Learning the Jointly Optimal Routing and Controller Placement Policy in Mobile Software-Defined Networks

Iordanis Koutsopoulos
Department of Informatics
Athens University of Economics and Business, Greece

***Abstract*—We study the joint problem of data traffic routing and controller placement that arises in tactical mobile Software Defined Networks after the establishment of forwarding rules at switches for a new, long-lasting flow. The aim is to minimize a cost metric that includes the routing cost, and the delay between the controller and the switches across the route. The latter depends on the relative locations of the controller and the switches, which in turn depends on both the routing and the controller placement policies. A small value of controller-to-switch delay is important for low-latency communication between the controller and the switches, which is essential for route adaptation. The static minimization problem is shown to be equivalent to a minimum-cost problem in an enhanced graph.**

Next, we study the online learning version of the problem, so as to learn a joint policy that achieves sub-linear regret in the presence of non-stationary dynamics of link costs and controller-to-switch delays. We present low-complexity algorithms based on the Follow-the-Perturbed-Leader (FTPL) one, which we fit to the SDN setting for full or limited feedback after each decision. The low complexity of these algorithms makes them suitable for distributed dynamic tactical scenarios.

I. Introduction

Software Defined Networking (SDN) has seen several success stories over the last decade in wire-line core networks and at the wireless last mile of infrastructure-based networks, e.g. Long Term Evolution (LTE) cellular, and WiFi [1]. Recently, several works advocate the use of SDN in mobile ad-hoc networks (MANETs) [2]-[4].

In SDN, the control and data-plane functionalities are separated. The former are placed at dedicated nodes, the controllers, while data-plane forwarding is performed by SDN-enabled switch nodes according to their flow tables. For each new flow, a path needs to be found from the source to the destination. The controller sends PACKET-OUT messages to all switches to discover the network topology and collect link state information. That is, it periodically queries switches for connectivity information and for statistics such as packet error rate (PER), packet delay for adjacent links, number of retransmissions, or link capacity usage in terms of time slots. It then uses (some of) these statistics as link weights and solves a centralized minimum-cost routing problem in the communication graph to find a route, namely a subset of SDN switches and links that form a path from the source to the destination. Next, the controller transmits the new forwarding rules to these switches via shortest paths through a communication protocol such as OpenFlow, and the rules are installed on the switches. In the setting above, it is important to find a minimum-cost path for data traffic routing.

However, in *tactical* MANETs, long-lasting flows, e.g. live video or voice streaming sessions are common. In the presence of dynamic network conditions during the session, it is important to continually compute a minimum-cost route, but also to continually select an appropriate node as controller. This will result in low-latency communication between the controller and each switch across the path, so as to fulfill the requirement for fast controller reaction and route recalculation in cases of route quality deterioration during the session. If the controller-to-switch delays are not considered, route adaptation will be slow. Thus, although traditionally routing upon a new flow establishment and the controller placement decision arise at different time scales, in the long-lasting flows scenario above, *routing and controller placement occur at the same time scale, and they are coupled.*

In this paper, we study the joint problem of routing and controller placement in tactical MANETs, with the aim to minimize a cost metric that comprises the routing cost, namely the sum of costs of links on the path, and the delay between the controller and the switches along the path. For given link cost and controller-to-switch (C-S) delays, we show that the joint problem becomes a minimum-cost routing problem in an appropriately defined enhanced graph.

Next, we address the problem of *online learning* of the jointly optimal routing and controller placement policy when link costs and C-S delays are unknown and time-varying. The first challenge is that these time variations may be hard to characterize statistically because system parameters vary rapidly and unpredictably with time, according to *non-stationary* random processes due to node mobility and other dynamics. Thus, wireless links have time-varying quality, while the C-S delay through the (possibly) multi-hop C-S control path may be unpredictable as well. Therefore, even if we knew the instantaneous parameter values at time t, these would change arbitrarily by the time of the next decision at $t+1$. Another challenge due to the nature of the SDN protocol is the limited received feedback after each decision. Namely, link costs can be measured *only* for those links on the selected

routing path and not for all network links, while C-S delays are measured *only* for the currently selected controller and not for other candidate locations. Finally, the algorithm needs to be lightweight and amenable to distributed implementation.

To address these challenges, we propose lightweight algorithms that further develop the Follow-the-Perturbed-Leader (FTPL) one. In our setting, the idea is at each time slot t: (*i*) for the current routing and controller placement, measure routing link cost and C-S delays, (*ii*) in an appropriately defined enhanced graph, update a cumulative link cost up to time $t-1$, which captures the routing link cost and C-S delays, (*iii*) perturb cumulative link cost with a random variable, and (*iv*) solve a minimum-cost routing problem on the enhanced graph with perturbed link costs to find the next route and controller placement. The contribution to the state of the art is as follows:

- We formulate the joint problem of traffic routing and controller placement, and we show that it becomes a minimum-cost routing problem in an appropriately defined, enhanced graph.
- We formulate the online-learning version of the problem.
- We adapt the FTPL learning algorithm to the SDN setting, for the cases of full or limited feedback, where the latter is dictated by the SDN protocol. The algorithms are computationally efficient and amenable to the distributed tactical MANET environment.
- We show through simulations that these algorithms have different regret performance for different scenarios of the link cost and controller-to-switch delay dynamics.

The paper is organized as follows. In section II, we review the state of the art. In section III we consider the static problem, and we present the problem statement and solution. In section IV, we solve the learning problem for full and limited feedback. Section V presents numerical results, and section VI concludes the paper.

