Admission Control with Online Algorithms in SDN

Jérémie Leguay, Lorenzo Maggi, Moez Draief, Stefano Paris, Symeon Chouvardas
Mathematical and Algorithmic Science Lab, France Research Center
Huawei Technologies Co. Ltd., Boulogne-Billancourt, France

***Abstract*—By offloading the control plane to powerful computing platforms running on commodity hardware, Software Defined Networking (SDN) unleashes the potential to operate computation intensive machine learning tools and solve complex optimization problems in a centralized fashion. This paper studies such an opportunity under the framework of the centralized SDN Admission Control (AC) problem. We first review and adapt some of the key AC algorithms from the literature, and evaluate their performance under realistic settings. We then propose to take a step further and build an AC meta-algorithm that is able to track the best AC algorithm under unknown traffic conditions. To this aim, we exploit a machine learning technique called Strategic Expert meta-Algorithm (SEA).**

***Keywords*—Software Defined Networking, Online Algorithms, Machine Learning, Routing, Admission Control**

I. Introduction

Software-Defined Networking (SDN) technologies have radically transformed the network architecture of data centers, network overlays, and carrier networks [1]. They provide programmable data planes that can be configured from a remote controller platform. This engenders a separation between control and data planes, thus creating an opportunity to implement routing processes that are more efficient than classic ones: in fact, the controller can take real-time decisions at a (logically) *centralized* location using an accurate and global view of the network.

A key task of the SDN controller is the *Admission Control* (AC) on incoming connection requests. Its goal is to gracefully manage service requests when the network becomes highly utilized. AC accepts or drops new requests depending on the resource availability. Non-myopic decisions have to be made with the aim of maximizing a given profit, such as the total accepted throughput, the financial revenue generated or the quality of service experienced by users. Nowadays, most of the deployed AC procedures are threshold-based. They use max-, min-, exclusive- and non-exclusive-limits on resource portions that the network operator can define for different classes of flows. The main problem here lies in defining the threshold in a dynamic fashion, as the optimal configuration depends on the network traffic conditions, which fluctuate over time.

In this paper we wish to raise the awareness that the ability of SDN controllers to *centrally* manage the network is an opportunity to revisit the way AC is performed. More specifically, we propose to apply AC online algorithms originally conceived for covering and packing problems. Thus, we first make a thorough review of online algorithms proposed in the operation research literature and adapt them to the AC problem.

Email: firstname.lastname@huawei.com

The authors would like to thank Moti Medina for useful comments and inspiring discussions.

We then take a step further, and we propose to exploit the computational power offered by SDN controllers to implement *machine learning* techniques to boost the admission control performance (e.g., in terms of accepted throughput) via expert meta-algorithms, which are able to adaptively track the best AC algorithm without knowing the traffic statistics *a priori*. Specifically, we pinpoint a meta-algorithm called Strategic Expert meta-Algorithm (SEA) [2] which shows theoretical guarantees under our reactive scenario, on which there is little research.

II. Offline Admission Control

We represent the network as a capacitated graph $G(V, E)$, where V and E are the set of nodes and directed edges in the graph, respectively. Let $n = |V|$ denote the number of nodes in the graph. Each link $e \in E$ has capacity u_e. Connection requests arrive sequentially, and we denote the set of all connection requests by K. The i-th request, with *guaranteed* bandwidth[1] r_i, is described by a source-destination pair (s_i, d_i), a pair of non-negative starting and ending times (t_i^s, t_i^f) and a profit b_i. We denote $\mathcal{P}_i$ as the set of feasible paths for connection request i (if i is accepted). We define $f(i,p)$ as the portion of flow i that has been allocated to path $p \in \mathcal{P}_i$. Since we do not deal with fractional routing, $f(i,p) \in \{0,1\}$, i.e., only one path can be used for each flow. The objective of the *offline* admission control problem is to maximize the total profit over the whole sequence of requests, known *a priori*, as follows:

$$\max_{f} \sum_{i \in K} \sum_{p \in \mathcal{P}_i} b_i \, f(i,p) \tag{1}$$

$$\text{s.t.} \sum_{i \in K} \sum_{p \in \mathcal{P}_i | e \in p} f(i,p)\, r_i(t) \leq u_e, \quad \forall e \in E,\ t \geq 0 \tag{2}$$

$$\sum_{p \in \mathcal{P}_i} f(i,p) \leq 1, \qquad \forall i \in K$$

$$f(i,p) \in \{0,1\}.$$

Nevertheless, solving (1) is not possible in practice: the controller receives information on the arrival and departures of requests as soon as they occur, and it has to make a decision on-the-fly. Therefore, the controller needs to decide whether to reject or accept (and on which path, if accepted) a connection request when the request itself materializes. Moreover, the controller is oblivious to future requests and it cannot revisit past decisions.

