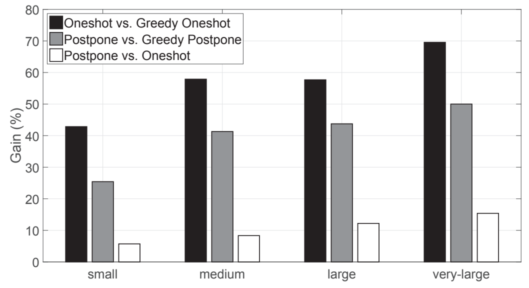
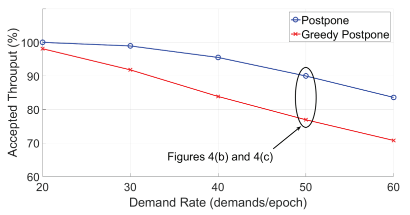
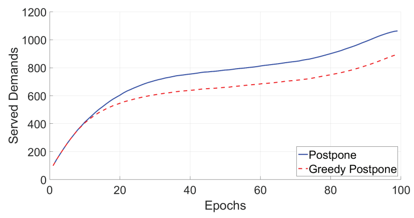


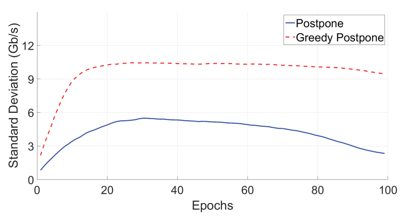
Fig. 3: Gain of our algorithms against Greedy in terms of accepted traffic for different network sizes.



(a) Accepted Traffic



(b) Time Evolution of Accepted Demands



(c) Variation of resource utilization

Fig. 4: Performance evaluation of the Postpone and Greedy over GEANT topology.

congested in the future. In contrast, Postpone serves a demand in the first epoch where its profit compensates the admission price of used resources, leaving the network unloaded in the far future. In other words, Postpone uses the smallest portion of the scheduling window necessary to accept the demand. The counterpart is that it does not take decisions immediately.

C. *Realistic Network*

To evaluate the effect of the traffic load, we generate demands according to a Poisson process with arrival rate varying in the range $[20; 60]$ demands/s. The size of the scheduling window and the duration are drawn from negative exponential distributions with parameter 8 and 16 epochs (i.e., 2 and 4 hours), respectively. Since Postpone provides better results than OneShot, in the following we present only the comparative evaluation of Postpone against the corresponding greedy approach, referred to as *Greedy*.

Figure 4 shows the main results we obtained using the GEANT topology. It can be observed from Figure 4(a) that even in realistic settings Postpone with admission price admits up to 20% more traffic than Greedy that simply uses the shortest path over the residual graph. The reason is illustrated in Figures 4(b) and 4(c), which show the temporal evolution of served demands and standard deviation of the link utilization for arrival rate equal to 50 demands/s. During the first 10 epochs, both algorithms accept and serve the same amount of traffic. However, the traffic is not fairly distributed over network resources, as we can observe from the evolution of the instantaneous standard deviation of the link utilization in Figure 4(c). Greedy consumes quickly the capacity of cheapest links and after the 10th epoch it starts rejecting more demands than Postpone. The accepted traffic between the two approaches starts to diverge as illustrated in Figure 4(b) and the greedy scheme is not able to catch up with of our scheme. The decrease in the standard deviation of the link utilization is simply due to the increase of demand arrivals, which require the connection of different endpoints (recall that we have a peak starting around noon). The rate of decrease is higher with the admission price since it better loads balances the utilization of residual resources.

VI. Conclusion

BWC services, where operators have the possibility to shift temporally future bandwidth requests, helps enterprises or cloud providers in establishing connectivity at low cost for bulk data transfers with guaranteed bandwidth and quality of service. The impossibility of knowing or accurately predicting future traffic represents the limiting factor for the use of BWC in real network deployments. In this paper, we propose two algorithms to jointly decide the admission, scheduling and routing of bandwidth reservations being totally oblivious to the future. The online nature of the proposed algorithms permits to quickly provide an admission feedback to applications and commit resources to secure the admission decision. Numerical results show that our algorithms increase the accepted throughput by up to 70% and load balance the resources of the system with respect to other approaches with the same complexity.

References

[1] L. Gkatzikis, S. Paris, I. Steiakogiannakis, and S. Chouvardas, “Bandwidth calendaring: Dynamic services scheduling over software defined networks,” in *Proc. IEEE ICC*, May 2016.

[2] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[3] B. Awerbuch, Y. Azar, and S. Plotkin, “Throughput-competitive on-line routing,” in *Proc. FOCS*, 1993.

[4] B. Fortz, J. Rexford, and M. Thorup, “Traffic engineering with traditional ip routing protocols,” *Communications Magazine, IEEE*, vol. 40, no. 10, pp. 118–124, 2002.

[5] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, “A roadmap for traffic engineering in sdn-openflow networks,” *Computer Networks*, vol. 71, pp. 1–30, 2014.

[6] H. Zhang, K. Chen, W. Bai, D. Han, C. Tian, H. Wang, H. Guan, and M. Zhang, “Guaranteeing deadlines for inter-datacenter transfers,” in *Proc. ACM EuroSys*, 2015.

[7] S. Kandula, I. Menache, R. Schwartz, and S. R. Babbula, “Calendaring for wide area networks,” in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM ’14. New York, NY, USA: ACM, 2014, pp. 515–526, printed.

[8] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, “B4: Experience with a globally-deployed software defined wan,” in *Proc. ACM SIGCOMM CCR, volume=43, number=4, pages=3–14, year=2013,*.

[9] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, “Achieving high utilization with software-driven wan,” in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 15–26.

[10] A. Kumar, S. Jain, U. Naik, A. Raghuraman, N. Kasinadhuni, E. C. Zermeno, C. S. Gunn, J. Ai, B. Carlin, M. Amarandei-Stavila, M. Robin, A. Siganporia, S. Stuart, and A. Vahdat, “Bwe: Flexible, hierarchical bandwidth allocation for wan distributed computing,” in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM ’15. New York, NY, USA: ACM, 2015, pp. 1–14.

[11] D. Xie, N. Ding, Y. C. Hu, and R. Kompella, “The only constant is change: Incorporating time-varying network reservations in data centers,” in *Proc. ACM SIGCOMM*, 2012.

[12] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, “Providing public intradomain traffic matrices to the research community,” *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 83–86, 2006.