Admission Control with Online Algorithms

Jérémie Leguay, Lorenzo Maggi, Moez Draief, Stefano Paris, Symeon Chouvardas

Mathematical and Algorithmic Science Lab, France Research Center, Huawei Technologies Co. Ltd.

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

1 Introduction

Software-Defined Networking (SDN) technologies provide programmable data planes that can be configured from a remote controller. This separation between control and data planes creates 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 loaded. 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 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 extend online algorithms originally conceived for covering and packing problems. 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 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) [dFM03] which shows theoretical guarantees under our reactive scenario, on which there is little research.

2 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[†] r_i, is described by a source-destination pair (s_i,d_i), a pair of

[†] 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.

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:

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

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

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

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

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.

3 Online Admission Control Algorithms

Traditionally, online algorithms for admission control fall into two 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.

3.1 Worst-case Admission Control (AC) Algorithms

Among the worst-case scenario AC algorithm, we first mention AAP algorithm [AAP93], 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 $r_j \le \min_e \frac{b(e)}{\log(\mu)}$ (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 the *a priori* knowledge of requests duration. However, Buchbinder et al. proposed in [BN09] a primal-dual framework to derive a practical algorithm with the same performance guarantees. The rationale behind it 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 [BN09]. 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.

3.2 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. Agrawal et al. [AD15] 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. It solves an online convex problem where the objective function is defined as the difference between the sum of rewards and the cost of accepted objects. The updates of θ and w are standard multiplicative weight updates used in the context of online optimization. Moreover, Agrawal et al. provide an alternative definition of competitive ratio as the

Algorithm 1 Primal-Dual AAP Algorithm [BN09]

```
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**
```

ratio between the average reward and the average optimal reward. In this sense, the algorithm is $1-O(\varepsilon)$-competitive for any $\varepsilon > 0$ such that $\min(B, k.\text{OPT}) > \log(|E|)/\varepsilon^2$ for an simple online covering packing problem. We adapted it to our multi-commodity flow problem.

The Primal-Beats-Dual (PBD) algorithm has also been introduced by Kesselheim et al. in [KTRV14] for online packing problems with a finite number of objects. We considered it in our evaluation but it requires to solve a large LP at each step, which impacts on its scalability.

Algorithm 2 Agrawal's Algorithm [AD15]

```
Initialize $\theta_{1,e} = \frac{1}{1+|E|}, \forall e \in E$. Initialize $w_{0,e} = 1, \forall e \in E$. Initialize $Z = \frac{\text{OPT}}{(B/T)}$
**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:
        $w_{j+1,e} = w_{j,e}(1+\varepsilon)^{(r_j - u_e/T)}, \forall e \in E$    AND    $\theta_{j+1,e} = \frac{w_{j,e}}{1+\sum_k w_{j,k}}, \forall e \in E$
    **else** Reject request $j$
    **end if**
**end function**
```

4 Admission Control with Experts in SDN

As shown in Sec.5, there is no 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, can track and follow the best AC algorithm in hindsight.

This setting is classical in machine learning, and it is called *prediction with expert advice*. 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).

SEA's performance guarantees are evaluated in terms of the regret with respect to a (non implementable) *oracle* which steadily selects the algorithm with best average performance, known beforehand. In our

Algorithm 3 Strategic Expert Meta-Algorithm (SEA) [dFM03]

```
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 $1, \dots, N$; 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
```

reactive scenario and 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 ([dFM03], Thm. 5.1).

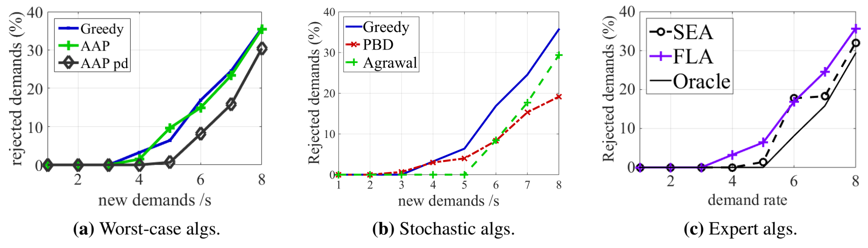
5 Performance Evaluation and Conclusion

To evaluate the algorithms under realistic conditions, we used a dataset captured in 2006 by Uhlig et al. on GEANT. We evaluated the percentage of rejected demands. The traffic matrix is generated with Poisson 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 random source-destination pairs. We compared the results with the Greedy policy, that accepts all requests on the minimum cost path whenever there is enough capacity.

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

In Fig. 1 we show the performance of the SEA expert meta-algorithms which 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.

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.



(a) Worst-case algs. **(b)** Stochastic algs. **(c)** Expert algs.

Figure 1: Performance evaluation in terms of rejected demands percentage. Expert algorithms choose among Greedy, Agrawal and AAP-pd.

References

[AAP93] Baruch Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive on-line routing. In *Proc. FOCS*, 1993.

[AD15] Shipra Agrawal and Nikhil R. Devanur. Fast algorithms for online stochastic convex programming. In *Proc. ACM SODA*, 2015.

[BN09] Niv Buchbinder and Joseph (Seffi) Naor. Online primal-dual algorithms for covering and packing. *Math. Oper. Res.*, 34(2), May 2009.

[dFM03] Daniela Pucci de Farias and Nimrod Megiddo. How to combine expert (and novice) advice when actions impact the environment? In *Proc. NIPS*, 2003.

[KTRV14] Thomas Kesselheim, Andreas Tönnis, Klaus Radke, and Berthold Vöcking. Primal beats dual on online packing lps in the random-order model. In *Proc. ACM STOC*, 2014.