In the next sections we will show efficient online strategies that show the tendency to reject low-profit connection requests

[1] For simplicity, we will denote $r_i(t) = r_i$ for all $t \in [t_i^s, t_i^f]$ and $r_i(t) = 0$ otherwise.

over highly utilized paths. Indeed, such requests quickly saturate the links and hinder the acceptance of future highly profitable connections.

III. Online Admission Control Algorithms

Traditionally, online algorithms for admission control fall into two main categories: *i)* worst-case and *ii)* average-case. *i)* Worst-case algorithms are characterized by max-min performance guarantees under *specific* worst-case scenarios where a malicious adversary chooses the worst possible sequence of connection requests. Due to their conservative nature, they generally underperform under more standard traffic conditions. On the other hand, *ii)* average-case algorithms show high *expected* performance over random traffic conditions, but cannot guarantee good performance in specific adversarial scenarios.

A. Worst-case Admission Control (AC) Algorithms

Among the worst-case scenario AC algorithm, we first mention AAP algorithm [3], taking admission control decisions based on the current utilization of network links. It computes path costs over a modified network graph where weights depend exponentially on the link utilization. This trick aims at pre-emptively driving traffic away from the links being highly utilized. The acceptance decision is based on a comparison between the cost of accepting the request and the resulting maximum future accepted throughput. Authors of AAP showed that the choice $\mu = 2nTR/r + 1$ and $\rho = nRT$ guarantees a competitive ratio of $O(\log(nT))$ for any sequence of requests with $r_j \leq \min_e \frac{b(e)}{\log(\mu)}$ (R denotes the maximum possible request bandwidth and T the maximum possible request duration). This means that the number of accepted requests is in the worst-case $O(\log(nT))$ smaller than the number of requests that could be routed by the optimal solution of the offline problem in (1).

AAP may not be easy to implement in reality, as it requires *i)* the *a priori* knowledge of requests duration and *ii)* the calculation of an integral which becomes time consuming when the number of flows is large. For these practical reasons, a simpler version of AAP called EXP has been introduced by Gawlick [4]. It uses a simple sum on the instant weights in the acceptance criteria and route requests on the minimum cost path, taking any additive weight (e.g., hop count) of interest in the original graph. Unfortunately, these modifications to AAP invalidate all the competitiveness guarantees.

Buchbinder et al. proposed in [5], [6] a primal-dual framework to derive algorithms for online packing and covering problems with performance guarantees in the worst-case scenario. Such framework developed the theory behind the initial intuition of AAP.

The rationale behind primal-dual AAP (AAP pd) is that, when a demand arrives, the corresponding primal and dual variables are set while maintaining feasibility in both problems and while making sure that the derivative of the primal objective subject to the new dual variable evolves linearly with respect to primal variables, as proposed by [6]. The second constraint guarantees the competitiveness of the algorithm. In the same manner as AAP, the acceptance decisions is taken by comparing the request cost (primal cost increase) and its profit (dual cost increase). We describe the steps of the Primal-Dual version of the AAP algorithm in Alg. 1, by using more efficient incremental updates of the primal variable x_e.

Algorithm 1 Primal-Dual AAP Algorithm [5], [6]

Initialize $x_e = 0$
function ROUTE(request j)
  if $\exists$ a path $P \in \mathcal{P}_j$ of cost < 1 in the graph weighted by x_e **then**
    Route request j on P
    for each edge $e \in P$ **do**
      $x_e = x_e \exp \frac{\ln(1+n).r_j}{u_e} + \frac{1}{n}(\exp \frac{\ln(1+n).r_j}{u_e} - 1)$
    end for
  else
    Reject request j
  end if
end function

B. Beyond Worst-case AC Algorithms

We now turn our attention towards online average-case (also called "stochastic") algorithms, showing good expected performance under random traffic conditions. We present recent algorithms that have been originally proposed for online packing and covering problems with a finite number of objects, and that we have adapted to our admission control problem.

