Minimum Cost SDN Routing With Reconfiguration Frequency Constraints

Apostolos Destounis, *Member, IEEE*, Stefano Paris, *Member, IEEE*, Lorenzo Maggi, *Member, IEEE*, Georgios S. Paschos, *Senior Member, IEEE*, and Jérémie Leguay, *Senior Member, IEEE*

***Abstract*— Software-defined network (SDN) controllers include mechanisms to globally reconfigure the network in order to respond to a changing environment. As demands arrive or leave the system, the globally optimum flow configuration changes over time. Although the optimum configuration can be computed with standard iterative methods, convergence may be slower than system variations, and hence it may be preferable to interrupt the solver and restart. In this paper, we focus on the class of iterative solvers with an exponential decrease over time in the optimality gap. Assuming dynamic arrivals and departures of demands, the computed optimality gap at each iteration $Q(t)$ is described by an auto-regressive stochastic process. At each time slot, the controller may choose to: 1) stop the iterative solver and apply the best found configuration to the network or 2) allow the solver to continue the iterations keeping the network in its suboptimal form. Choice 1) reduces the optimality gap leading to smaller routing costs but requires flow reconfiguration which hurts QoS and system stability. To limit the negative impact of reconfigurations, we propose two control policies that minimize the time-average routing cost while respecting a network reconfiguration budget. We experiment with realistic network settings using standard linear programming tools from SDN industry. In the experiments conducted over the GEANT networks and fat tree networks, our policies provide a practical means of keeping the routing cost small within a given reconfiguration constraint.**

***Index Terms*— Communication system traffic control, network optimization, software defined networking.**

I. Introduction

SOFTWARE-DEFINED Network (SDN) architectures unleash the potential to compute routing at a powerful central controller and then reconfigure the network accordingly in real-time [1]. An SDN controller centrally decides traffic engineering (TE) rules to meet performance requirements such as QoS and resilience, which mirrors the past TE techniques [2] but with a new global and online twist. To maintain the best network flow configuration, the SDN controller has to solve variants of the classical Multi-Commodity Flow (MCF) problem [3] in real-time, which may involve millions of variables and constraints in large network instances. As the problem instance itself evolves over time due to time-varying demands, the SDN controller solves a sequence of routing problems and needs to constantly reconsider the flow configuration. Finally, to satisfy application requirements, the controller needs to solve these problems under tight timing constraints.

To cope with the above challenges, state of the art one-shot approaches propose methods to schedule routing configurations [4], [5]. Alternatively, research ideas from the community of online algorithms [6] can be leveraged to yield a static configuration that fares well in the future when unknown demands have arrived. In this work we depart from the static approach and we propose a purely dynamic one. New demands are tentatively treated in a suboptimal way in order to meet timing requirements. Then the controller continuously recomputes global routing and reconfigures the network from time to time with the goal to maintain low running cost.

In particular, this paper considers a general class of iterative routing optimization solvers which yield a sharp improvement of the objective function during the very first iterations and exhibit *diminishing returns*, in the sense that the smaller the optimality gap, the longer it takes to improve it. In our dynamic setting, while the iterative solver converges to the solution, new demands arrive and old demands leave the system, changing the instance of the optimization. These considerations lead naturally to an autoregressive model for the evolution of the optimality gap, whereby the gap decreases exponentially at each iteration of the solver and increases whenever the demands change. Although our model assumes exponential improvement of the optimality gap, and therefore the optimality of our control is established in this context, our simulations with an actual LP solver show that the proposed policies perform well on problems on the GEANT and fat tree networks, which mimic real scenarios, therefore suggesting that they may be good policies to use in practice.

There might be a discrepancy between the best found solution of the solver and the actual network configuration. At each time slot the SDN controller has the option to reconfigure the network flows as per the current computed solution. However, flow reconfigurations degrade QoS and introduce inertia into the system, hence in some time slots it may be preferable to avoid them. Motivating use cases include optical transport networks, where changing the reconfiguration has repercussions on the stability of the optical system and takes a given amount of time and datacenter networks, where flow programing on hardware is slow as recently mentioned in [7]. In these cases, even disruption-free reconfigurations are undesirable if performed frequently. Protocols like TCP could be used to mitigate the effects of path changes, however we consider flow aggregates that from an operator perspective might need to be re-routed to reduce the cost. On the other hand, the solver generates a sequence of routing configurations

Manuscript received June 21, 2017; revised January 1, 2018 and May 4, 2018; accepted May 7, 2018; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Banchs. The work in this paper has been presented at the IEEE International Conference on Computer Communications (INFOCOM), San Fransisco, CA, USA, 2016. *(Corresponding author: Apostolos Destounis.)*

The authors are with the Mathematical and Algorithmic Sciences Laboratory, France Research Center, Huawei Technologies Company Ltd., 92100 Boulogne-Billancourt, France (e-mail: Apostolos.Destounis@hauwei.com).

Digital Object Identifier 10.1109/TNET.2018.2845463

with decreasing cost. Therefore leaving the network in its current status costs money to the operator. For Internet Service Providers (ISPs), network reconfiguration is beneficial to avoid blocking of new connections even though it might result in service disruption [8], [9]. Therefore, reconfiguration of large ISP networks must be performed only when it is beneficial for the system.

The cheapest average cost over time would be obtained by reconfiguring the network according to the solution provided by the solver at each iteration. However, as already mentioned reconfiguring the network at such a frequency comes at the cost of stability. Motivated by the above considerations, in this paper we study a fundamental question for online SDN routing: *when to reconfigure the network in order to minimize routing cost subject to reconfiguration frequency constraints.*

To provide a partial answer we formulate a stochastic optimization problem where we want to minimize the actual network cost by selecting the reconfiguration instances subject to a budget. The budget refers to a time-average constraint on the frequency of network reconfigurations, so that no more than $h_{max} < 1$ network reconfigurations occur per iteration of the solver. Thinking of the reconfiguration as sampling, the problem refers to sampling an autoregressive process with a given sampling frequency so that the extra surcharge incurred by non-sampled instances is minimized.

We restrict ourselves to a subclass of control policies that always "sample" after the optimality gap has increased. This constraint leads to a renewal structure, which in turn permits the characterization of the best policy of the subclass. The policy works in two levels: (i) a virtual queue captures the price of sampling as it evolves over renewal frames, and (ii) a dynamic programming method is used to find the optimal "sampling" within a frame subject to the current price. Finally, we have tested our approaches on scenarios that mimic real scenarios with a network and traffic patterns widely used by the research community in order to compare the performance of the optimal renewal policy to a heuristic method which minimizes a drift-plus penalty function at each time slot and a policy that periodically reconfigures the network.

II. System Model

In this section, we present the routing system model that we consider.

A. System Architecture

We consider an online routing system with two main stages 1) a stage where we accept new demands and 2) a stage where we re-consider flow configuration over time. This two-step structure is typical for admission control systems, since solving a multi-commodity flow can take a significant amount of time (typically minutes for large network instances) and delaying the establishment of a path until this computation is over can lead to undesirable delays. The target is to minimize routing cost which is motivated in the domain of data center interconnection or enterprise networks, where the goal is leasing the cheapest connections from Internet Service Providers.

Fast Connection Setup (FCS): When connection requests arrive at ingress nodes, the controller finds a feasible path satisfying multiple constraints (e.g., capacity, and QoS). For QoS purposes, the time requirements for finding a solution might be very strict. Hence at this stage, the goal is not to optimize the network, but rather to find a quick feasible solution. Example of FCS methods include (constrained) shortest path algorithms which run on residual graphs.

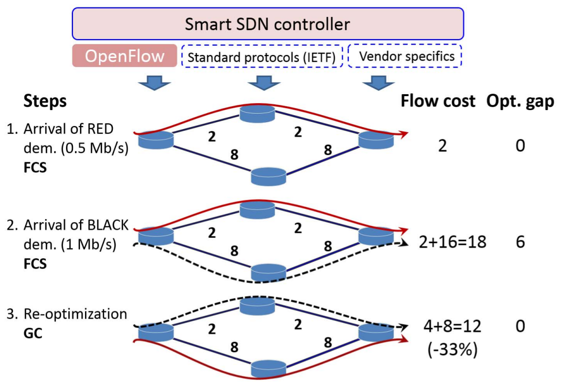

Fig. 1. Example of an online SDN routing optimization with two demands. Edge labels represent link costs. All links have capacity of 1 Mb/s. The cost evolution is computed according to the allocation performed by FCS and GC.

Garbage Collection (GC) of Network Resources: The sequence of sub-optimal network configurations obtained from FCS, poses significant concerns on the evolution over time of the global objective function. Therefore, periodic or event-based reconfiguration of the overall flow can help reduce the optimality gap. We call this mechanism *Garbage Collection (GC) of network resources* since it mirrors the way a Java virtual machine collects garbages and reorganizes the memory. Example of GC methods include algorithms that solve the minimum cost MCF problem.

Fig. 1 shows an example of an SDN controller which strives to minimize cost. The FCS uses a shortest path algorithm, while the GC uses a min-cost MCF solver. Edge labels indicate the link costs, while all link capacities are 1Mb/s. In the example two demands arrive subsequently. First, the red demand arrives and requires 0.5Mb/s, while the FCS allocates it to the low cost path. Then the black demand arrives and requires 1Mb/s. Since the low cost path does not have enough remaining capacity, the FCS allocates the black demand to the high cost path. The network configuration after step 2 is suboptimal and leads to higher running costs. Hence, the controller employs GC, which computes the optimal solution shown at step 3. The SDN controller ultimately reconfigures the network with the optimal solution, saving in this way 33% of the cost.

B. Min-Cost Optimization

In the following, we turn our attention to the GC step and explain our model for globally optimize routing.

We model the network infrastructure with an undirected graph $G = (\mathcal{N}, \mathcal{L})$, where set $\mathcal{N}$ represents the set of network nodes and set $\mathcal{L}$ models the links $e = \{i, j\}$, connecting network nodes $i, j \in \mathcal{N}$. Each link $e \in \mathcal{L}$ has a limited capacity b_e and a cost c_e, which refer to the maximum amount of flow that can be routed and the price paid per unit of routed flow, respectively.

A unicast demand $k \in \mathcal{K}$ is identified by a source-destination pair $(s_k, d_k) \in \mathcal{N}^2$, and the amount of traffic r_k that has to be transmitted from s_k to d_k. The set $\mathcal{K}$ represents the active demands on this problem instance that need to be routed through the network. To satisfy the demands in the cheapest way, the controller needs to solve an evolving instance of the minimum cost Multi-Commodity Flow (MCF) problem which can be formulated at a given time as the linear program (1)-(3), where real variables $(x_p)_{p\in\mathcal{P}}$ and $(y_e)_{e\in\mathcal{L}}$ represent the path and link utilization and take values in $[0,1]$. In this model, P is the set of all network paths, while $P_k \subseteq P$ represents the set of all paths which connects the source s_k to the destination d_k of a demand $k \in \mathcal{K}$. In contrast, the set $P_e \subseteq P$ contains all paths that use link $e \in \mathcal{L}$.

$$C^{OPT} = \min_{(x_p),(y_e)} \sum_{e\in\mathcal{L}} y_e c_e \tag{1}$$

$$\text{s.t.} \sum_{p\in P_k} x_p = 1 \quad \forall k \in \mathcal{K} \tag{2}$$

$$\sum_{k\in\mathcal{K}} \sum_{p\in P_e : k\in P_k} x_p r_k \le y_e b_e \quad \forall e \in \mathcal{L}. \tag{3}$$

The objective function (1) models the overall price paid for using the network links and results in a resource allocation that spreads the traffic across network links and keep the link flows away from maximum link capacity [10]. The constraints (2) ensure that the entire demand r_k is routed through a set of paths with routing splits $x_p r_k$, and the constraints (3) are the link capacity constraints.