II. Background and Related work

A. Online Learning

In online learning, the learner at time t chooses an action $\mathbf{x}_t$ in a continuous or discrete decision space $\mathcal{X}$ and observes the outcome $f_t(\mathbf{x}_t)$, where $f_t(\cdot) : \mathcal{X} \to \mathbb{R}$ is a loss function chosen arbitrarily by the environment. The learner needs to select a policy, namely a sequence of actions $\{\mathbf{x}_t\}_{t=1}^T$ that minimizes regret, defined as $\sum_{t=1}^t f_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{X}} \sum_{t=1}^T f_t(\mathbf{x})$, namely the difference between the cumulative loss of the policy and that of the policy that always selects the best action in hindsight. The aim is to achieve sub-linear regret in terms of the learning horizon T.

Convex decision set and cost functions. For continuous-valued convex $\mathcal{X}$ and convex loss functions, both deterministic and randomized learning algorithms exist. The deterministic Follow-the-Leader (FTL) algorithm at each time t selects the policy for which the cumulative loss up to $t-1$ is smallest, namely $\mathbf{x}_t = \arg\min_{\mathbf{x}\in\mathcal{X}} \sum_{s=0}^{t-1} f_s(\mathbf{x})$. However FTL suffers from a linear regret for certain scenarios with linear loss functions. Follow-the-Regularized-Leader (FTRL) [5] is of similar spirit but adds a convex regularization function to the cumulative cost. On the other hand, randomized algorithms such as Follow-the-Perturbed-Leader (FTPL) at each time t select the action that minimizes the perturbed cumulative loss up to t, where the added random perturbation is drawn from a probability distribution such as uniform or exponential. That is, $\mathbf{x}_t = \arg\min_{\mathbf{x}\in\mathcal{X}}(\sum_{s=0}^{t-1} f_s(\mathbf{x}) + \mathbf{n}^T\mathbf{x})$, where $\mathbf{n}$ is a random perturbation vector. Both FTRL and FTPL achieve $O(\sqrt{T})$ regret [5], [6, Sec.5.5]. However, FTPL is computationally more efficient than FTRL, since the latter needs to implement a computationally expensive projection step, while FTPL solves a convex optimization problem. The sublinear regret property of FTRL and FTPL is also achieved for linear loss functions [6, Sec.5.5].

d-expert selection. When $\mathcal{X} \subseteq \{0,1\}^d$ or the loss functions are non-convex, only randomized algorithms achieve sublinear regret [7]. A representative problem with non-convex decision space is d-expert selection, where each expert incurs a loss at each time t. There, $\mathcal{X}$ is the set of d unit vectors, with 1 in only one component, and 0s in others.

With *full* feedback, at each time t, we pick an expert a_t and observe the losses of *all* experts. The problem is to find an expert selection policy with sublinear regret with respect to the best expert in hindsight. The randomized Exponentially Weighted Average (EWA) algorithm selects at each time t an expert with probability that is inversely proportional to an exponential function of the cumulative loss of that expert up to $t-1$ and achieves $O(\sqrt{T\log d})$ regret. On the other hand, FTPL subtracts a random perturbation from the cumulative loss of each expert and chooses the one with minimum perturbed cumulative loss [8], and it achieves regret $O(\sqrt{dT})$.

In the *semi-bandit* feedback scenario, when the learner picks an expert a_t at time t, it gets to see *only* the loss $\ell^t_{a_t}$ associated with the selected expert and cannot observe the full vector of losses $\boldsymbol{\ell}^t$ of all experts. For adversarial losses, the EXP3 algorithm achieves $O(\sqrt{2Td\log d})$ regret [20]. EXP3 is based on EWA and selects at each time t an expert a_t with probability $p^t_{a_t}$ that is inversely proportional to an exponential function of the cumulative loss of the expert up to t. However, there are two differences compared to EWA. First, an unbiased estimate of the loss vector is formed, through vector $\tilde{\boldsymbol{\ell}}^t$ with components equal to $\tilde{\ell}^t_j = \ell^t_j/p^t_j$, if $j = a_t$, and 0 otherwise. This is an unbiased estimate of the incurred loss at time t, since for each j,

$$\mathbb{E}[\tilde{\ell}^t_j \,|\, \mathbf{p}^t] = p^t_j \cdot \frac{\ell^t_j}{p^t_j} + (1-p^t_j)\cdot 0 = \ell^t_j\,. \tag{1}$$

Second, the cumulative cost of *only* the selected expert is updated each time with the loss estimate above.

Online Combinatorial optimization. An important problem with a combinatorial decision set $\mathcal{X}$ that gives rise to the thread of *online combinatorial optimization* is the *online shortest path* one in a directed acyclic graph $G(V,E)$. Given a graph with arbitrarily time-varying link costs, the goal is to find a path selection policy with sub-linear regret with respect

to the best fixed route in hindsight. At each time t, the learner selects a path, namely a subset of links, denoted by vector $\mathbf{x}_t \in \{0,1\}^d$, where $d = |E|$, and vector entries are 1 or 0 respectively for links that are or are not in the path, subject to $\sum_{i=1}^{d} x_{t,i} \leq m$, namely at most m links are selected.