The Primal-Beats-Dual (PBD) algorithm has been introduced by Kesselheim et al. in [7] for online packing problems with a finite number of objects. PBD uses the fractional solution of the packing LP (dual in our case, but primal in [7]) as an advice to select paths along which requests have to be routed. At the j-th request, out of a total number L of requests, PBD computes the fractional solution $\hat{f}$ of the relaxed covering linear program by taking into account the currently active demands, and by scaling the capacity by a factor $\frac{j}{L}$. PBD then choose for j a candidate path p with probability proportional to $\hat{f}$, and accepts demand j only if path p is feasible. The original algorithm is $1 - O(\sqrt{\frac{\log d}{B}})$-competitive for a finite number of requests, where d is the maximum hop count of the routed requests, and B is the capacity ratio between the minimum link capacity and the maximum demand.

PBD requires to solve a large LP at each step, which impacts on its scalability. Agrawal et al. [8] have proposed a fast algorithm with multiplicative updates to solve this issue. This recent algorithm described in Alg. 2 works for general convex problems. It applies to i.i.d. and random order inputs. Agrawal's algorithm solves an online convex problem where the objective function is defined as the difference between the sum of rewards and the cost of accepted flows. The updates of θ and w are standard multiplicative weight updates used in the context of online optimization [9]. Moreover, Agrawal et al. provide an alternative definition of competitive ratio as the ratio between the average reward and the average optimal reward. In this sense, the algorithm is $1 - O(\epsilon)$-competitive for any $\epsilon > 0$ such that $\min(B, k.\mathrm{OPT}) > \log(|E|)/\epsilon^2$.

Algorithm 2 Agrawal's Algorithm [8]

Initialize $\theta_{1,e} = \frac{1}{1+|E|}, \forall e \in E$
Initialize $w_{0,e} = 1, \forall e \in E$
Initialize $Z = \frac{\mathrm{OPT}}{(B/T)}$
function ROUTE(request j)
  Consider G(V,E) with edge cost of $Z\theta_{j,e}, \forall\, e \in E$
  if p is a feasible min cost path in G **then**
    Route request j on p
    Perform the following multiplicative updates:
    $w_{j+1,e} = w_{j,e}(1+\epsilon)^{(r_j - u_e/T)}, \forall\, e \in E$
    $\theta_{j+1,e} = \frac{w_{j,e}}{1+\sum_k w_{j,k}}, \forall e \in E$
  else
    Reject request j
  end if
end function

C. Performance Evaluation

In order to evaluate the performance of the online AC algorithms mentioned above under realistic conditions, we used a real-life dataset captured in 2006 by Uhlig et al. [10] on GEANT, the high bandwidth pan-European research and education backbone. The dataset contains a topology of 22 nodes and 36 high capacity 40G links. We evaluated the percentage of rejected demands under different traffic conditions. The traffic matrix is generated with Poisson demand arrivals at rate λ demands/s. Demands have an equal size of 200 Mb/s and a duration exponentially distributed with mean 30s. We considered two traffic cases: *i) Random* where source-destination pairs are randomly selected, and *ii) Adversarial* where source-destination pairs are selected over the most loaded path at any time. We compared the results with the Greedy policy, that accepts all requests on the minimum cost path whenever there is enough capacity.

The random nature of PBD however yields to some unnecessary rejections in intermediate traffic regimes ($3 \leq \lambda \leq 5$). In more intense traffic conditions ($\lambda \geq 6$), this becomes an advantage and it better load balances traffic around highly utilized paths. In the Adversarial scenario, Agrawal's algorithm plots the best performance, demonstrating its robustness to adversarial behaviors. We remark that the poor performance of EXP are due to the fact that it quickly fills shortest paths and creates bottlenecks.

Remarkably, we notice that there is no algorithm that outperforms all the others under all traffic conditions. This behavior naturally calls for a machine learning technique able to track the best AC algorithm under unknown traffic scenarios.

IV. Admission Control with Experts in SDN

As shown in the previous section, there is no admission control (AC) algorithm that outperforms all the other ones under all traffic conditions. Hence, due to the unpredictable nature of traffic, it proves difficult to know *a priori* the identity of the best algorithm to be utilized. We thus need an *online* meta-algorithm that, based on past decisions and past rewards obtained by the different AC algorithms, is capable to track and follow the advice of the best AC algorithm in hindsight.