C. Iterative Solver With Diminishing Returns

Due to the immense size of the problem instance (network graph and set of demands), we resort to the *Column Generation (CG)* method coupled with the simplex algorithm, which has been proposed as a key method for solving large MCF problems [11]. Such an approach is powerful because it iteratively improves the solution by considering only a small number of variables at each step, without considering the set of all possible network paths, whose size is exponential. To simplify the analysis, we assume that the optimality gap is reduced exponentially fast. Our analysis applies to other iterative techniques with exponential convergence rate, including the full gradient and the stochastic gradient methods for strongly convex objective functions [12]. We leave as future work the analysis of solution techniques with sublinear and linear convergence rates [13], [14].

At every iteration $t = 1, 2, 3, \ldots$ of the CG solver we obtain a feasible flow solution with cost $C(t) \ge C^{OPT}(t)$, where $C^{OPT}(t)$ is the cost obtained when the solver reaches the optimum. Let us denote the absolute optimality gap with $Q(t)$:

$$Q(t) \triangleq C(t) - C^{OPT}(t) \ge 0.$$

$Q(t)$ is an indication of the amount of surcharge an operator needs to pay for not having the network completely optimized at time t, hence we would like $Q(t)$ to be as small as possible. Although the iterations monotonically decrease the optimality gap $Q(t)$, we observe "diminishing returns", in the sense that the smaller the gap, the longer it takes to improve it. To model this situation we assume that the evolution of the optimality gap follows an exponential decay:

$$Q(t+1) = (1-\rho)Q(t), \tag{4}$$

where $\rho \in (0,1)$ is a constant that relates the volume of the next improvement to current optimality gap values. This corresponds to an exponential improvement function of the type $(\frac{1}{1-\rho})^{-t}, t = 1, 2, \ldots$.

Throughout the paper we assume that each iteration has the same duration, and matches squarely a time slot. In practice iterations take a random amount of time, however they consist on operations that do not fluctuate a lot, especially for large network instances (i.e. matrix inversions and shortest paths). Therefore assuming that each iteration has the average duration is practically meaningful.

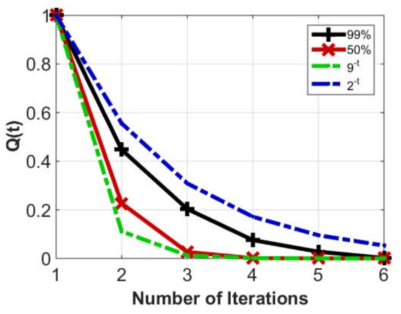

Fig. 2. Evolution of the optimality gap Q(t) in the GEANT scenario. The 50 and 99 percentiles are computed considering 500 simulations.

We validate our model (4) using the CG solver in different traffic conditions in the GEANT network topology [15]. The traffic matrix is generated by randomly selecting source-destination pairs and rescaling their demand according to the capacity of the bottleneck link. Fig. 2 shows the evolution of the optimality gap as a function of the number of iterations. Specifically, red and black solid lines represents 50 and 99 percentiles obtained over 500 simulations. The optimality gap $Q(t)$ obtained in numerical simulations is always lower and upper bounded by two exponential functions, namely 9^{-t} and 2^{-t}, which are depicted as green and blue dashed lines in the figure.

D. Modeling Event Arrivals

In the above section we defined the MCF instance for a given set of demands $\mathcal{K}$. Next, we consider the arrival of "new events" in the system which potentially lead to a different set of demands $\mathcal{K}(t)$. The new events correspond to arrivals of new demands, or departure of old demands and in both cases they result in a "jump" in the optimality gap $Q(t)$, due to the suboptimal network status resulting from the occurred event.[1] We remark that we consider flow aggregates rather than end-to-end connections, therefore considering the nominal rate instead of the actual consumed bandwidth is more realistic. To simplify the considerations here, we assume that all events incur the same "jump" (i.e., the same extra cost), which is denoted by e. Our work can be extended to consider more elaborate models. More precisely, we describe the influence of new events (arrivals and departures of demands) on the optimality gap $Q(t+1)$ via an additive term $A(t)$ which follows an *i.i.d.* stochastic process with mean $E[A(t)] = \lambda$, and variance $\text{Var}[A(t)] = \sigma_A^2$, both finite.

The controller will perform the t-th iteration of the GC procedure by solving the $\mathcal{K}(t)$–instance of the optimization (1)-(3). The optimality gap evolution can now be rewritten to include the addition of the cost due to changing demands

$$Q(t+1) = (1-\rho)Q(t) + eA(t). \tag{5}$$

This is a first order auto-regressive stochastic process with discrete non-Gaussian disturbance. It is known that $Q(t)$ is

[1]A newly arriving demand is treated by the FCS procedure. We will assume that such a path can always be found. This assumption is not restrictive in practice because (i) systems are often overprovisioned, and (ii) in the case of a network overload, a congestion controller may reject some demands to make the system feasible.

strongly stable as long as $\rho > 0$ (see for example [16, Sec. 2.1.1]). Conclusively, the mean optimality gap $Q(t)$ remains finite irrespective of how high the arrival rate of events is, or how slow the exponential slope of our solver is. We can derive the stationary mean and variance:

$$\overline{Q} = \lim_{t\to\infty} \mathbb{E}\left[Q(t)\right] = \frac{e\lambda}{\rho},$$
$$\sigma_Q^2 = \lim_{t\to\infty} \text{Var}[Q(t)] = \frac{e^2\sigma_A^2}{\rho(2-\rho)}.$$

However, note that $\overline{Q}$ is in fact the gap between the optimal routing and the computed solution at the solver. To achieve this routing cost in the real network, the SDN controller must reconfigure the network at each iteration of the solver. The objective of the next sections is to derive control policies that yield small optimality gap without reconfiguring continuously the network.

III. Control Problem Formulation

Flow reconfigurations take time and cause small disturbances that affect the QoS, hence we would like to minimize them. However, the goal of minimizing flow reconfigurations conflicts with the goal of minimizing the routing cost. For example, always applying the new solver iteration yields the best possible average optimality gap $\overline{Q}$, but incurs at the same time the maximum number of reconfigurations (one every iteration). On the other hand, we may decide to periodically reconfigure once every ten iterations, which limits the reconfiguration frequency to 10% but results in operational cost higher than $\overline{Q}$, since in many iterations improved solutions are available but not applied to the network.

We consider a controller which at each time slot decides whether to use the improved solution of the iterative solver or not. By selecting $u(t) = 1$ the controller decides to "spend" one reconfiguration to bring the network into a form that agrees with the current output of the iterative solver. If $u(t) = 0$ is selected, then the network is left untouched.[2]

While the solver's solution has an optimality gap $Q(t)$ which evolves according to (5), the actual operational optimality gap is higher when we do not apply the best solution at each iteration, since the distance from the optimal point increases as illustrated in Fig. 3, where the *"surcharge cost"* between the solid black line (the output from the solver) and dashed green line (the cost of the current network configuration) keeps increasing as long as the $u(t) = 0$. We denote by $S(t)$ the surcharge cost at time slot t caused by not using the most recent improved solution. The evolution is given by

$$S(t) = \left(S(t-1) + \rho\, Q(t-1)\right)(1 - u(t)), \tag{6}$$

where we note that (i) if $u(t) = 1$ then $S(t)$ becomes zero, while (ii) if $u(t) = 0$ then $S(t)$ increases by a term $\rho Q(t-1)$ which is the new cost improvement computed by the solver but not applied to the network.

The objective is to find a control policy that selects $u(t)$ at each time slot in order to minimize the average surcharge cost subject to a constraint on the average number of flow reconfigurations (i.e., the number of times we select $u(t) = 1$). We may formalize a stochastic optimization problem as follows.

[2]Without loss of generality, we leave for future work other granularities of reconfiguration like per-device and per-flow.

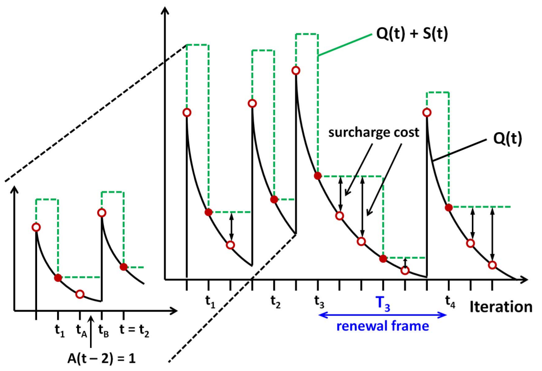

Fig. 3. Evolution of the optimality gap obtained using the MCF solver (red points) and the control policy with renewals (dashed green curve). The black curve is the interpolation of the points computed by the MCF solver at each iteration. Solid dots show where the solution computed by the solver is applied.

Problem $\mathcal{P}$ (Minimum Surcharge Subject to Reconfiguration Frequency):

$$\min_{\{u(t)\}_t} \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\{S(t)\} \tag{7}$$

$$\text{s.t.} \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\{u(t)\} \leq h_{max}. \tag{8}$$

where $0 < h_{max} \leq 1$ is the constraint on the frequency of time slots where we reconfigure. There exist potentially other objectives or constraints with which we would like to generalize problem $\mathcal{P}$, such as for example to minimize the number of routers that become reconfigured. In this paper we focus on the solution of $\mathcal{P}$ which is fundamental for SDN controllers.

Problem $\mathcal{P}$ is a stochastic online optimization that admits a class of policies Π. Here, we are interested in causal policies, that is policies with no knowledge/prediction about the time instances of future arrivals. A policy π will be called "feasible" if the long-term average reconfiguration rate that results from applying this policy to the system satisfies the constraint (8). In more detail, at the beginning of time slot t, the controller is given the optimality gap $Q(t)$ as computed by the solver. Additionally, the controller knows the past evolution of the system, i.e., the values $(A(\tau), Q(\tau)), \forall \tau < t$. Any new flow arrivals/departures that occur at time slot t are taken into account for computing the (new) optimal flow allocation from time slot $t+1$ and onwards. The system is then Markovian, with state vector

$$\mathcal{S}(t) = [Q(t), S(t-1), Q(t-1)]$$

(the last two state variables are needed to find the cost (6) at each time slot) for $t = 1, 2, 3, \ldots$ and the problem is a constrained Markov Decision Process (MDP) with infinite horizon and time-average cost and constraint, where the system dynamics and the cost at every slot are given by (5) and (6). It is known [17, Th. 6.2] that we can restrict to the set of Markov controls (i.e. one that makes the control decision at time slot t based only on $\mathcal{S}(t)$ and knowledge of statistics of the arrivals) without any loss of optimality; this means that policies that use older information or are allowed to depend on the time slot index do not lead to a better performance than those that make decisions based only on the current state of the system and its statistics. Nevertheless, since $\mathcal{P}$ includes the solution of a constrained MDP with infinite state space, it is not efficiently solvable using standard methods. In the next section we restrict our attention to a subclass of policies that invoke a renewal structure and include an efficient solution.

As before, we have

$$Q(t+1) = (1-\rho)Q(t) + eA(t).$$

The evolution of the operational cost $X(t)$ is then

$$X^\pi(t+1) = (1-\rho)Q(t)I^\pi(t) + X^\pi(t)(1-I^\pi(t)) + eA(t), \tag{9}$$

where $I^\pi(t) \in \{0,1\}$, and π is used to denote the control policy we are using. We define the control policy to be a mapping from the state of the system $(X^\pi(t), Q(t), D(t))$ to the action set $\{0,1\}$.

