Reroute Sequence Planning : Scheduling Reconfigurations in Software Defined Networks

Lorenzo Maggi, Stefano Paris, Paolo Medagliani, Jérémie Leguay

Mathematical and Algorithmic Sciences Lab, France Research Center, Huawei Technologies Co. Ltd.
{name.surname}@huawei.com

Mots-clés : *Routing optimization, sequence planning, SDN.*

1 Introduction

In computer networks, the online optimization of routing through the use of an external powerful controller is now possible using Software-Defined Networking (SDN) technologies [1]. To maintain the best configuration, the SDN controller has to constantly solve a Multi-Commodity Flow (MCF) problem and compute a sequence of feasible re-routing steps to move from the current configuration to the new target. Ideally these two problems should be jointly solved by the controller but instances are very large (e.g., 10k nodes, 40k links, 10k flows). In this work, we decouple the two problems and aim at finding a fast and scalable algorithm for the *reroute sequence planning* problem (also called hitless or zero-loss scheduling of network updates). As SDN controllers are adopting powerful distributed computing architectures, we propose a parallel algorithm to speed up computation. This topic has gained a lot attention in the last two years with the fast deployment of SDN in data centers [2, 3].

2 Problem Formulation

In this section, we formulate the RSP problem as a Mixed Integer Linear Programming (MILP) model. We consider a network topology represented by graph $G(\mathcal{V}, \mathcal{E})$. Each vertex $v \in \mathcal{V}$ corresponds to a network device, whereas each edge $e \in \mathcal{E}$ corresponds to a network link and is characterized by its capacity b_e and its cost c_e per unit of flow. A set of demands $\mathcal{K}$ needs to be rerouted from an initial to a final path in at most T reconfiguration steps. Each demand $k \in \mathcal{K}$ is associated with a tuple $\langle o_k, t_k, d_k, P_0^k, P_T^k \rangle$. The parameters o_k and t_k represent the source and destination nodes of the demand, whereas d_k is the amount of traffic that must be routed in the network. P_T^k and P_T^k represents the initial and final paths, respectively. To avoid service interruption, the reconfiguration steps must follow the Make-Before-Break (MBB) approach. In this approach, routing rules are not immediately deleted when there is a change in the path connecting the origin to the destination. Instead, routing rules are maintained in the switches for an additional step with respect to the reconfiguration. This avoids the drop of packets that are flowing on the old path when the change occurs. We denote with $\mathcal{P}_k$ the set of paths available to route demand k, and with $\mathcal{P}_e$ the set of paths that traverse edge e.

For the formulation of the RSP problem we define the following decision variables. $U^k \in \{0, 1\}$ captures the rerouting decision for demand k. Decision variable $X_{pt} \in \{0, 1\}$ provides the path $p \in \mathcal{P}_k$ used at time t, while $Y_{et}^k \in \{0, 1\}$ models the MBB and it is used to identify the links used by a demand k at reconfiguration steps $t-1$ and t. Finally, real variable $Z_{et} \in [0, b_e]$ captures the overall bandwidth of link e reserved for reconfiguration step t. The RSP problem can be formulated as follows :

$$\min \quad \sum_{t=1}^{T} \sum_{e \in \mathcal{E}} c_e Z_{et} + \sum_{k \in \mathcal{K}} M U^k \tag{1}$$

$$s.t. \quad \sum_{p \in \mathcal{P}_k} X_{pt} = 1 \qquad \forall k \in \mathcal{K}, t = 1, 2, ..., T \quad (2)$$

$$Y^k_{et} \geq \sum_{p \in \mathcal{P}_k : e \in p} X_{p(t-1)} \qquad \forall k \in \mathcal{K}, t = 1, 2, ..., T \quad (3)$$

$$Y^k_{et} \geq \sum_{p \in \mathcal{P}_k : e \in p} X_{pt} \qquad \forall k \in \mathcal{K}, t = 1, 2, ..., T \quad (4)$$

$$\sum_{p \in \mathcal{P}_e} \sum_{k \in \mathcal{P}_k : p \in \mathcal{P}_k} d_k Y^k_{et} \leq Z_{et} \qquad \forall e \in \mathcal{E}, t = 1, 2, ..., T \quad (5)$$

$$X_{P^k_0 0} = 1 \qquad \forall k \in \mathcal{K} \quad (6)$$

$$X_{P^k_T T} = U^k \qquad \forall k \in \mathcal{K} \quad (7)$$

$$X_{P^k_0 T} = 1 - U^k \qquad \forall k \in \mathcal{K} \quad (8)$$

The objective function (2) maximizes the number of rerouted demands [1] and at the same time minimizes the reconfiguration cost. Constraints (2) forces the use of a single path to route each demand during each reconfiguration step. Constraints (3) and (4) formalize the MBB approach. When a path reconfiguration occurs in step t, MBB requires to keep in the memory of the switches the rules of the path used in the previous step $t-1$. Therefore, these constraints activate the variables Y^k_{et} corresponding to the union of links used by a demand k during both steps $t-1$ and t. The set of constraints (5) represents the capacity constraints. Constraints (6) represents the initial path configuration for any demand, whereas constraints (7) and (8) corresponds to the final path configuration. Depending on the rerouting decision U^k for demand k, either path P_0 or P_T is used at the end of the rerouting procedure.

3 Parallel Algorithm

The RSP problem is NP-Hard since it contains the integral multicommodity flow problem. In order to solve large scale instances in a short time (e.g., a few seconds), heuristics that compute a sequence of shortest paths for each demand can be used to efficiently compute a solution. To drive the decision of the heuristics towards the optimal solution, the sequence of shortest paths is computed using a modified graph where the edge distance depends on the cost, whether the edge belongs both to the initial and final paths of the demands, and the difference between the capacity and the actual traffic that traverses the edge. Since the complexity of the algorithm is dominated by the number of links ($|\mathcal{E}|$), demands ($|\mathcal{K}|$), and reconfiguration steps (T), we further propose to partition the set of demand $\mathcal{K}$ in N subsets $\mathcal{K}_i \subseteq \mathcal{K}$, $i = 1, ..., N$. The T reconfiguration steps are also split into N consecutive subintervals, which are assigned to the N subsets of demands $\mathcal{K}_i$. This type of partitioning fully benefits from distributed computing architectures since it can be iterated, thus reducing exponentially the problem size. Nonetheless, trade-offs among partitioning, optimality, and complexity needs to be further investigated.

Références

[1] D. Kreutz, F. M. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking : A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[2] H. H. Liu, X. Wu, M. Zhang, L. Yuan, R. Wattenhofer, and D. Maltz, "zupdate : Updating data center networks with zero loss," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, Aug. 2013.

[3] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer, "Dynamic scheduling of network updates," in *Proc. ACM SIGCOMM*, 2014.

1. M is a large value that dominates the second term (e.g., $M = 1 + \max\{b_e c_e\}(|\mathcal{V}| - 1)T$)