We argue that modern SDN control platforms, which are running on top of commodity servers and are built upon cutting-edge distributed computing technologies, enable the execution of different algorithms at the same time to solve a single decision problem.

This setting is classical in machine learning, and it is commonly called *prediction with expert advice* [11]. To be more formal, let us define $m_{i,j}$ as the traffic volume accepted by the AC algorithm i when request j arrives. The meta-algorithm takes its decisions iteratively, based on the profit $m_{i,j'}$ obtained by each AC algorithms i over past decision instants $j' \leq j$.
The bulk of the literature focuses on proving theoretical performance bounds in the basic *non-reactive* scenario where the action taken by the decision maker does not affect the state of the system. Nevertheless, our AC scenario is clearly a *reactive* one, since the decision taken at time t also influences the decisions (and the profits) of the AC algorithms at future time instants $t' > t$.

To this aim, we hence propose to use Strategic Expert meta-Algorithm (SEA), described in Alg. 3. SEA select algorithm i for an increasing number $N_i > 1$ of *consecutive* steps, and does not revisit its choice at each new connection request. This allows each online AC algorithm to approach its asymptotically average performance. Moreover, SEA is only based on the profit effectively obtained by each algorithm when it has been actually selected. In other words, SEA does not exploit the information in hindsight on the profits that would have been obtained if a different algorithm had been used (as for FLA)[2].

SEA's performance guarantees are evaluated in terms of the regret with respect to a (non implementable) *oracle* which

[2]For this reason, SEA also works under the Multi-Armed Bandit setting [11]

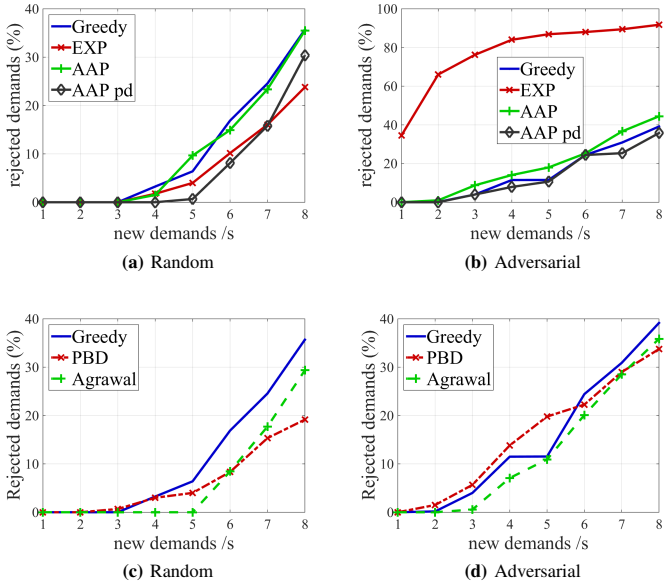


Figure 1: Performance evaluation of various online AC algorithms in terms of rejected demands percentage under random traffic conditions (left) and adversarial ones (right), where source-destination pairs are selected over the most loaded path at any time.

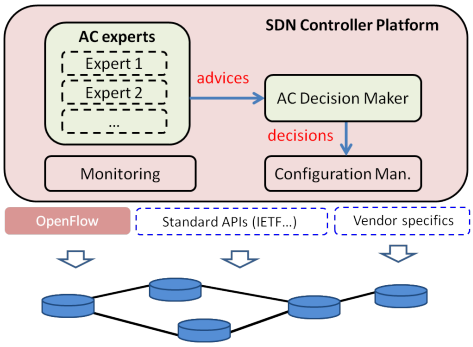


Figure 2: Admission Control (AC) with expert advice in SDN controller platforms.

Algorithm 3 Strategic Expert Meta-Algorithm (SEA) [2]