Based on the simple model explained above, the total number of reconfigurations under policy π by time t is simply

$$R^\pi(t) = h\sum_{\tau=1}^{t} I^\pi(\tau)$$

We define the frequency of reconfigurations as

$$\overline{R}^\pi \triangleq \limsup_{t\to\infty} \frac{E[R^\pi(t)]}{t}$$

We are interested in the following optimization

$$\min_\pi \limsup_{t\to\infty} \frac{\sum_{\tau=1}^{t} E[X^\pi(\tau)]}{t} \tag{10}$$

$$\text{s.t. } \overline{R}^\pi \le r_{\max} \tag{11}$$

where we pick a control policy with decisions $\{I^\pi(t)\}, t = 1,2,\ldots$ in online fashion to minimize the average operational error $\limsup_{t\to\infty} \frac{\sum_{\tau=1}^{t} E[X^\pi(\tau)]}{t}$ while keeping the reconfiguration frequency less than $r_{\max}$.

Even in the case where centralized algorithms can optimize paths within seconds as shown in [18] and [19], the problem of limiting the number of reconfigurations in a dynamic scenario remains. If the controller quickly (ideally instantaneously) computes an optimal allocation every time a demand join or leave the system, the evolution of the optimality gap $Q(t)$ can be modeled as a sequence of Dirac delta functions with the amplitude equal to the difference between the cost paid before and after the optimization. In this case, the problem boils down to dynamically select the largest impulses according to a target reconfiguration frequency set by the operator.

IV. Control Policy With Renewals

We say that a Markov process has a renewal structure if the system visits states where it "*statistically restarts*", cf. [20, Ch. 7]. In more detail, our system (described by $\mathcal{S}(t)$) has a renewal structure if there exists a state s such that the return time to that state, as well as the cost and number of reconfigurations during this are identically distributed and independent of the past history. The time period between two successive returns to that state will be called a "renewal frame". Such systems are much easier to analyze since the online problem can be broken down to smaller control problems within each renewal frame. Unfortunately, the problem $\mathcal{P}$ we are interested in does not have such a renewal structure. Our approach in this section is to use an artificial constraint to invoke a renewal structure into our system. We will later show that subject to this constraint, there is an efficient policy that optimizes problem $\mathcal{P}$, thus providing a complete performance characterization of the system under this constraint.

Policy Constraint 1 (Reconfigure After Demands Change): Consider the constrained set of policies $\Pi_c \subset \Pi$. For any policy $\pi \in \Pi_c$ the reconfiguration is applied whenever there was a change in the demands, i.e., at any t

$$u^\pi(t) = 1 \quad \text{if} \quad A(t-2) > 0, \quad \forall t, \quad \forall \pi \in \Pi_c.$$

The delay of 2 time slots in $A(t-2)$ relates to our notation and ensures that the controller is forced to reconfigure at the first iteration after the arrivals have been considered, see the blow-up box of Figure 3. In the figure, $A(t_A)$ represents arrivals in $[t_A; t_B)$ that are soon handled by FCS using the network configuration in state $\mathcal{S}(t_A)$. These arrivals are handled by GC at time $t_{A+1} = t_B$ (i.e., their paths are integrated in the problem (1)-(3) at t_{A+1}) and only at $t_A + 2$ the benefit of GC is available. Intuitively the constraint is justified since the first step of the solver has the steepest decrease in the optimality gap and hence by choosing $u(t) = 1$ in such time slots a great extra cost is avoided.

Note that for Π_c to be feasible, it must be $\mathbb{P}\{A(t) > 0\} < h_{max}$. If otherwise, the reconfiguration constraint (8) will be violated by any policy in Π_c.

Under a policy $\pi \in \Pi_c$ the time interval between two consecutive time slots t with the property "$A(t-2) > 0$" constitutes a renewal frame. Let $t_0 = 0$, denote t_n the $n-$th slot satisfying $A(t-2) > 0$, and denote $T_n = t_{n+1} - t_n$ the number of time slots in the $n-$th frame (note that T_n is a random variable).

In the remaining of this section we propose RP (Renewal Policy), a policy that satisfies the Constraint 1 and hence RP $\in \Pi_c$. By exploiting the renewal structure that policies in Π_c induce, we will prove that RP is a feasible policy that achieves near-optimal value of (7) over all policies in class Π_c.

*A. Renewal Policy (*RP*)*

A standard way to solve stochastic optimization problems involves the use of policies that greedily balance the penalty in a time slot with the instability of virtual queues; see the *drift-plus-penalty algorithm* [20]. In fact, this dynamic algorithm can also be applied to systems with renewals. In this case, the policy keeps track of the budget spent thus far and puts a price per unit of budget to be spent in the next renewal frame; in our system we have a price per reconfiguration. This budgeting method guarantees the satisfaction of the limit constraint (8). Within the renewal frame, an oracle policy is used to decide the control taking into account the price per reconfiguration. When the uncontrolled states of the system are i.i.d. renewals, the problem is tackled in [20] and [21]. Here we extend it to our non-i.i.d. framework.

To introduce a "price" into our system we define a virtual queue $U(t_n)$ whose value we track only at the end of each renewal frame:

$$U(t_{n+1}) = \left[U(t_n) - T_n h_{max} + \sum_{t=t_n+1}^{t_n+T_n} u(t)\right]^+, \tag{12}$$

where the term $\sum_{t=t_n+1}^{t_n+T_n} u(t)$ equals the number of reconfigurations used in the last renewal frame, while $T_n h_{max}$ is the product of the last renewal frame length T_n times the average constraint on reconfigurations from (8).

Mean rate stability of $U(t_n)$, that is

$$\lim_{N\to\infty} \frac{\mathbb{E}\{U(t_{N-1})\}}{N} = 0,$$

see [20, p. 17], guarantees the property "arrivals $\leq$ departures", hence

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{u(t)\} \leq h_{max}$$

which satisfies (8). Hence, any policy that stabilizes $U(t_n)$ is a feasible policy. To track in real-time the stability of $U(t_n)$, we define the quadratic Lyapunov drift as

$$\Delta(\mathbf{s}, U(t_n)) = \mathbb{E}\left[U^2(t_n + T_n) - U^2(t_n) \,|\mathbf{S}(t_n) = \mathbf{s}, U(t_n)\right]$$

For a given policy, $\Delta(\mathbf{s}, U(t_n))$ measures how the Lyapunov function $L(x) = x^2$ of the changes over a renewal frame. The idea is that the Lyapunov function measures how large is the queue $U(t)$ and its drift is a means of quantifying the immediate impact of a policy on the constraint satisfaction. Any policy that yields bounded $\Delta(\mathbf{s}, U(t_n))$ for any $(\mathbf{s}, U(t_n))$ can be shown to stabilize $U(t_n)$ in the mean rate sense, hence such a policy is feasible.

Apart from feasibility, we are also interested in minimizing the cost (7), thus we combine the drift $\Delta((\mathbf{s}, U(t_n))$ with the cumulative penalty (7) within a renewal frame:

$$DPP(\mathbf{s}, U(t_n)) = \Delta(\mathbf{s}, U(t_n)) + V\mathbb{E}\left\{\sum_{t=t_n+1}^{t_n+T_n} S(t)\right\}, \quad (13)$$

where V is a constant that plays an important role in any Lyapunov optimization-based scheme, since it controls the tradeoff between utility optimality and expected delay [20, p. 48-49]. The metric (13) is often called *Drift-Plus-Penalty* (DPP). Minimizing drift-plus-penalty includes two conflicting goals, (i) to minimize the drift $\Delta(\mathbf{s}, U(t_n))$ thereby satisfying in the long term the constraint (8), or (ii) to greedily minimize the penalty in the next renewal frame. In fact, $u(t) = 0$ favors (i) and $u(t) = 1$ favors (ii). A policy that minimizes drift-plus-penalty at each renewal frame is essentially striking a good balance between the two in a greedy fashion.

If we perform standard calculations and expand (13) we obtain an upper bound expression on $DPP(\mathbf{s}, U(t_n))$ which is optimized at every renewal by a policy that solves the following optimization problem:

$$J^*(t_n; V) = \min_u \mathbb{E}\left\{\sum_{t=t_n+1}^{t_n+T_n} VS(t) + U(t_n)u(t)\right\} \quad (14)$$

This optimization seeks to find an appropriate sequence of controls $u(t_n + 1), \ldots, u(t_n + T_n)$ within the n–th renewal frame to balance the expected price of the extra cost $VS(t)$ with the price of reconfigurations $U(t_n)u(t)$.

Next, we propose RP which is designed to solve (14) at every renewal frame, subject to Constraint 1. The RP policy works as follows. Following a time slot with an arrival, it always selects $u(t) = 1$ and notes the beginning of a renewal frame. The virtual queue (12) is used to track the evolution of the price across renewal frames. At the beginning of the n–th renewal RP observes the value $U(t_n)$ and calls a routine ϵ–$\mathsf{OPT}(U(t_n))$ which approximately solves (14). The routine returns an infinite sequence of controls $u(t_n{+}1), u(t_n{+}2), \ldots$. RP uses this sequence until the n–th renewal is over at which point it discards the remain subsequence and starts over. The next section describes the RP policy and proves its near-optimal performance from the class Π_c.

B. Performance Analysis of RP

In this subsection we build intuition about why RP is optimal in class Π_c.

1) $Q(t)$ Is an Ergodic Markov Chain: For the analysis we will make the following assumption

Assumption 2: If $Q(t) = q_\epsilon$ for some small constant $0 < q_\epsilon \ll 1$, then we can approximate $Q(t) = 0$.

In practice the assumption implies that tiny optimality gaps may be disregarded, hence it is mild. Technically, it is used to prove the following intermediate result:

Lemma 3: The process $Q(t)$ evolves in a countable state space. In addition, under Assumption 2, $Q(t)$ is irreducible and aperiodic.

Proof: Refer to Appendix A for the proof. ■

2) Optimal Cost of Π_c: Consider the optimization problem $\mathcal{P}$ over all policies belonging to Π_c. The problem can still be seen as a constrained MDP, with state space $\widetilde{\mathcal{S}}(t) = [Q(t), S(t-1), Q(t-1), A(t-2)]$. From the general theory of constrained Markov Decision Processes (see e.g. [17]) we can show the following:

Renewal Policy (RP)

Parameter Selection. $t_0 = 0, V = 100, U(t_0) = V$.
Renewal frame. Check every time slot. If at slot $t - 2$ we had $A(t - 2) > 0$, then slot t is the beginning of a renewal frame. This recursively defines a sequence of time instances $t_0, t_1, \ldots, t_n, \ldots$ that indicate the beginning of renewals, where the slots $\{t_n, \ldots, t_{n+1} - 1\}$ define the n–th renewal frame. Within n–th renewal the policy performs the following steps.
Step 1: Constraint 1. Choose $u(t_n) = 1$.
Step 2: Price Update. Update the price

$$U(t_n) = \left[U(t_{n-1}) - T_{n-1}h_{max} + \sum_{\tau=t_{n-1}+1}^{t_{n-1}+T_{n-1}} u(\tau)\right]^+.$$

Step 3: Solution of (14)**.** Call routine ϵ–$\mathsf{OPT}(U(t_n))$ to obtain an infinite sequence of controls $\mathcal{U}^\epsilon = \{u(t_n + 1), u(t_n + 2), \ldots\}$ that is ϵ-optimal with respect to the problem:

$$\min_u \mathbb{E}\left\{\sum_{t=t_n+1}^{t_n+T_n} VS(t) + U(t_n)u(t).\right\}$$

We describe routine ϵ–$\mathsf{OPT}(U(t_n))$ below.
Step 4: Actions within the renewal. For time slots $\{t_n + 1, \ldots, t_{n+1} - 1\}$ choose the controls according to $\mathcal{U}^\epsilon$.