With *full* feedback, the link cost vector $\boldsymbol{\ell}_t \in [0,1]^d$ is revealed, and the learner suffers a total cost $\mathbf{x}_t^T \boldsymbol{\ell}_t$. In FTPL, the learner at each time t observes each link cost, it updates the cumulative link cost, and it perturbs it with a random quantity. Then, it computes the minimum-cost path with perturbed link costs [9]. On the other hand, in EWA, we cannot consider each path as an expert due to the large number of paths. Instead, the path can be formed by selecting links one by one based on a probabilistic rule, while ensuring that a chain of links exists after each link selection. However, the entire set of paths between two nodes is needed when doing random link selection. While a regret of $O(m^{3/2}\sqrt{T\log(d/m)})$ is achieved by EWA [10] and FTPL [11], FTPL is more efficient and straightforward to implement as it solves a linear optimization problem, and it avoids the d-dimensional categorical distribution and the enumeration of the set of paths.

With *semi-bandit* feedback, only the costs of the links on the current path are revealed to the learner. Bandit algorithms use some form of loss estimate to handle limited feedback. However, while in EXP3 the selection probabilities required for loss estimates are available, this is not the case for FTPL, hence some method is needed to estimate these probabilities, and one method is called Geometric Resampling (GR) [11].

The idea in GR is to estimate the probability of inclusion of a link in the minimum-cost path by measuring the re-occurrence time of the link on minimum-cost paths. This emerges from the fact that for a binary event with occurrence probability p, the expected number of trials between two successive occurrences of the event is $1/p$. Suppose that at time t for a link (i,j) in the min-cost path, we see cost $c_{i,j}^t$. Let $p_{i,j}^t$ be the unknown probability of selecting link (i,j) in a min-cost path. Suppose we perform multiple trials, and in each trial we perturb each link cost with a random variable and compute the min-cost path. If $N_{i,j}^t$ is the random number of trials until the link appears again in a min-cost path, we can use $\tilde{c}_{i,j}^t = c_{i,j}^t N_{i,j}^t$ as a link cost estimate at time t. This is an unbiased estimate of the link cost since,

$$\begin{aligned}\mathbb{E}[\tilde{c}_{i,j}^t] &= \sum_{e\in\mathcal{E}} p_e^t \mathbb{E}[\tilde{c}_{i,j}^t \mid I_t = e] = p_{i,j}^t \mathbb{E}[c_{i,j}^t N_{i,j}^t \mid I_t = (i,j)] \\ &= p_{i,j}^t c_{i,j}^t \mathbb{E}[N_{i,j}^t \mid I_t = (i,j)] = p_{i,j}^t c_{i,j}^t \tfrac{1}{p_{i,j}^t} = c_{i,j}^t,\end{aligned}$$

where I_t is the selected link at t. FTPL with GR achieves a regret of $O(m\sqrt{dT\log d})$ [11].

B. SDN routing and controller placement

SDN routing. SDN routing is equivalent to the multi-commodity flow problem. For given flow demands, the problem to find paths for different commodities (flows) so as to either minimize the maximum link utilization, essentially balancing the load on different links, or to minimize the total cost of transporting each flow from source to destination. A taxonomy of SDN routing algorithms is presented in [12]. The authors distinguish between algorithms that use a static link cost (e.g. hop count, distance, link capacity), a dynamic link cost (e.g. link utilization, available link capacity), or metrics that reduce inter-flow interference. The impact of imperfect network state information due to periodic querying of switches is discussed. In [13], the authors extend the multi-commodity flow problem for hybrid SDN, where some nodes are SDN switches, while others are ordinary nodes that run shortest-path routing protocols. Several Machine Learning (ML) approaches have been proposed in SDN routing, mainly based on supervised or reinforcement learning. We refer the reader to [14] for a taxonomy.

Controller placement. Many works study the static version of the controller placement problem. The work [15] finds the number of required controllers and their locations so as to: (*i*) minimize the average C-S latency over switches (assuming that each switch connects to its associated controller through a shortest path), which is the k-median problem; (*ii*) minimize the maximum C-S latency over all switches, which is the k-center problem; (*iii*) minimize the number of switches with C-S delays that are bigger than a value. The paper concludes that for medium-sized networks, one controller is enough to achieve good latency.In [16], the authors find the controller placement and switch association so as to minimize the maximum delay to the associated controller, subject to a maximum controller load. In [17], the authors formulate the problem of controller placement so as to minimize a weighted sum of delay and message overhead cost for C-S and controller communication. Models are inspired by measurements on the OpenDaylight and ONOS SDN platforms.

Learning approaches for controller placement are much fewer. The work [18] casts the problem of learning the controller placement as a multiarmed bandit, where each arm is a different option for controller placement, and the performance metric is C-S round-trip time (RTT). The cases of stationary and non-stationary RTT random processes are studied through stochastic and adversarial bandits respectively. In [19], Deep Reinforcement Learning is used for placing a certain number of controllers. The state of a switch is the set of flows addressed to each controller, while the action is the switch assignment to a controller, and the reward is the sum of average latency and load variance at each controller.

To the best of our knowledge, the joint problem of SDN routing and controller placement has not been studied to date, neither in its static version nor in the online learning one. Supervised learning techniques require labeled datasets which are not easy to create, especially in a tactical MANET setting with various limitations. On the other hand, reinforcement learning is known for requiring a significant amount of data, compute resources, and training time to train models. On the contrary, in this work we adhere to computationally lightweight online learning techniques which learn to adapt the policy based on simple observation feedback, and they have regret performance that renders them quite practical.

III. Joint SDN Routing and Controller Placement: Static version

A. Model and problem statement

1) Model: Consider a MANET depicted as a graph $G = (V, E)$, with a set of nodes (SDN switches) $\mathcal{V}$ and a set of links $\mathcal{E}$. A link (i, j) between switches i and j means that i and j can directly communicate. Time is slotted. Without loss of generality, we consider only one long-lasting session from a source s to a destination d. We assume that the network size is such that a single SDN controller suffices.