Set $M_i = N_i = 0$ for each expert i. Set $k = 1$.
function EXPERT SELECTION(Sequence of requests)
With probability $1/k$ perform an exploration phase, namely, choose an expert i from the uniform distribution over $1, \ldots, N$; otherwise, perform an exploitation phase, namely, choose an expert i with maximum M_i. (If such i is not unique, select one out of a uniform distribution)
Set $N_i = N_i + 1$. Follow expert i for the next N_i requests. Denote by $\widetilde{R}$ the average payoff accumulated during the current phase (i.e., these N_i stages), and set

$$M_i = M_i + \frac{2}{N_i + 1}(\widetilde{R} - M_i)$$

Set $k = k + 1$.
end function

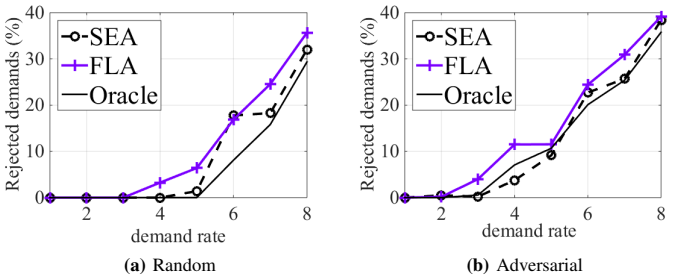
steadily selects the algorithm with best average performance, known beforehand. In our reactive scenario, SEA is in some sense optimal with respect to the new definition of regret hinted above. More specifically, the *expected average* performance of SEA is always superior to the single performance achieved by the algorithm sequentially chosen by SEA itself during its play ([2], Thm. 3.1). Moreover, under some stationary conditions on the system (in our case, of the traffic load on the links) SEA performs on average and asymptotically as well as the oracle ([2], Thm. 5.1).

A. Performance Evaluation

As a benchmark, we compare SEA with the classic Follow-the-Leader meta-Algorithm (FLA) [11]. At each new connection request, FLA prescribes to follow the advice of the AC algorithm that has best performed up to the current decision instant. Although showing good performance in the basic non-reactive scenario, here FLA always fails to track the best expert. More precisely, it always ends up steadily selecting the greedy algorithm: in fact, if we only consider the one-step-ahead profit of the AC algorithms, the greedy algorithm is always at least as good as the other ones.

In Fig. 3 we show the performance of the SEA expert meta-algorithms in terms of their rejected throughput demand. SEA runs on top of three computationally efficient online algorithms: Greedy, AAP-pd and Agrawal's AC algorithm.

As expected, SEA always outperforms the naive FLA. Moreover, under some scenarios SEA even outperforms the oracle. In fact, switching among different expert algorithms sometimes improves the network conditions with respect to the situation where the best algorithm is steadily chosen.



(a) Random **(b)** Adversarial

Figure 3: Performance evaluation of the expert meta-algorithm SEA against the oracle, that knows beforehand the average performance of each algorithm, and the naive FLA. Here, 3 experts to choose among are considered: Greedy, Agrawal and AAP-pd.

V. CONCLUSIONS

The ultimate aim of this paper is to raise the awareness that the new centrally managed SDN architecture, combined with the computational power of the SDN controller, calls for a revisit of admission control algorithms, studied in the computer science literature but never really implemented in practice. We hope that our work will spur new research directions on new AC algorithms for SDN and new expert meta-algorithms which select the proper AC algorithm according to varying traffic conditions.

REFERENCES

[1] B. N. Astuto, M. Mendonça, X. N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 3, pp. 1617 – 1634, 2014.

[2] D. P. de Farias and N. Megiddo, "How to combine expert (and novice) advice when actions impact the environment?" in *Advances in Neural Information Processing Systems*, 2003.

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

[4] R. Gawlick, A. Kamath, S. Plotkin, and K. G. Ramakrishnan, "Routing and admission control in general topology networks," *Tech. rep.*, 1995.

[5] N. Buchbinder and J. Naor, "Improved bounds for online routing and packing via a primal-dual approach." in *Proc. FOCS*, 2006.

[6] N. Buchbinder and J. S. Naor, "Online primal-dual algorithms for covering and packing," *Math. Oper. Res.*, vol. 34, no. 2, May 2009.

[7] T. Kesselheim, A. Tönnis, K. Radke, and B. Vöcking, "Primal beats dual on online packing lps in the random-order model," in *Proc. ACM STOC*, 2014.

[8] S. Agrawal and N. R. Devanur, "Fast algorithms for online stochastic convex programming," in *Proc. ACM SODA*, 2015.

[9] S. Arora, E. Hazan, and S. Kale, "The multiplicative weights update method: a meta-algorithm and applications." *Theory of Computing*, vol. 8, no. 1, pp. 121–164, 2012.

[10] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Computer Communication Rev*, vol. 36, no. 1, 2006.

[11] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. New York, NY, USA: Cambridge University Press, 2006.