Lemma 4: The optimal policy $\pi^* \in \Pi_c$ is one where the controller is a (possibly randomized) function of only $[Q(t), S(t-1), Q(t-1)]$ if $A(t-2) = 0$ and $u(t) = 1$ if $A(t-2) > 0$ (the latter is due to the constraint of class Π_c).

Denote the incurred cost of π^* with $\overline{S}^*$, where

$$\overline{S}^* = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{S^{\pi^*}(t)\}.$$

Although we cannot directly characterize π^*, we will use its existence to prove the optimality of RP.

3) Analysis of RP*:* The setting we have here is slightly different with respect to optimization over renewal processes as presented in [20] and [21], since $Q(t_{n+1})$ depends on $Q(t_n)$ (in the references the uncontrolled process is i.i.d. at renewal periods). In order to resolve this, we use the idea of $T-$slot drifts as employed in [20] and [22] for Markovian dynamics of the uncontrolled state processes, extending it over renewal frames this time.

Specifically, define $I(t) = \mathbb{1}_{\{A(t-2)>0\}}$ and the Markov chain $\mathbf{Z}(n) = [Q(t_n), I(t_n)]$ (note that $S(t_n) = 0$ since $u(t_n) = 1\ \forall n$, therefore $S(t_n-1), Q(t_n-1)$ do not matter). Then choose any of the states $\mathbf{z} = [q, 1]$. Since we also need an arrival to happen at states $\mathbf{z}$, we are essentially looking at the beginning of renewal frames of the initial problem. Starting from a point in time where $\mathbf{Z}(n) = [q, 1]$ and denoting N_q the return time to that state, we consider a variable drift for these N_q renewal frames. Assuming a routine that solves (14) with an ϵ error and standard analysis from Lyapunov optimization theory [20], we compare the drift-plus-penalty of RP to π^* to obtain

$$\overline{S}^{\mathsf{RP}} \leq \frac{\epsilon + B_q}{V} + \overline{S}^*. \tag{15}$$

where

$$B_q = \frac{\mathbb{E}\left\{T_n^2\right\}(1-h_{max})}{2}\mathbb{E}\left\{N_q\right\} + \frac{\mathbb{E}\left\{T_n\right\}(1+h_{max})}{2}\mathbb{E}\left\{N_q^2 - N_q\right\}.$$

For every $q \in \mathcal{Q}$ we prove in Appendix B that the return time has bounded second moment:

Lemma 5: $\mathbb{E}\left\{N_q^2\right\} < \infty, \forall q \in \mathcal{Q}$.

which asserts that $B_q < \infty$. Hence (15) shows that the cost achieved by RP is near-optimal (consider for example large values of V). Feasibility is shown by using the same drift analysis to prove that $U(t_n)$ is mean rate stable. Formally:

*Theorem 6 (*RP *Is Near-Optimal in* Π_c*):* Let $J^*(t_n; V)$ be the optimal value of (14) and $\mathcal{U}^\epsilon$ be an $\epsilon-$optimal control policy for every renewal frame, i.e. a control policy that for every renewal frame n achieves a cost $J(t_n; V) \leq J^*(t_n; V)+\epsilon$, for a constant $\epsilon > 0$. Then, for RP the following hold:

1) The constraint of eq. (8) is satisfied.
2) $\overline{S}^{\mathsf{RP}} \leq \overline{S}^{\pi^*} + \frac{B+\epsilon}{V}$ (i.e. the average cost (7) is $O(1/V)$ close to the optimal one, achieved by π^*), where $B = \min_{q\in\mathcal{Q}} [B_q]$ is a finite constant.

Proof: Refer to Appendix C for the full proof of the Theorem. ■

C. A Polynomial Approximation Algorithm Within Renewal Periods

Above we required a routine that approximately solves (14) at every renewal period. We now describe such routine, which is based on a Dynamic Programming (DP) optimization technique. As we will show, the proposed DP method is particularly appealing since it produces a polynomial approximation scheme.

To begin with, we define the following auxiliary variable:

$$\beta = 1 - \mathbb{P}\{A(t) > 0\}, \quad \forall t. \tag{16}$$

Next, we observe that the stochastic control problem (14) is equivalent to the deterministic control problem of finding a control sequence $u(t) \in \{0, 1\}$ for $t = t_n + k$, with $k \geq 1$, that solves the following program for each t_n:

$$\min_{\{u(t)\}_{t>t_n}} \sum_{t=t_n+1}^{\infty} \beta^{t-t_n}\left(VS(t) + U(t_n)u(t)\right) := g(u), \tag{17}$$

where the system evolves over time steps $t \geq t_n$ according to the laws described in (4) and (6). For notation simplicity, we drop the dependency of u and g on the time instance t_n.

1) ϵ-Optimality Formulation: In order to come up with a ϵ-optimal strategy for the problem (17) we simply truncate at time step T the infinite time window over which the optimal control is evaluated. In Lemma 7 below we show that the correctness of this approach is ensured by the boundedness of the cost function. Moreover, the ϵ-optimal truncated horizon T is a logarithmic function of ϵ^{-1}. This observation will prove to be crucial to show that the resulting policy provides a polynomial approximation scheme for the problem (17).

Lemma 7: Let $\epsilon > 0$. Let $\widetilde{u}'$ be the optimal solution of the following T-truncated version of (17):

$$\begin{aligned} \widetilde{u}' = \underset{u'}{\operatorname{argmin}} & \sum_{k=1}^{T} \beta^k \left(VS_k + U(t_n)u'_k\right) \\ \text{s.t. } & Q_k = (1-\rho)Q_{k-1}, && 1 \leq k \leq T \\ & S_k = (1-u'_k)\left(S_{k-1} + \rho Q_{k-1}\right), && 1 \leq k \leq T \\ & u'_k \in \{0, 1\}, && 1 \leq k \leq T \\ & S_0 = 0, \quad Q_0 = Q(t_n) \end{aligned} \tag{18}$$

where the truncated time horizon T is computed as

$$T = \left\lceil \log\left(\frac{(VQ(t_n) + U(t_n))}{\epsilon(1-\beta)}\right) / \log(\beta^{-1}) - 1 \right\rceil. \tag{19}$$

Then, any policy $\overline{u}'$ such that $\overline{u}'(t_n + k) = \widetilde{u}'_k$ for $1 \leq k \leq T$ is ϵ-optimal, i.e., $g(\overline{u}') \leq g(\underline{u}') + \epsilon$, where $\underline{u}'$ is optimal for the original problem in (17). □

The truncated problem defined in (18) is a *deterministic* control problem in finite time horizon and finite state space. We next show that it can be efficiently solved via a Dynamic Programming (DP) algorithm, which results in a polynomial approximation scheme for the original problem (17).

ϵ-optimal routine ϵ–OPT$(U(t_n))$

Parameters. Choose optimality gap ϵ, probability of arrivals β, constant V.
Input. Price $U(t_n)$, optimality gap $Q(t_n)$.
Initialization. Compute T as in (19). Define the initial state as $s_{0,1} = \bar{s}_0 = 0$ and its relative accumulated cost $c(s_{0,1}) = 0$.
Procedure. For $k = 1, \ldots, T$:

```
(a.k) Define new states $s_{k,i+1} = s_{k-1,i} + \rho Q_k$, for all $i \in [1;k]$, and $s_{k,1} = 0$.
(b.k) Set costs $c(s_{k,i+1}) = c(s_{k-1,i}) + \beta^k V s_{k,i+1}$ and predecessors $\mathrm{pred}(s_{k,i+1}) = s_{k-1,i}, \forall i \in [1;k]$. Set label $u(s_{k-1,i}, s_{k,i+1}) = 0$.
(c.k) Compute $\bar{s}_{k-1} = \arg\min_{i\in[1,k]} c(s_{k-1,i})$. Set $c(s_{k,1}) = c(\bar{s}_{k-1}) + \beta^k U(t_n)$ and $\mathrm{pred}(s_{k,1}) = \bar{s}_{k-1}$. Set label $u(\bar{s}_{k-1}, s_{k,1}) = 1$.
(d) Finally, compute $\bar{s}_T = \arg\min_{i\in[1,T+1]} c(s_{T,i})$. Read the optimal control sequence backward on the labels appended to the optimal state sequence, i.e., $\widetilde{u}'_{T-k} = u(\mathrm{pred}^{(k)}(\bar{s}_T), \mathrm{pred}^{(k-1)}(\bar{s}_T))$ for all $k \in [0; T-1]$.
```

Output. The ϵ-optimal control sequence $\mathcal{U}^\epsilon$ is obtained for time slots $t_n+1, t_n+2, \ldots$ as $u(t_n+k) = \widetilde{u}'_k$, $1 \leq k \leq T$, and $u(t_n+k) = 0, \forall k > T$.

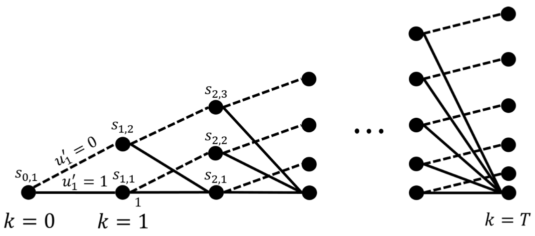

Fig. 4. State evolution of ϵ–$\mathsf{OPT}(U(t_n))$ routine.

2) Dynamic Programming (DP): We now turn to the actual computation of the ϵ-optimal strategy $\widetilde{u}'$ defined in Eq. (18). To this aim, we use a forward induction Dynamic Programming (DP) algorithm that we dub ϵ–$\mathsf{OPT}(U(t_n))$. Typically, the amount of memory required by DP grows exponentially over the iteration steps, which then makes the DP implementation a daunting task. This phenomenon is commonly referred to as the "curse of dimensionality" [23]. Fortunately, our case is tractable, as the state space grows linearly over time, as we illustrate in Fig. 4 and explain next.

The ϵ–$\mathsf{OPT}(U(t_n))$ routine iterates over all possible values that the surcharge cost S_k may take on at each step k, that we denote by the set of DP *states* $\{s_{k,i}\}_i$. At step $k = 0$, the ϵ–OPT routine is initialized. The set of states is a singleton, i.e., $\{s_{0,i}\}_i = s_{0,1} = 0$ with relative accumulated cost $c(s_{0,1}) = 0$.

At step $k = 1, \dots, T$, if no reconfiguration is applied and $u'_k = 0$ (see items *(a.k)*,*(b.k)*), then state $s_{k-1,i}$ evolves into $s_{k,i+1} = s_{k-1,i} + \rho Q_k$ and a cost $\beta^k V s_{k,i+1}$ is incurred. We then store in $c(s_{k,i+1}) = c(s_{k-1,i}) + \beta^k V s_{k,i+1}$ the cost accumulated by state $s_{k,i+1}$. Then, state $s_{k,i+1}$ has only one predecessor, namely $s_{k-1,i}$.