First, we present the model for the static problem, where parameters are known and fixed. This version applies either when we take a system snapshot, in which case the parameters' instantaneous values are known, or when we study the system in an average sense, so that parameter values stand for averages.

Let $c_{i,j}$ denote the cost of link (i, j). This may stand for packet delay, packet error rate, number of packet retransmissions, or number of time slots consumed for transmission on link (i, j). Let $\mathbf{c}$ denote the vector of link costs of the graph.

We assume single-path routing from s to d. A sequence of nodes $\{i_0, i_1 \ldots, i_n\}$ with $i_0 = s$ and $i_n = d$ is a path from s to d, if $(i_0, i_1), (i_1, i_2), \ldots, (i_{n-1}, i_n) \in \mathcal{E}$. Consider the links indexed in some way. Let variable $\mathbf{x} = (x_{i,j} : (i, j) \in \mathcal{E})$ be a binary vector of size $|\mathcal{E}|$, with a component equal to 1, if the corresponding link is on the path from s to d, and 0 otherwise. The set of all such paths is denoted by $\mathcal{P}$; it is the set of all binary vectors $\mathbf{x}$ whose components satisfy the constraints $\sum_i x_{i,d} = 1$, $\sum_i x_{s,i} = 1$, and for each $i \in \mathcal{V}$, the flow conservation constraint $\sum_j x_{i,j} - \sum_k x_{k,i} = 0$.

For $\mathbf{x} \in \mathcal{P}$, let $\mathbf{n}(\mathbf{x})$ be the binary vector of size $|\mathcal{V}| \times 1$. The components of $\mathbf{n}(\mathbf{x})$ that are 1 denote those switches along the path whose adjacent links are in $\mathbf{x}$.

Let $\mathcal{L}$ denote the set of L possible locations for placing the SDN controller. In general, these locations may or may not be nodes of the graph. Let $\mathbf{y}$ be the binary $L \times 1$ unit vector where a component $y_k = 1$ if the controller is placed at location $k \in \mathcal{L}$, while all others are 0. Let $\mathcal{Y}$ be the set of L unit vectors. For each possible controller placement location $k \in \mathcal{L}$ and switch $i \in \mathcal{V}$, let $d_{i,k}$ denote the delay between switch i and location k. This is the time required for a control message to be passed from the controller at location k to switch i via the control channel. Let $\mathbf{D}$ be the $|\mathcal{V}| \times L$ matrix whose (i, k)-th element is $d_{i,k}$.

2) Problem statement: During a long-lasting flow, both the traffic routing cost and the controller-to-switch (C-S) delays should be minimized. On the one hand, one should find the SDN data routing policy that minimizes traffic routing cost from s to d. On the other hand, the controller placement should be selected appropriately, since it affects the controller-to-switch delays (C-S delays) along the path. In fact, C-S delays are also affected by routing, thus routing and controller placement are interrelated in what concerns C-S delays. A certain route traverses a subset of switches, and thus it affects the placement of the controller, since the controller location should have small delays to *these* switches. Also, placing the controller at a certain location affects SDN routing, in the sense that the selected path should traverse switches with small C-S delays. If C-S delays are not taken into account, the controller will receive delayed and outdated network state information from switches and will not react in a timely manner to a potential link deterioration so as to initiate route recalculation.

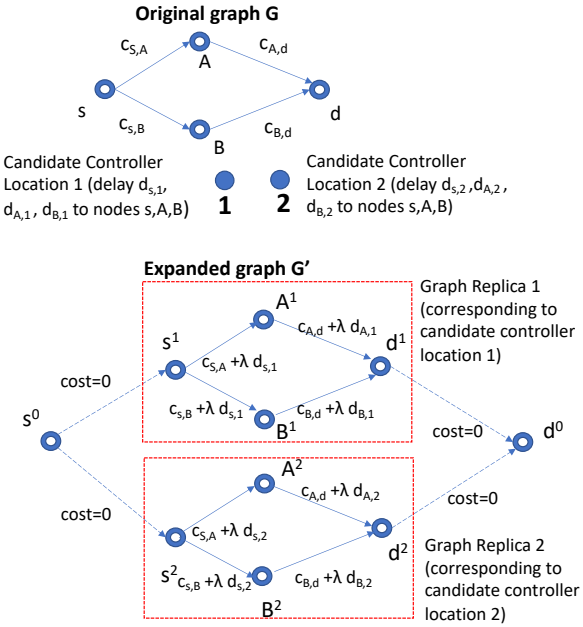

Figure 1. Example graph G and enhanced graph G' for $L = 2$ candidate controller locations 1,2. For visibility, these locations are taken to be outside of graph G. Graph G' consists of one replica for each controller location. There also exist auxiliary nodes s^0 and d^0, and appropriately defined link costs. For example, if the solution to the joint problem is path $s \to B \to d$ in G and controller location 1, this solution corresponds to path $s^0 \to s^1 \to B^1 \to d^1 \to d^0$ in G'.

Thus, in the static problem version where the link cost vector $\mathbf{c}$ and delay matrix $\mathbf{D}$ are assumed to be known and fixed, the joint routing and controller placement problem is formulated as follows:

$$\min_{\mathbf{x} \in \mathcal{P}, \mathbf{y} \in \mathcal{Y}} g(\mathbf{x}, \mathbf{y}) = \langle \mathbf{c}, \mathbf{x} \rangle + \lambda \langle \mathbf{n}(\mathbf{x}), \mathbf{D}\mathbf{y} \rangle \qquad (2)$$

where $\langle \mathbf{a}, \mathbf{b} \rangle$ is the inner product of vectors $\mathbf{a}$ and $\mathbf{b}$, and $\lambda > 0$ is an importance weight.