Otherwise, if $u'_k = 1$ (see item *(c.k)*) then all states $\{s_{k-1,i}\}_i$ converge to $s_{k,1} = 0$ and the incurred incremental cost equals $\beta^k U(t_n)$. In fact, the surcharge cost becomes null after a reconfiguration is applied. Consequently, state $s_{k,1} = 0$ has multiple candidate predecessors, $\{s_{k-1,i}\}_i$. By Bellman optimality principle, it suffices to select the one with minimum accumulated cost, hence the predecessor of state $s_{k,1} = 0$ is computed as $\arg\min_i c(s_{k-1,i})$.

Finally, for $k = T$ (see item *(d)*) the state with final state $\bar{s}_T$ with minimum cost is selected. The ϵ-optimal control $\widetilde{u}'$ can be then read backward, by recursively applying the predecessor operator pred from the final state $\bar{s}_T$ back to the initial state $s_{0,1}$.

It is easy to see that there are $k{+}1$ states at step k. Therefore, the number of operations carried out at time k (i.e., the state creation at items *(a.k)*,*(b.k)* and the state selection at item *(c.k)*) is linear in k. This allows us to claim that the ϵ–OPT routine solves the optimization problem (17) within a factor ϵ with complexity $\Theta(T^2)$. Finally, this suggests that the complexity of ϵ–OPT is polynomial in the length of the input (as T is a logarithmic function of $V, Q(t_n), U(t_n)$ and $1 - \beta$) and even sub-polynomial with respect to the approximation factor ϵ (as $T = \Theta(\log^2 \epsilon^{-1})$). This proves the following result.

Theorem 8: The ϵ–$\mathsf{OPT}(U(t_n))$ routine is a polynomial approximation scheme for the original optimization problem (17) within a renewal period.

D. Properties of RP

Using Constraint 1, we converted the original problem of a constrained MDP in a very large state space to a stochastic control problem which is amenable to Lyapunov optimization solutions. In particular, it is possible to solve the constrained problem optimally by a two level approach: (i) A virtual queue is used to capture the evolution of price of reconfiguration (and monitor the long-term feasibility of the frequency constraint) and (ii) inside the renewal frame we use deterministic optimal control taking into account the price. We have shown that the latter is a feasible methodology since it can be solved in polynomial time with respect to T, which is lower than the complexity of a single iteration executed by the solver. Another advantage of the proposed methodology is that only the probability that a flow arrives/departs is needed and not the whole distribution of $A(t)$. Apart from $\mathbb{P}(A(t) > 0)$ which needs to be estimated, the RP policy is purely adaptive and oblivious to past events or other system statistics. Furthermore, the exponential decrease of the optimality gap is necessary to prove the near-optimality of RP, but it does not affect the constraint on the reconfiguration frequency. From a practical perspective, this means that the reconfiguration frequency is always satisfied even if the routing solver exhibits a different convergence rate. Finally, we observe that the constraint is not very harmful since the surcharge $S(t)$ increases significantly when the traffic in the network change due to demand arrivals and departures. Therefore, it is desirable to reconfigure the network according to the solver solution at these time instances.

V. Greedy Policy (GP)

It is tempting to propose a heuristic approach which uses the drift-plus-penalty method at every time slot instead of every renewal frame. The virtual queue that corresponds to the constraint (8) on the reconfiguration budget is then updated at every time slot as

$$U(t+1) = [U(t) - h_{max}]^+ + u(t) \tag{20}$$

and we seek a policy that observes $U(t), \mathcal{S}(t)$ and tries to minimize the right-hand-side of the drift-plus-penalty expression $\mathbb{E}\left\{\frac{U^2(t+1)-U^2(t)}{2}\right\} + V\mathbb{E}\{S(t)\}$. Expectations are conditional on $U(t), S(t)$ and possible randomizations of the policy. Since $u(t) \in \{0,1\}$ and $([x]^+)^2 \le x^2$, we have $U^2(t+1) = u^2(t) + \left([U(t) - h_{max}]^+\right)^2 + 2[U(t) - h_{max}] \le 1 + h_{max}^2 + 2U(t)(u(t) - h_{max})$. Using (6), the drift plus penalty quantity can then be bounded as:

$$\begin{aligned}
&\mathbb{E}\left\{\frac{U^2(t+1) - U^2(t)}{2}\right\} + V\mathbb{E}\left\{S(t)\right\} \\
&\quad \le \frac{h_{max}^2}{2} + 1 + U(t)(\mathbb{E}\{u(t)\} - h_{max}) \\
&\qquad + V(S(t-1) + \rho Q(t-1))\mathbb{E}\left\{(1 - u(t))\right\} \\
&\quad \le \frac{h_{max}^2}{2} + 1 + U(t)h_{max} + V(S(t-1) + \rho Q(t-1)) \\
&\qquad + (U(t) - V(S(t-1) + \rho Q(t-1)))\mathbb{E}\{u(t)\} \qquad (21)
\end{aligned}$$

Minimizing the above bound on the drift-plus-penalty expression is achieved by the following threshold policy:

Greedy Policy (GP)

$$u(t) = \begin{cases} 1 & \text{if } U(t) < V\left(S(t-1) + \rho Q(t-1)\right) \\ 0 & \text{otherwise.} \end{cases}$$

Although GP minimizes the right hand side of (21), it is not a provably near-optimal policy for problem $\mathcal{P}$ within Π.

The reason is the following: drift-plus-penalty algorithms can be proven to be near-optimal for cases where *the cost that is added at every time slot depends only on the control taken at that slot and state variables that evolve independently of the control actions* [20]. This is not the case in our problem, since the evolution of the cost to be added at time slot t depends on the control policy taken in the previous slot, as seen by (6).

On the other hand, GP has numerous advantages. (i) Using the drift-plus-penalty we may show that it stabilizes $U(t)$ and hence it is a feasible policy (i.e., the reconfiguration constraint is satisfied). (ii) Contrary to RP, GP can be applied for constraints smaller than the probability of arrival of a flow and (iii) it does not require any information on the statistics of how the demands change-e.g. it does not need $\mathbb{P}(A(t) > 0)$. (iv) Similar to RP, it is oblivious to $Q(t)$ and can be applied without knowledge of the optimal value of the optimization. To implement GP we only need to keep track of the virtual queue $U(t)$, and the costs $Q(t), S(t)$ from the previous time slot. The constant V is chosen high enough to approximate well the optimal solution; a typical value is $V = 1000$.

By weak duality $C^{DUAL}(t) \leq C^{OPT}(t)$. Therefore, we can use the value of the objective function of the dual problem C^{DUAL} to bound the optimality gap $Q(t)$, as we did in the simulations presented in Section VI. We observe that $C^{DUAL}(t)$ converges to $C^{OPT}(t)$, hence $\widehat{Q}(t) = C(t) - C^{DUAL}(t)$ converges to $Q(t) = C(t) - C^{OPT}(t)$ as iterations go by. Since dual variables are computed by the Simplex algorithm as a byproduct, the computational complexity of the iterative solver does not change.

VI. Numerical Results

To evaluate the performance of our control policies on a realistic online SDN routing optimization system, we have implemented a scalable algorithm based on column generation to solve iteratively the linear program (Eq. (1)-(3)). In what follows, we first describe the experimental methodology and then illustrate the results of our experiments in two network settings that mimic realistic scenarios and are widely used by the research community.

A. Experimental Methodology

Since we are interested in solving large MCF problems, our implemented SDN solver works as follows. It first computes a feasible solution using FCS, allocating the demands that arrive to the system on cheapest paths over the residual graph. After this initialization phase, the solver proceeds by considering only a subset of paths and iterates by adding and removing paths (i.e., x_p variables) to a restricted version of the problem until it converges to the optimal point. At each iteration, the solver typically uses the dual costs of Eq. (1)-(3) to add only those paths that can improve the objective function.

In order to evaluate our policies we consider the real-life dataset captured in 2006 by Uhlig *et al.* [15] on *GEANT*, the high bandwidth pan-European research and education backbone and *a fat-tree topology*, a typical interconnection used in data-centers. GEANT contains a topology of 22 nodes and 36 high capacity 40G bidirectional links. The link cost has been rescaled in the range $[1; 100]$. The Fat-Tree topology consists of 4 pods. Link bandwidth and cost increase inversely with the tree depth in order to mimic the higher performance of network devices closer to the core. Table I shows the link parameters.

TABLE I
Link Bandwidth and Cost of Fat-Tree Network

Link	Bandwidth	Cost
Server-Edge	1 G	10
Edge-Aggregation	10 G	20
Aggregation-Core	20 G	40

In all our numerical experiments, we perform 50 independent measurements by generating as many traffic patterns and then present the averaged results over the measurements. The number of 50 different trials is large enough to yield very narrow confidence intervals.

B. Performance Evaluation of Rate-Limited Policies

In this set of experiments, we compare our control schemes RP and GP against a *Periodic Policy* (PP) that consists in reconfiguring the network periodically with a period equal to $\frac{1}{h_{max}}$, where h_{max} denotes the bound on the reconfiguration rate. To this end, we generate random traffic demands according to a Poisson process with inter-arrival time of 2 s and fixed duration of 20 s. The total simulation time is fixed to 10 min. In the following, we first present the results obtained over GEANT. Then, we show the performance obtained over the fat-tree topology.

1) GEANT Network: Fig. 5(a) illustrates the total surcharge (i.e., $\sum_{t=0}^{T} S(t)$) while 5(b) shows the reconfiguration rate $h = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{u(t)\}$ used by the control policies as a function of the bound on the reconfiguration rate h_{max}. Note that for RP we can only show the results for $h_{max} \geq 0.7$, which is slightly larger than the arrival rate λ in our scenario. Indeed, by definition RP needs to reconfigure at least as many times as the number of arrivals/departures.

Fig. 5(a) shows that RP and GP achieve performance almost identical to continuously reconfiguring the network ($S(t) = 0$) even with a reconfiguration rate lower than 100%. Furthermore, as illustrated in in Fig. 5(b), we can observe that both RP and GP satisfy the constraint on the reconfiguration rate, namely $h < h_{max}$ and achieve the same cost. Due to the imposed constraint "reconfigure after demand change" the RP requires a reconfiguration rate slightly larger than the arrival rate to obtain such a result, while GP seems to work well even with smaller arrival rates. Differently from RP which respects the constraint but optimally selects reconfigurations subject to it, GP selects the iterations with the largest improvement in terms of cost reduction. In our simulations, we observe that selecting the very first iterations of the column generation algorithm after an arrival/departure provides a very good performance.

From the figures, it can be further observed that the PP policy performs poorly when the SDN operator has a low budget on the reconfiguration rate. Specifically, when the controller operates with a limit on the reconfiguration rate in the $[0.1; 0.4]$ interval, using a periodic policy incurs in a surcharge 3 times larger than using the GP scheme. As the reconfiguration rate approaches 1, then all policies are allowed to reconfigure continuously, leading to the same cost.

Fig. 5(c) shows a snapshot of an evolution over time of the surcharge $S(t)$ for GP (solid line) and PP (dashed line) among the 50 trials with $h_{max} = 0.1$. While PP blindly

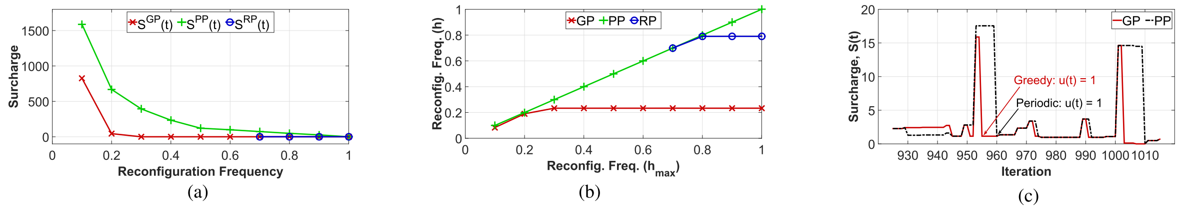

Fig. 5. Performance evaluation of the proposed control policies over GEANT as a function of the bound h_{max} on the reconfiguration rate. (a) Total surcharge. (b) Reconfiguration rate. (c) $S^{PP}(t)$ vs $S^{GP}(t)$ with $h_{max} = 0.1$.