B. Solution of the static problem version

We construct an extended graph $G' = (\mathcal{V}', \mathcal{E}')$ out of $G = (\mathcal{V}, \mathcal{E})$. First, we define two auxiliary nodes s^0 and d^0. Next, for each candidate controller location $k = 1, \ldots, L$, we construct a replica $G^k = (\mathcal{V}^k, \mathcal{E}^k)$ of graph G. Let (i^k, j^k) denote link (i, j) in replica G^k. Also denote by s^k, d^k the source and destination node in replica G^k.

Next, we draw an edge (s^0, s^k) from s_0 to each node s^k, $k = 1, \ldots, L$, and we assign zero cost to it. Similarly, we draw a link (d^k, d^0) from each node d^k, $k = 1, \ldots, L$, to d^0,

again of zero cost. Finally, to each edge (i^k, j^k) of replica G^k, we assign cost:

$$w(i^k, j^k) = c_{i,j} + \lambda d_{i,k} \tag{3}$$

An example graph G and its expanded graph G' are depicted in Fig. 1.

Proposition 1. *A solution $(\mathbf{x}^*, \mathbf{y}^*)$ to the joint SDN routing and controller placement problem (2) in graph G from s to d is a minimum-cost path from s^0 to d^0 in graph G'.*

Therefore, in order to solve the joint problem (2) in the original graph G, it suffices to find a minimum-cost path from s^0 to d^0 in graph G'. The first link in that path is one of links $(s^0, s^1), (s^0, s^2), \ldots, (s^0, s^L)$, and the last link is one of $(d^1, d^0), (d^2, d^0), \ldots (d^L, d^0)$. Besides these first and last links, the minimum-cost path in G' will traverse links $(i_0^k, i_1^k), (i_1^k, i_2^k), \ldots, (i_{n-1}^k, i_n^k)$ of *one* replica G^k. This means that the solution to the joint problem is the path $i_0 \to i_1 \to \ldots \to i_n$ and controller location is k.

IV. Online Learning problem

A. Problem statement

We now consider the online learning problem, where the link cost vector $\mathbf{c}^t$ and delay matrix $\mathbf{D}^t$ vary according to arbitrary, non-stationary random processes. The non-stationarity assumption is motivated by mobility and other network dynamics. A single long-lasting flow is again assumed between s and d.

At each time t, a location k^t is selected to host the controller; namely vector $\mathbf{y}^t \in \mathcal{Y}$ is selected, with $y^t_{k_t} = 1$ and $y^t_j = 0$ for $j \neq k_t$. Also, a routing path $\mathbf{x}^t \in \mathcal{P}$ is selected. Let $\mathcal{R}^t \subset \mathcal{E}$ be the subset of links on the path at time t. Following the choices above, data traffic flows on the path, and the link cost and C-S delay feedback from switches on the path is received by the learner. The learner is a node that runs the learning algorithm, and as we will see in the sequel, this node may change during the learning process. In fact, the learner can be the selected controller each time.

Given a time horizon T, the problem is to find a policy, i.e. a sequence of routing and controller placement actions $\mathbf{z}^t = \{(\mathbf{x}^t, \mathbf{y}^t)\}_{t=1,\ldots,T}$, so as to minimize the average regret,

$$\bar{R}_T(\mathbf{z}) = \frac{1}{T} R_T(\mathbf{z}) = \frac{1}{T}\left(\mathbb{E}\big[\sum_{t=1}^{T} G^t(\mathbf{z}^t) \big] - \min_{\mathbf{x}\in\mathcal{P}, \mathbf{y}\in\mathcal{Y}} \sum_{t=1}^{T} G^t(\mathbf{z}) \right) \tag{4}$$

where $G^t(\mathbf{z}^t) = G^t(\mathbf{x}^t, \mathbf{y}^t) = \langle \mathbf{c}^t, \mathbf{x}^t \rangle + \lambda \langle \mathbf{n}(\mathbf{x}^t), \mathbf{D}^t \mathbf{y}^t \rangle$, and the expectation is with respect to the randomness of the distribution of selecting a path and a controller location at time t. The average regret measures the average performance difference over the horizon between the total cost of our policy and the cost of the optimal fixed policy in hindsight.

B. Solution of the learning problem

With Proposition 1, the online learning version of the joint controller placement and SDN routing becomes an online shortest-path (minimum-cost) routing problem. We consider two cases of feedback to the learner, namely full and limited feedback.

1) Full feedback: First, we consider the case where, after controller placement and route selection at time t, it is possible through specially designed control signaling to collect the link costs from *all* links in the network, and to measure C-S delays between *all* possible controller locations and *all* switches. Thus, after decision $(\mathbf{x}^t, \mathbf{y}^t)$, the full link cost vector $\mathbf{c}^t = (c^t_{i,j} : (i,j) \in \mathcal{E})$ and delay matrix $\mathbf{D}^t = (d^t_{i,k} : i \in \mathcal{V}, k \in \mathcal{L})$ are available to the learner.

For each time t, define the link cumulative cost vector $\hat{\mathbf{c}}^t = \sum_{r=0}^{t} \mathbf{c}^r = \hat{\mathbf{c}}^{t-1} + \mathbf{c}^t$, and the cumulative delay matrix $\hat{\mathbf{D}}^t = \sum_{r=0}^{t} \mathbf{D}^r = \hat{\mathbf{D}}^{t-1} + \mathbf{D}^t$.Also, for each link (i^k, j^k) in replica G^k, let $\hat{w}^t(i^k, j^k) = \sum_{r=0}^{t} w^r(i^k, j^k)$ be the cumulative link cost up to t, with $\hat{w}^t(i^k, j^k) = \hat{c}^t_{i,j} + \lambda \hat{d}^t_{i,k}$. Let $\hat{\mathbf{w}}^t = (w^t(i^k, j^k) : (i,j) \in \mathcal{E}, k = 1, \ldots, L)$ be the vector of cumulative costs of links (i^k, j^k) in all replicas.

We need to solve an online minimum-cost path learning problem on graph G'. This problem has a combinatorial decision set and a linear loss function. We start from the Follow-the-Perturbed-Leader (FTPL) algorithm which is known to achieve sublinear regret. At each time t, FTPL selects the action that minimizes the perturbed cumulative loss up to t, where a random perturbation is added to the cumulative loss, based on a uniform or exponential probability distribution. For our setting, we develop the following algorithm to run on G', whose steps at each time t are as follows:

- **STEP 0:** $t = 0$; initialize link costs $\mathbf{c}^0 = \mathbf{0}$ and C-S delays $\mathbf{D}^0 = \mathbf{0}$. For each link (i^k, j^k) of G', set $w^0(i^k, j^k) = 0, \hat{w}^0(i^k, j^k) = 0$.
- **STEP 1:** For each link (i^k, j^k), generate an independent exponentially distributed random variable $Z^t(i^k, j^k)$, which acts as the perturbation. Set the cumulative cost of link (i^k, j^k) to $\hat{w}^t(i^k, j^k) - Z^t(i^k, j^k)$.
- **STEP 2:** Find the min-cost path on G', and from that find the next path $\mathcal{R}^t$ (or $\mathbf{x}^t$), and controller location k^t (controller location vector $\mathbf{y}^t$).
- **STEP 3:** If $t \geq 1$ and the new controller location is different than the current one, i.e. if $k^t \neq k^{t-1}$, the current controller k^{t-1} transfers to next controller k^t the current vector $\hat{\mathbf{c}}^{t-1}$ and current delay matrix $\hat{\mathbf{D}}^{t-1}$.
- **STEP 4:** Set k^t, $\mathcal{R}^t$ as current controller and current path. Traffic is then routed through that path.
- **STEP 5:** The current controller k^t receives as feedback the link costs from links $(i,j) \in \mathcal{E}$. It then updates the link cumulative cost vector $\hat{\mathbf{c}}^t$.
- **STEP 6:** The current controller k^t receives as feedback the delay measurements between switches $i \in \mathcal{V}$ and all possible controller locations $k \in \mathcal{L}$. It then updates the cumulative delay matrix $\hat{\mathbf{D}}^t$.
- **STEP 7:** For each edge (i^k, j^k) in G^k, update the cumulative cost: $\hat{w}^t(i^k, j^k) = \hat{c}^t_{i,j} + \lambda \hat{D}^t_{i,k}$.
- **STEP 8:** Set $t \leftarrow t + 1$. Go to STEP 1.

2) Limited feedback: We now consider the case where at each time slot, after the controller placement and routing

decision, the learner receives as as feedback *only* the costs of the links along the selected path and *only* the delays from the current controller to the switches along the path. Such a model is more practical and it arises under state-of-the-art SDN control protocols that issue messages only between the current controller and the switches and collect feedback only from the links along the selected path.

This case of limited feedback is the semi-bandit case, and it is an intermediate scenario between the full-feedback case, in which the learner receives all link costs and C-S delays from all possible controller locations to all switches, and the bandit-feedback case, in which the learner receives only the total value of loss, but not individual cost and delay values.

Let M be a hyper-parameter for the maximum number of trials. The FTPL-based algorithm steps for the semi-bandit feedback case are:

- **STEP 0:** $t = 0$; initialize $\mathbf{c}^0 = \mathbf{0}$ and $\mathbf{D}^0 = \mathbf{0}$. For links (i^k, j^k) of G^k, set $w^0(i^k, j^k) = 0, \hat{w}^0(i^k, j^k) = 0$.
- **STEP 1:** For each link (i^k, j^k), generate an independent exponentially distributed random variable $Z^t(i^k, j^k)$, and set link cost to $\hat{w}^t(i^k, j^k) - Z^t(i^k, j^k)$.
- **STEP 2:** Find the min-cost path in G' and thus the next path $\mathcal{R}^t$ (or $\mathbf{x}^t$) and next controller location vector $\mathbf{y}^t$. Set $k^t \in \{1, \dots \mathcal{L}\}$ as the next controller, i.e. $y^t_{k_t} = 1$.
- **STEP 3:** For each link (i^k, j^k) in G^k, draw M independent exponential random variables $\{Z_m(i^k, j^k)\}$, $m = 1, \dots, M$. For each m, set cost to (i^k, j^k) to $\hat{w}^t(i^k, j^k) - Z_m(i^k, j^k)$, and compute min-cost path in G', call it $\mathcal{R}^t_m$.
- **STEP 4:** For each link $e = (i^k, j^k)$, $(i, j) \in \mathcal{R}^t$, and for $k = k^t$, record the earliest instance when e belongs in a min-cost path, $m^*_e = \min\{m : e \in \mathcal{R}^t_m\}$. If $1 \leq m^*_e \leq M$, link e was found in a min-cost path; set $N^t_e = m^*_e$. Else, set $N^t_e = M$.
- **STEP 5:** If $t \geq 1$ and $k^t \neq k^{t-1}$, the current controller k^{t-1} transfers to next controller k^t the vector $\hat{\mathbf{w}}^{t-1}$.
- **STEP 6:** Set current controller to k^t and path to $\mathbf{x}^t$. Traffic is routed through that path.
- **STEP 7:** Controller k^t receives link costs from links on the path and delays from k^t to switches on the path.
- **STEP 8:** For those links $e = (i^k, j^k)$ in replica G^k, such that $(i, j) \in \mathcal{R}^t$ and $k = k^t$, update cumulative cost as: $\hat{w}^t(i^k, j^k) \leftarrow \hat{w}^t(i^k, j^k) + N^t_{m^*_e} w^t(i^k, j^k)$.
- **STEP 9:** Set $t \leftarrow t + 1$. Go to STEP 1.