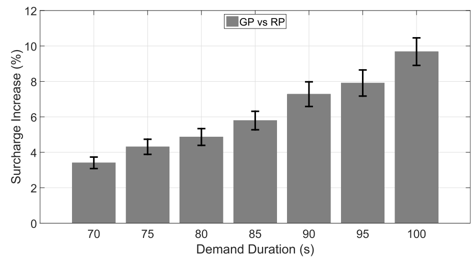

Fig. 6. Comparative evaluation of RP and GP over GEANT with low arrival rate and long demand duration. Each bar represents the increase of the surcharge of GP with respect to RP.

selects which iterations to use, leading to bad performance, GP "waits" until a big increase in the surcharge before applying the configuration of the solver, thus reducing the surcharge cost paid by the network operator.

As illustrated in the previous set of experiments, GP performs very closely to RP when demand arrivals and departures rarely move the optimal operational point. In this case, the routing solver can compute the optimal solution in few iterations, with the very first iteration providing a huge decrease in the optimality gap followed by small improvements. To illustrate the gain of RP over GP, we reduce the demand arrival rate and increase the demand duration. This permits to increase the time during which multiple demands coexist, thus increasing the number of iterations executed by the solver to converge to the optimal routing configuration. More specifically, demands are generated according to a Poisson process with inter-arrival time of 10 s. The demand duration ranges between 70 s and 100 s, while the total simulation time is fixed to 100 min in order to offset the lower demand arrival rate. Fig. 6 shows the percentage of extra surcharge paid using GP with respect to RP as a function of the demand duration. While GP is not optimal it performs very close to the RP policy (the surcharge is only 3%-11% higher than the optimal), thus representing a simple yet effective policy to decide which reconfigurations to apply to the network.

2) Fat-Tree Network: We now illustrate the results obtained in the Fat-Tree topology composed of 4 pods. Source and destination nodes of the arriving demands are selected among the leaves of the Fat-Tree.

As for GEANT, we measured the total surcharge (Fig. 7(a)) and the reconfiguration rate really used by our control policies (Fig. 7(b)) as a function of the bound on the reconfiguration rate h_{max}. As in the previous scenario, for all policies the total surcharge fades away as the reconfiguration rate increases. However, RP and GP perform consistently better than PP, since they select the best reconfiguration timeslots instead of deciding blindly when to reconfigure the network. We can further observe that GP achieves the same total surcharge as RP with a smaller reconfiguration rate. This is mainly due to negligible impact that demand departures cause to

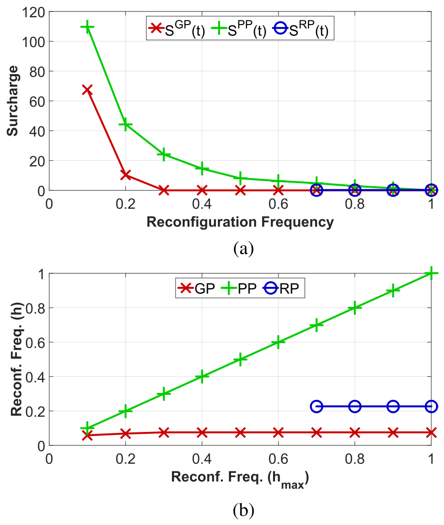

Fig. 7. Performance evaluation of the proposed control policies over a Fat-Tree network as a function of the bound h_{max} on the reconfiguration rate. (a) Total surcharge. (b) Reconfiguration rate.

the optimality gap. These events are usually ignored by GP, whereas by construction RP always reconfigures the network when a demand leaves the system.

C. Performance Evaluation of Fixed-Threshold Policy

In this set of experiments, we compare our GP against the Fixed-Threshold Policy (FTP) proposed in [24]. Fig. 8 illustrates the total surcharge obtained in this comparative evaluation. In our setting, FTP reconfigures the network whenever the decrease rate of the optimality gap is larger than a fixed threshold, namely $\frac{Q(t)}{Q(t-1)} > \alpha$, since we are interested in the minimization of the surcharge. As with other policies, a new arrival is firstly handled by the FCS, which quickly selects a path, and then FTP decides whether to reconfigure the network. Therefore, we always have $Q(t) \leq Q(t-1)$. Since FTP does not limit the reconfiguration rate, we executed this policy with several thresholds, which are illustrated on the x-axis at the bottom of Fig. 8. At the end of each simulation, we computed the reconfiguration rate that in turn is used as an input for the h_{max} parameter of our GP. The reconfiguration rates, which corresponds to the thresholds that have been used as input for FTP, are illustrated on the x-axis at the top of Fig. 8.

We can observe that GP always outperforms FTP, since FTP does not update dynamically the threshold that triggers the reconfiguration like our GP, thus skipping important network reconfigurations that can contribute to reduce the surcharge. Furthermore, FTP cannot be easily tuned by a network operator. As it can be observed in the figure, slightly increasing the threshold on the relative optimality gap to obtain

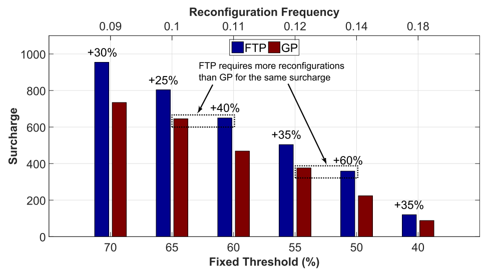

Fig. 8. Performance evaluation of the proposed Greedy Policy (GP) against the Fixed-Threshold Policy (FTP) over GEANT. On the x-axis we show the threshold on the relative optimality gap used by FTP (bottom) and the corresponding reconfiguration rate computed at the end of each simulation run (top).

the same surcharge of GP causes the violation of the constraint on the reconfiguration rate.

VII. Related Work

SDN enables a global and online routing optimization to improve link utilization and increase reactivity to failures. Google has showed in 2013 that they could achieve nearly 100% of link utilization [7] with their OpenFlow WAN controller. Two main factors are behind this architectural evolution: programmable data planes and logically centralized controller platforms. Several propositions have emerged in recent years to make data plane elements programmable and offload the control logic to external units. To name a few: Forces [25], PCEP [26] and OpenFlow [27].

Recent research on routing problems has been carried-out along many lines, which can be broadly divided into solving large problem instances, solving online versions of the problem, and computing consistent flow migration. Solving large routing problems has received a lot of interest from the traffic engineering community [28]. Several approaches have been proposed using column generation to solve large problem instances [3]. However, they have been mostly developed for offline network planning tools considering the worst case scenario and do not consider their use in a dynamic routing system.

The online multi-commodity flow version of this problem, where the parameters are revealed over time, has been studied for throughput maximization or load minimization, as detailed by Even and Medina [6]. In more general settings, the problem has been formulated as an online packing and covering problem [29], where the objective function as well as the packing constraints are not known in advance. While these works show sublinear competitiveness ratios, they have been mainly designed for admission control and do not propose any re-optimization of the flow allocation.

The network updates problem, which consists in computing the sequence of intermediary steps to move the network from an initial to a final flow configuration, has recently gained momentum [30]–[33]. In [30] Brandt *et al.* show that finding a sequence of reconfiguration steps for the migration of unsplittable flows without violating capacity constraints is NP-Hard. Similar results have been demonstrated in [31], where Xin *et al.* study the problem of jointly minimizing the reconfiguration steps and the network resource overhead during the migration process. Wang *et al.* [32] analyze the problem of deciding the order of flow table updates in order to avoid routing loops or deadlocks during the switch from an old to a new path. A complete survey on the network updates problem can be found in [33]. We would like to observe that the network updates problem is orthogonal and complementary to our problem. Indeed, we can envisage a SDN controller where our control policy selects the best routing configurations according to a limited budget for the reconfiguration rate and an algorithm for network updates schedules the sequence of steps necessary to set up the selected configuration.

VIII. Conclusion and Perspectives

Software-Defined Networking enables efficient utilization of network resources by dynamically adapting the routing configuration over time. In this context, this paper addresses an important question about the interplay between the high degree of configuration flexibility and the computational limits of the SDN controller logic.

Specifically, we examine the problem where the optimality gap of iterative routing algorithms decreases exponentially fast and we want to minimize the average routing cost subject to a constraint for the average reconfiguration frequency. Furthermore, we present two control policies working on top of the online routing optimization engine to decide whether to apply or not the current yet not optimal global network configuration. Numerical results on the GEANT network and fat-tree network topologies show that our control schemes can effectively track the evolution of the system using a bounded number of reconfigurations, thus pursuing the double objective of optimizing the performance and the system stability.

Appendix A
Proof of Lemma 3

The state space is countable due to $A(t)$ taking integer values and the form of eq. (5). Indeed, at time t, $Q(t)$ can be characterized completely from the vector $[A(\tau)]_{\tau=0,1,..,t}$, which is a $t-$dimensional vector with elements from a countable space (non-negative integers).

For proving irreducibility of $Q(t)$, note that under Assumption 1, state $Q(t) = 0$ is reachable from any state with positive probability (e.g. if there are no arrivals for a very long times), and every other state is reachable from state 0 with an appropriate sample path of $A(t)$.

Aperiodicity follows from irreducibility and the fact that state 0 is aperiodic (since $A(t)$ is i.i.d. over time, the return time to state 0 can be one with positive probability).

Appendix B
Proof of Lemma 5

Defining $L(x) = x^2$ and taking its drift for $Q(t)$, we have

$$\begin{aligned}\Delta L(Q(t)) &= \mathbb{E}\left\{L(Q(t+1)) - L(Q(t))\middle|Q(t)\right\}\\ &= e^2\mathbb{E}\{A^2(t)\} + 2(1-\rho)e\lambda Q(t) - \rho^2 Q^2(t).\end{aligned}$$

From the above, we can show that for all $\kappa_1, \kappa_2 > 0$ such that $\kappa_2 < \kappa_1 < \rho^2$ and $q_0(\kappa_1, \kappa_2) = \max\left[\frac{2(1-\rho)e\lambda}{\rho^2-\kappa_1}, \sqrt{\frac{e^2\sigma_A^2}{\kappa_1-\kappa_2}}\right]$ it holds

$$\Delta L(Q(t)) \leq -\kappa_2 Q^2(t) = -\kappa_2 L(Q(t)), \quad \forall Q(t) > q_0(\kappa_1, \kappa_2).$$