For L controller locations, graph G' has $L(|\mathcal{E}| + 2)$ links, and this is the dimension of the decision set. If P denotes the maximum path length in G, the maximum path length in G' is $P + 2$. Following the findings from [11], FTPL achieves a regret of $O\left((P+2)^{3/2}\sqrt{T \log \frac{(|\mathcal{E}|+2)L}{P+2}}\right)$ for full feedback and $O\left((P+2)\sqrt{TL(E+2)\log(L(E+2))}\right)$ for limited (semi-bandit) feedback.

V. Numerical results

We consider the toy graph of Fig. 1 with two paths from s to d, i.e. $\mathcal{P} = \{s \to A \to d, s \to B \to d\}$, and two possible controller locations, $\mathcal{L} = \{L_1, L_2\}$. This leads to enhanced graph G' in Fig. 1 with four options for joint routing and controller placement, each of which corresponds to a path from s_0 to d_0.

We study the scenario of full feedback and compare the Follow-the-Leader (FTL) and Follow-the-Perturbed-Leader (FTPL) algorithms with respect to the following performance metrics:

- Regret per time slot (or time-average regret) $\overline{R}_T$, over the horizon T, given by (4).
- Cumulative regret over T, R_T.
- Percentage of time instants P_T, in which the best option (path in G') is selected over time horizon T. If K_T is the number of times over horizon T when the best option is selected, then $P_T = K_T/T$.

We simulate 2 cases for non-stationary link costs and delays to the controller:

- Case A: Link costs and C-S delays at time t for each option $j = 1, 2, 3, 4$ are given as $r_j \cdot |\cos t| + 3j \cdot r'_j \cdot r''_j$, where r_j, r'_j, r''_j are uniformly distributed in $(0, 1)$.
- Case B: This corresponds to an intelligent adversary that attempts to disrupt the learning process, by observing at each time t the selection of the learner, and by setting the cost of this option to 10 in the next slot.

Results are averaged over 100 experiments. In Figures 2 and 3, we show the Cumulative Regret and Average Regret respectively for the FTL and FTPL algorithms for Cases A and B. For Case A, both Cumulative Regrets stabilize, and in fact FTL achieves a little lower Cumulative and Average Regret than FTPL. However, for Case B, FTL suffers a linear Cumulative regret with time, while FTPL achieves sublinear regret (Fig. 2). Equivalently, in case B, the time-average regret for FTL is constant, while for FTPL it is decreasing with time (Fig 3). Thus, FTPL turns out to be superior to FTL in handling adversarial scenarios. Finally, in Figure 4 we depict the percentage of correct option selection for FTL and FTPL. While in the first 100 iterations FTL is slightly better than FTPL, eventually both algorithms reach almost the same percentage.

VI. Conclusion

We studied the joint problem of routing and controller placement in mobile SDN networks in its static version and its online learning version. We showed that the joint problem becomes one of minimum-cost routing on an appropriately defined graph. Subsequently, we identified the online learning problem as an online combinatorial optimization one, and we developed low-complexity algorithms based on the FTPL approach for the cases of full and limited feedback. The former case requires a modification to the SDN control signaling protocol, while the latter applies for the current SDN control messaging protocols. These algorithms are simple to implement in a distributed tactical MANET environment, where scalability is required.

There exist several directions for future work. First, the problem can be extended to one with multiple flows, and

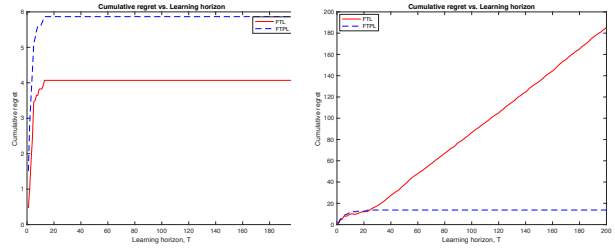

Figure 2. Cumul. Regret for FTL and FTPL. Left: Case A; right: Case B.

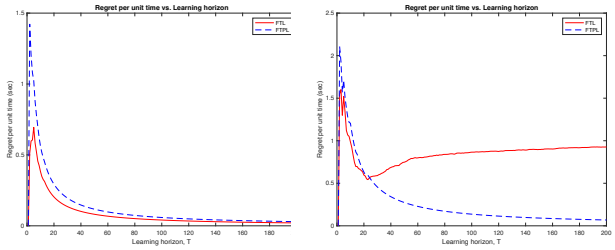

Figure 3. Average Regret for FTL and FTPL. Left: Case A; right: Case B.

also multiple controllers. Second, in this work, we used an additive delay metric from the controller to switches, which facilitated the definition of link costs in the enhanced graph. If the delay metric becomes nonlinear, e.g. the maximum of the delays from the controller to all switches, this additive relation does not exist any more, and a new approach is needed. Another extension is to include switching costs for controller location and routing path changes. Finally, from the simulations, it seems that the relative performance of FTL and FTPL depends on the nature of the non-stationary cost process. FTPL is better in terms of regret when the cost process is more "adversarial", while FTL is better in handling ordinary' non-stationary processes. It would be thus interesting to devise an algorithm that progressively learns the regime, adversarial or not, under which the network operates, so that the best learning algorithm is applied.