The above implies that $Q(t)$ is geometrically ergodic, i.e. there exist positive constants $M_0 < \infty, r \in (0,1)$ such that

$$|\mu_t - \mu|_{TV} \leq M_0 r^{-t},$$

where μ is the invariant distribution of $Q(t)$ and μ_t is its probability distribution at time slot t. For state q we then have

$$|\mu_t(q) - \mu(q)| \leq 2|\mu_t - \mu|_{TV} \leq 2M_0 r^{-t}. \tag{22}$$

For the second moment of return time, define $\mu_t(q,a) = \mathbb{P}\left(Q(t) = q, A(t-2) = a\right)$, $\mu_t(q|a) = \mathbb{P}\left(Q(t) = q|A(t-2) = a\right)$ and $\mu_t(q|a_2,a_1) = \mathbb{P}\left(Q(t) = q|A(t-2) = a_2, A(t-1) = a_1\right)$. We then have

$$\begin{aligned}\mu_t(q,a_2) &= \mu_t(q|a_2)\mu_t(a_2) = \mu_t(q|a_2,a_1)\mu_t(a_2)\mu_t(a_1)\\ &= \mu_{t-2}(q'(a_1,a_2))\mu_t(a_2)\mu_t(a_1),\end{aligned}$$

where we have defined $q'(a_1,a_2)$ is the state such that $q = (1-\rho)^2 q'(a_1,a_2) + (1-\rho)ea_1 + ea_2$. Eq. (22) then implies

$$\begin{aligned}\mu_t(q) &= \sum_{a_2=1}^{\infty}\sum_{a_1=0}^{\infty}\mu(a_2)\mu(a_1)\mu_{t-2}(q'(a_2,a_1))\\ &\leq \sum_{a_2=1}^{\infty}\sum_{a_1=0}^{\infty}\mu(a_2)\mu(a_1)\left(2M_0 r^{t-2} + \mu(q'(a_2,a_1))\right)\end{aligned} \tag{23}$$

and

$$\mu_t(q) \geq \sum_{a_2=1}^{\infty}\sum_{a_1=0}^{\infty}\mu(a_2)\mu(a_1)\left(-2M_0 r^{t-2} + \mu(q'(a_2,a_1))\right)$$

Taking in addition into account that $\sum_{a_2=1}^{\infty}\sum_{a_1=0}^{\infty}\mu(a_2)\mu(a_1) = 1-\beta$, we have

$$\begin{aligned}&\mathbb{E}\left\{N_q^2\right\}\\ &= \sum_{t=1}^{\infty} t^2\mu_t(q)\left(1-\mu_t(q)\right)^{t-1}\\ &\leq 2M_0(1-\beta)\sum_{t=1}^{\infty} t^2 r^{-t+2}\left(1-\mu_t(q)\right)^{t-1}\\ &\quad + \sum_{t=1}^{\infty} t^2 \sum_{a_2=1}^{\infty}\sum_{a_1=0}^{\infty}\mu(a_2)\mu(a_1)\mu(q'(a_2,a_1))(1-\mu_t(q))^{t-1}\\ &:= S_1 + S_2.\end{aligned}$$

The first series S_1 converges to a finite number (can be seenby e.g. a ratio test), therefore we need to bound S_2. Denoting $\widetilde{r} = \sum_{a_2=1}^{\infty}\sum_{a_1=0}^{\infty}\mu(a_2)\mu(a_1)\mu(q'(a_2,a_1))$, it holds

$$\begin{aligned}&S_2\\ &\leq \sum_{t,a_2,a_1} t^2\mu(a_2)\mu(a_1)\mu(q'(a_2,a_1))\big(1 + 2(1-\beta)M_0 r^{-t+2}\\ &\quad - \sum_{a_2,a_1}\mu(a_2)\mu(a_1)\mu(q'(a_2,a_1))\big)^{t-1}\\ &\leq \sum_{t=1}^{\infty} t^2\widetilde{r}\left(1-\widetilde{r}\right)^{t-1} + \widetilde{r}\sum_{t=1}^{\infty} t^2\left(2(1-\beta)M_0 r^{2-t}\right)^{t-1}\end{aligned}$$

Both series above are convergent: For the first one we can show it by a ratio test, and for the second by noting that there exists a $t_0 < \infty$ such that $2(1-\beta)M_0 r^{2-t} < 1, \forall t > t_0$. Therefore S_2 is convergent, which implies that at the end $\mathbb{E}\left\{N_q\right\} < \infty$.

Appendix C
Proof of Theorem 6

The setting we have here is slightly different with respect to optimization over renewal processes as presented in [20] and [21], as $Q(t_{n+1})$ depends on $Q(t_n)$ (in the references the uncontrolled process is i.i.d. at renewal periods). In order to resolve this, we use the idea of $T-$slot drifts as employed in [20] and [22] for Markovian dynamics of the uncontrolled state processes, extending it over renewal frames this time. Specifically, define $I(t) = \mathbb{1}_{\{A(t-2)>0\}}$ and the Markov chain $\mathbf{Z}(n) = [Q(t_n), I(t_n)]$. Then choose any of the states $\mathbf{z} = [q,1]$. Note here that since we also need an arrival to happen at states $\mathbf{z}$, we are essentially looking at the beginning of renewal frames of the initial problem. Starting from a point in time where $\mathbf{Z}(n) = [q,1]$ and denoting N_q the return time to that state, we write the variable drift for these N_q frames (all expectations are conditioned on $\mathbf{Z}(n) = [q,1]$):

$$\begin{aligned}&DPP(q,U(t_n))\\ &= \mathbb{E}\left\{L(U(t_{n+N_q}))\right\} - \mathbb{E}\left\{L(U(t_n))\right\}\\ &\quad + V\mathbb{E}\left\{\sum_{m=n}^{n+N_q-1}\sum_{t=t_m+1}^{t_m+T_m} S(t)\right\}\\ &\leq (\epsilon + B_q) + U(t_n)\mathbb{E}\left\{\sum_{m=n}^{n+N_q-1}\sum_{t=t_m+1}^{t_m+T_m}(u^*(t_n) - h_{max})\right\}\\ &\quad + V\mathbb{E}\left\{\sum_{m=n}^{n+N_q-1}\sum_{t=t_m+1}^{t_m+T_m} S^*(t)\right\},\end{aligned} \tag{24}$$

where

$$\begin{aligned}B_q = &\frac{\mathbb{E}\left\{T_n^2\right\}(1-h_{max})}{2}\mathbb{E}\left\{N_q\right\}\\ &+ \frac{\bar{T}(1+h_{max})}{2}\mathbb{E}\left\{N_q^2 - N_q\right\}.\end{aligned}$$

Since no arrivals occur within a renewal frame, the running cost at the beginning of a renewal frame is always zero and the solver is always applied the expected budget spent and cost accumulated within a renewal frame depend only on $Q(t_n)$. In addition, the state $[q,1]$ is also a renewal state for the process $\mathbf{Z}(n)$. Defining

$$\bar{u}^{\pi^*} = \limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\{u^*(t)\right\}$$

the optimal average budget used by π^* we have (using the renewal theorem)

$$\frac{1}{\mathbb{E}\left\{N_q\right\}\bar{T}}\mathbb{E}\left\{\sum_{m=n}^{m+N_q-1}\sum_{t=t_m+1}^{t_m+T_m} u^*(t)\middle| Q(t_n) = q\right\} = \bar{u}^*,$$

which implies

$$\begin{aligned}&\mathbb{E}\left\{\sum_{m=n}^{m+N_q-1}\sum_{t=t_m+1}^{t_m+T_m}(u^*(t) - h_{max})\middle| Q(t_n) = q\right\}\\ &\qquad \leq \mathbb{E}\left\{N_q\right\}\bar{T}\bar{u}^* - \mathbb{E}\left\{N_q\right\}\bar{T}h_{max} \leq 0,\end{aligned} \tag{25}$$

since by construction of π^* it is $\bar{u}^{\pi^*} \leq h_{max}$. We further define $n'_q = \min\left\{n \geq 0 : \mathbf{Z}(n) = [q,1]\right\}$, $N_q(m) =$

$\min\{k \in \mathbb{Z}_+ : \mathbf{Z}(m+k) = \mathbf{z}, \mathbf{Z}(m) = \mathbf{z}\}$ with $N_q(0) = 0$ and $\sigma_{q,m} = \sum_{k=0}^{m} N_q(k)$. Using (25) and summing (24) over M periods where $\mathbf{Z}(n)$ returns to state $[q, 1]$, we have

$$\frac{\mathbb{E}\left\{L(U(t_{n'_q+\sigma_{q,M}}))\right\}}{MV\bar{T}} - \frac{\mathbb{E}\left\{L(U(t_{n'_q}))\right\}}{MV\bar{T}} + \frac{1}{M\bar{T}}\sum_{m=0}^{M-1}\mathbb{E}\left\{\sum_{n=n'_q+\sigma_{q,m}}^{n'_q+\sigma_{q,m+1}-1}\sum_{t=t_n+1}^{t_n+T_n} S(t)\right\}$$
$$\le \frac{\epsilon + B_q}{V} + \frac{1}{M\bar{T}}\sum_{m=0}^{M-1}\mathbb{E}\left\{\sum_{n=n'_q+\sigma_{q,m}}^{n'_q+\sigma_{q,m+1}-1}\sum_{t=t_n+1}^{t_n+T_n} S^*(t)\right\}.$$

In addition, starting from $t = t_0 = 0$, we get

$$\frac{\mathbb{E}\left\{L(U(t_{n'_q+\sum_{m=0}^{M} N_q(m)}))\right\}}{MV\bar{T}} - \frac{\mathbb{E}\{L(U(0))\}}{MV\bar{T}} + \frac{1}{M\bar{T}}\sum_{m=0}^{M-1}\mathbb{E}\left\{\sum_{n=n'_q+\sigma_{q,m}}^{n'_q+\sigma_{q,m+1}}\sum_{t=t+1}^{t_n+T_n} S(t)\right\} + \frac{1}{M\bar{T}}\mathbb{E}\left\{\sum_{n=0}^{n'_q-1}\sum_{t=t_n+1}^{t_n+T_n} S(t+1)\right\}$$
$$\le \frac{\epsilon + B_q}{V} + \frac{1}{M\bar{T}}\sum_{m=0}^{M-1}\mathbb{E}\left\{\sum_{n=n'_q+\sigma_{q,m}}^{n'_q+\sigma_{q,m+1}}\sum_{t=t_n+1}^{t_n+T_n} S^*(t)\right\} + \frac{1}{M\bar{T}}\mathbb{E}\left\{\sum_{n=0}^{n'_q-1}\sum_{t=t_n+1}^{t_n+T_n} S^*(t)\right\} \quad (26)$$

We first prove part 2) of the Theorem. Using again the fact that $[q, 1]$ is a renewal state for process $Z(n)$, we have (for the rest of the proof, N_q denotes the generic random variable describing the return time)

$$\bar{S}^{\pi} = \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{S(t)\} = \limsup_{M\to\infty} \frac{1}{\mathbb{E}\{N_q\}\bar{T}M}\sum_{m=0}^{M-1}\mathbb{E}\left\{\sum_{n=n'_q+\sigma_{q,m}}^{n'_q+\sigma_{q,m+1}}\sum_{t=t_n+1}^{t_n+T_n} S(t)\right\} \quad (27)$$

and

$$\bar{S}^{\pi^*} = \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{S^*(t)\} = \limsup_{M\to\infty} \frac{1}{\mathbb{E}\{N_q\}\bar{T}M}\sum_{m=0}^{M-1}\mathbb{E}\left\{\sum_{n=n'_q+\sigma_{q,m}}^{n'_q+\sigma_{q,m+1}}\sum_{t=t_n+1}^{t_n+T_n} S^*(t)\right\}. \quad (28)$$

Note now that the first term of the left hand side of (26) is nonnegative. In addition the expectations of the costs accumulated for any policy before $Z(n)$ reaches state $[q, 1]$ for the first time are finite. Assuming, as is usual in this kind of problems [20], that $\mathbb{E}\{L(U(0))\}$ is also finite (e.g. starting with $U(0) = 0$), taking limits as $M \to \infty$ in (26) and using (27), (28) we get

$$\bar{S}^{\pi} \le \frac{\epsilon + B_q}{V} + \bar{S}^{\pi*}.$$

For every $q \in \mathcal{Q}$, $B_q < \infty$, as asserted by the Lemma 5. Since any state $[q, 1]$ can be used (it follows from the above and Lemma 5 that the second moment of return to any of these states is finite), the best provable tradeoff constant is $B = \inf_{q\in\mathcal{Q}}[B_q]$, which completes the proof of part 2).

Finally, we prove part 1), that is the constraint for the budget we have for reconfigurations is satisfied. It is sufficient to prove that queue $U(t_n)$ is mean rate stable under the proposed policy [20]. Indeed, using (24) and (25) we have

$$\mathbb{E}\left\{L(U(t_{n+N_q})|q)\right\} - \mathbb{E}\{L(U(t_n)|q)\} \le (\epsilon + B_q) + \bar{T}\mathbb{E}\{N_q\}\left(\bar{S}^* - \bar{S}\right) < \infty, \quad \forall q \in \mathcal{Q},\ \forall U(t_n).$$

Since $L(x)$ is a Lyapunov function, the above implies mean rate stability of the virtual queue.

Appendix D
Proof of Lemma 7

We first observe that the formulation in (18) is equivalent to the original problem (17) with $k := t - t_n$ and boundary conditions $Q_0 = Q(t_n)$ and $S_0 = 0$. Let us define the instantaneous cost at step k as $g_k(S_k, u'_k) := \beta^k(VS_k + U(t_n)u'_k)$. We then notice that g_k is bounded for all $k \ge 1$:

$$0 \le g_k(S_k, u'_k) \le \beta^k\Big(VQ(t_n) + U(t_n)\Big). \quad (29)$$

Then, for any $T \ge 1$ and control u',

$$\sum_{k=T+1}^{\infty} g_k(S_k, u'_k) \le \frac{\beta^{T+1}}{1-\beta}\big(VQ(t_n) + U(t_n)\big) := \epsilon. \quad (30)$$

Let us now define $\underline{u}'$ as the optimal control of the non-truncated version of the optimization problem in (18). We then denote $\overline{u}'$ such that $\overline{u}'_k = \widetilde{u}'_k$ for $1 \le k \le T$ and $\overline{u}'_k = \{0, 1\}$ for $k > T$. Then we call $\underline{S}_k$ and $\overline{S}_k$ the state evolution generated by the controls $\underline{u}'$ and $\overline{u}'$, respectively. We can now bound the optimality gap of the control $\overline{u}'$ as follows:

$$\begin{aligned} g(\overline{u}') - g(\underline{u}') &= \sum_{k=1}^{\infty} g_k(\overline{S}_k, \overline{u}'_k) - \sum_{k=1}^{\infty} g_k(\underline{S}_k, \underline{u}'_k) \\ &= \sum_{k=1}^{T} g_k(\overline{S}_k, \widetilde{u}'_k) - \sum_{k=1}^{T} g_k(\underline{S}_k, \underline{u}'_k) \\ &\quad + \sum_{k=T+1}^{\infty} g_k(\overline{S}_k, \overline{u}'_k) - \sum_{k=T+1}^{\infty} g_k(\underline{S}_k, \underline{u}'_k) \\ &\le \sum_{k=T+1}^{\infty} g_k(\overline{S}_k, \overline{u}'_k) \le \epsilon. \end{aligned}$$

In fact, $\sum_{k=1}^{T} g_k(\overline{S}_k, \widetilde{u}'_k) \le \sum_{k=1}^{T} g_k(\underline{S}_k, \underline{u}'_k)$ since $\widetilde{u}'$ is optimal over the finite time horizon $[1; T]$, and moreover $\sum_{k=T+1}^{\infty} g_k(\underline{S}_k, \underline{u}'_k) \ge 0$. By rewriting T as a function of ϵ in equation (30), the thesis is proven.

References

[1] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, 3rd Quart., 2014.

[2] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, “An overview of routing optimization for Internet traffic engineering,” *IEEE Commun. Surveys Tuts.*, vol. 10, no. 1, pp. 36–56, 1st Quart., 2008.

[3] K. Murakami and H. S. Kim, "Optimal capacity and flow assignment for self-healing ATM networks based on line and end-to-end restoration," *IEEE/ACM Trans. Netw.*, vol. 6, no. 2, pp. 207–221, Apr. 1998.
[4] X. Jin *et al.*, "Dynamic scheduling of network updates," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 539–550, 2014.
[5] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 15–26, 2013.
[6] G. Even and M. Medina, "Online multi-commodity flow with high demands," in *Approximation and Online Algorithms* (Lecture Notes in Computer Science), vol. 7846, T. Erlebach and G. Persiano, Eds. Berlin, Germany: Springer, 2013.
[7] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.
[8] N. Jose and A. K. Somani, "Connection rerouting/network reconfiguration," in *Proc. IEEE DRCN*, Oct. 2003, pp. 23–30.
[9] D. Coudert, F. Huc, D. Mazauric, N. Nisse, and J.-S. Sereni, "Reconfiguration of the routing in WDM networks with two classes of services," in *Proc. Int. Conf. Opt. Netw. Design Modeling (ONDM)*, Feb. 2009, pp. 1–6.
[10] A. E. Ozdaglar and D. P. Bertsekas, "Routing and wavelength assignment in optical networks," *IEEE/ACM Trans. Netw.*, vol. 11, no. 2, pp. 259–272, Apr. 2003.
[11] J. Desrosiers, F. Soumis, and M. Desrochers, "Routing with time windows by column generation," *Networks*, vol. 14, no. 4, pp. 545–565, 1984.
[12] N. L. Roux, M. Schmidt, and F. R. Bach, "A stochastic gradient method with an exponential convergence _rate for finite training sets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 2663–2671.
[13] Y. Nesterov, "Gradient methods for minimizing composite functions," *Math. Program.*, vol. 140, no. 1, pp. 125–161, Aug. 2013.
[14] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.
[15] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 83–86, 2006.
[16] S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*. London, U.K.: Springer-Verlag, 1993.
[17] E. Altman, *Constrained Markov Decision Processes*. Boca Raton, FL, USA: CRC Press, 1999.
[18] V. Heorhiadi, M. K. Reiter, and V. Sekar, "Simplifying software-defined network optimization using SOL," in *Proc. Usenix NSDI*, 2016, pp. 223–237.
[19] R. Hartert *et al.*, "A declarative and expressive approach to control forwarding paths in carrier-grade networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 15–28, 2015.
[20] M. J. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan & Claypool, 2010.
[21] M. J. Neely, "Dynamic optimization and learning for renewal systems," *IEEE Trans. Autom. Control*, vol. 58, no. 1, pp. 32–46, Jan. 2013.
[22] L. Huang and M. J. Neely. (2010). "Max-weight achieves the exact $[O(1/V), O(V)]$ utility-delay tradeoff under Markov dynamics." [Online]. Available: https://arxiv.org/abs/1008.0200
[23] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA, USA: Athena Scientific, 1995.
[24] R. S. Bhatia, M. S. Kodialam, and T. V. Lakshman, "Method for fast network re-optimization," U.S. Patent 7 433 315, Oct. 7, 2008.
[25] A. Doria *et al.*, *Forwarding and Control Element Separation (ForCES) Protocol Specification*, document RFC 5810, IETF, Fremont, CA, USA, Mar. 2010.
[26] J. Vasseur and J. L. Roux, *Path Computation Element (PCE) Communication Protocol (PCEP)*, document RFC 5440, IETF, Fremont, CA, USA, Mar. 2009.
[27] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.
[28] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2211–2219.
[29] N. Buchbinder and J. Naor, "The design of competitive online algorithms via a primal—Dual approach," *Found. Trends Theor. Comput. Sci.*, vol. 3, nos. 2–3, pp. 93–263, 2009.
[30] S. Brandt, K.-T. Förster, and R. Wattenhofer, "On consistent migration of flows in SDNs," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.
[31] Y. Xin, M. Shayman, R. J. La, and S. I. Marcus, "Reconfiguration of survivable IP over WDM networks," *Opt. Switching Netw.*, vol. 21, pp. 93–100, Jul. 2016.
[32] W. Wang, W. He, J. Su, and Y. Chen, "Cupid: Congestion-free consistent data plane update in software defined networks," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.
[33] K.-T. Foerster, S. Schmid, and S. Vissicchio. (2016). "Survey of consistent software-defined network updates." [Online]. Available: https://arxiv.org/abs/1609.02305

Apostolos Destounis received the Diploma degree in electrical and computer engineering from the National Technical University of Athens, in 2009, the M.Sc. degree in communications and signal processing from Imperial College London, in 2010, and the Ph.D. degree in telecommunications from Supélec in2014. During the Ph.D., he was also with Alcatel-Lucent Bell Labs, France. Since 2014, he has been with the Mathematical and Algorithmic Sciences Laboratory, France Research Center, Huawei Technologies Company Ltd., where he is currently a Senior Research Engineer. His research interests include stochastic optimization and online learning with applications in wireless networks and SDN.

Stefano Paris received the M.S. degree in computer engineering from the University of Bergamo in 2007 and the Ph.D. degree in information engineering from Politecnico di Milano in 2011. He has been an Assistant Professor (currently on leave) with LIPADE, Computer Science Department, Paris Descartes University. He is currently a Senior Researcher with the Mathematical and Algorithmic Sciences Laboratory, France Research Center, Huawei Technologies Company Ltd. His main research interests include optimization and performance evaluation of software defined and wireless networks.

Lorenzo Maggi received the Ph.D. degree from INRIA and the University of Nice-Sophia Antipolis, in 2012, with a thesis on the interplay between game theory and Markov decision processes. From 2013 to 2015, he served as a Post-Doctoral Researcher with Create-Net, Trento, Italy. He is currently a Researcher with Huawei Technologies Company Ltd., Paris, France. His current research interests include the design of optimization and learning algorithms for resource allocation in communication networks. He was a recipient of the Best Paper Award at Wiopt 14 and ITC 17.

Georgios S. Paschos received the Diploma degree in electrical and computer engineering from the Aristotle University of Thessaloniki, Greece, in 2002, and the Ph.D. degree in wireless networks from the Electrical and Computer Engineering Department, University of Patras, Greece, in 2006. He held research positions at VTT, Finland, from 2007 to 2008, CERTH-ITI, Greece, from 2008 to 2012, and LIDS, MIT, USA, from 2012 to 2014. From 2009 to 2012, he was also with the Electrical and Computer Engineering Department, University of Thessaly. He has been a Principal Researcher with Huawei Technologies Company Ltd., Paris, France, leading the Network Control and Resource Allocation Team since 2014. Two of his papers received the best paper award, in GLOBECOM 07 and IFIP Wireless Days 09, respectively. He was the Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS special issue for content caching and delivery, and he actively serves as an Associate Editor for the IEEE/ACM TRANSACTIONS ON NETWORKING, and IEEE NETWORKING LETTERS, and as a Technical Program Committee member of INFOCOM, WiOPT, and Netsoft.

Jérémie Leguay received the Ph.D. degree from Pierre and Marie Curie University, where he worked on ad hoc and delay-tolerant networks. From 2007 to 2014, he was a Research Engineer and responsible for the Research and Technology Lab on Networked Systems, Thales Communications and Security, Genneviliers, France, where he developed activities on sensor networks, mobile networks, and software-defined networks. In 2014, he joined the Mathematical and Algorithmic Sciences Laboratory, Huawei Technologies Company Ltd., Boulogne-Billancourt, France, as a Principal Engineer and a leader of the Network and Traffic Optimization Team. His current research activities are in the control and management of IP and optical networks using optimization and machine learning tools.