ACKNOWLEDGMENT

This work has been carried out in the context of the project entitled Software defined MObile Tactical Ad hoc NETwork (SMOTANET), which has received funding from the European Defence Industrial Development Programme (EDIDP) under grant agreement No EDIDP-CSAMN-SDN-2019-038-SMOTANET. This paper reflects only the authors' views and the European Commission is not responsible for any use that may be made of the information contained herein.

REFERENCES

[1] I. F. Akyildiz, S.-C. Lin, and P. Wang, "Wireless software-defined networks (W-SDNs) and network function virtualization (NFV) for 5G cellular systems: An overview and qualitative evaluation, *Comput. Netw.,* vol.93, no.12, pp.66–79, 2015.

[2] ARL project. Dais ITA: The distributed analytics and information science international technology alliance, 2016–2021.

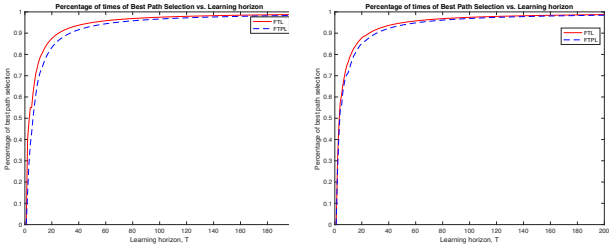

Figure 4. Percent of correct option selection for FTL and FTPL. Left: Case A; right: Case B.

[3] K. Poularakis, Q. Qin, E. Nahum, M. Rio and L. Tassiulas, "Bringing SDN to the mobile edge," *Workshop on Distributed Analytics InfraStructure and Algorithms for Multi-Organization Federations, (in proc. IEEE Smart World Congress),* 2017.

[4] K. Poularakis, G. Iosifidis and L. Tassiulas, "SDN-Enabled Tactical Ad Hoc Networks: Extending Programmable Control to the Edge," *IEEE Commun. Mag.*, vol. 56, no. 7, pp. 132-138, July 2018.

[5] H. B. McMahan, "A survey of algorithms and analysis for adaptive online learning", *The Journal of Machine Learning Research,* vol.18, no.1, pp. 3117–3166, 2017.

[6] E. Hazan, Introduction to online convex optimization, Foundations and Trends in Optimization, Now Publishers, 2016.

[7] A. S. Suggala and P. Netrapalli, "Online non-convex learning: Following the perturbed leader is optimal", *in Proc. Int. Conf. on Algorithmic Learning Theory,* 2020.

[8] M. Hutter and J. Poland, "Prediction with expert advice by following the perturbed leader for general weights," *in Proc. Algorithmic Learning Theory (ALT)*, 2004.

[9] A. Kalai and S. Vempala, "Efficient algorithms for online decision problems", *Journal of Computer and System Sciences,* vol.71, no.3, pp.291-307, 2005.

[10] J.-Y. Audibert, S. Bubeck, G, Lugosi, "Regret in Online Combinatorial Optimization", *Mathematics of Operations Research,* vol. 39, no. 1, pp. 31–45, 2014.

[11] G. Neu and G. Bartok, "Importance weighting without importance weights: an efficient algorithm for combinatorial semi-bandits", *J. Mach. Learn. Res.* vol.17, no. 1, Jan. 2016.

[12] E. Akin and T. Korkmaz, "Comparison of Routing Algorithms With Static and Dynamic Link Cost in Software Defined Networking (SDN)," *in IEEE Access,* vol. 7, pp. 148629-148644, 2019.

[13] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," *in Proc. IEEE INFOCOM,* 2013.

[14] R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Casillas-Perez and J. M. Arco, "A Survey on Machine Learning Techniques for Routing Optimization in SDN," *in IEEE Access,* vol. 9, pp. 104582-104611, 2021.

[15] B. Heller, R. Sherwood, and Nick McKeown, "The controller placement problem", *Proc. 1st ACM workshop on Hot topics in software defined networks (HotSDN)* 2012.

[16] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement prob- lem in software defined networks", *IEEE Commun. Lett.*, 18(8):1339–1342, 2014.

[17] Q. Qin, K. Poularakis, G. Iosifidis, S. Kompella and L. Tassiulas, "SDN Controller Placement With Delay-Overhead Balancing in Wireless Edge Networks", *IEEE Trans. on Network and Service Management,* vol. 15, no. 4, pp. 1446-1459, Dec. 2018.

[18] I. Koutsopoulos, "Learning the Optimal Controller Placement in Mobile Software-Defined Networks," *IEEE Int. Symp. on a World of Wireless, Mobile and Multimedia Networks (WoWMoM),* 2022.

[19] Y. Wu, S. Zhou, Y. Wei and S. Leng, "Deep Reinforcement Learning for Controller Placement in Software Defined Network," *Proc. IEEE INFOCOM Workshops*, 2020.

[20] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The nonstochastic multiarmed bandit problem", *SIAM J. Comput.*, vol.32, no.1, pp.48–77, Jan. 2003.