Migration Scheduling in Distributed SDN Controllers

Mohammad Amin Beiruti, Yashar Ganjali
Department of Computer Science, University of Toronto

***Abstract*—Load migration is essential in any distributed SDN control platform due to natural load imbalance and dynamic nature of input traffic. Existing solutions focus on migrating a single switch between two controller instances. Migrating multiple switches requires careful planning due to controller resource constraints, and to ensure minimum service interruption in the network. In this poster, we present a model and a solution for migration scheduling, taking a set of switch migrations as input, generating a migration schedule with respect to controller resource and service interruption constraints.**

I. Introduction

With a distributed control plane, having load imbalance among different network controllers is inevitable. Recently, new proposals have tried to address this issue through load migration [1], [2]. In a nutshell, these solutions present safe and reliable mechanisms to hand over a given switch from a highly loaded controller to a lightly loaded controller in order to reduce load imbalance in the control plane.

While migrating a single switch is straight forward, migrating multiple switches at the same time is not trivial and requires careful planning for two reasons. First, migrating a given switch requires significant processing and memory at source and target controllers. During the handover process, the source of migration pauses processing any messages from the switch. These messages are buffered and processed after the handover [1], [2]. The memory needed for buffering and CPU required to process these messages limit the number of concurrent migrations that a particular controller instance can handle. We call these limits *controller resource constraints.*

Additionally, we cannot migrate an arbitrary group of switches at the same time. During the migration process, any messages from the switch to the controllers involved in migration are processed with a delay. In order to ensure network services are not interrupted and network QoS is not adversely affected, we need to avoid concurrent migration of certain sets of switches. We refer to this as the *QoS constraints*.

In this poster, we present a model and a solution for migration scheduling problem. Our solution takes a set of switch migrations as input and finds a schedule for migrating switches in the shortest possible time, under controller resource and QoS constraints.

II. Switch Migration Specifications

Before formalizing the problem, we study the characteristics of the protocol(s) that our scheduler is built on, as those protocols can directly affect the complexity of our solution. Existing switch migration protocols [1], [2] transfer a specific switch from a master controller instance to a slave controller instance [3]. The destination controller buffers all messages from the switch during the migration process [1]. If failure resilience is required, the source controller also buffers messages from the switch [2], providing a fall back mechanism in case the destination controller fails during migration.

Once the role of the master and slave controllers are changed, the new master processes all the buffered messages before resuming its normal operation. This creates a transient load that requires significant CPU usage. We associate a positive weight to each switch migration to represent these potential overheads.

III. Problem Model

We assume switch migrations happen in several rounds. We schedule switch migrations in a way that no conflicts exist during each round. Our goal is to minimize the total number of rounds required to complete all migrations, while respecting controller resource and QoS constraints.

We model the switch migration scheduling problem as an Integer Linear Program (ILP). We assume there are n controller instances in the network, denoted by $C = \{c_1, c_2, \ldots, c_n\}$, and m switches $S = \{s_1, s_2, \ldots, s_m\}$.
Inputs. Let us assume we have k different switch migrations to schedule ($M = \{m_1, m_2, \ldots, m_k\}$). The i-th switch migration, m_i, has a weight of m_i^w. To simplify our presentation, let us consider two $k \times n$ binary matrices P and Q. We set P_{ij} equal to 1 if controller c_j is the source of migration m_i, and equal to 0 otherwise. Similarly, Q_{ij} is 1 if controller c_j is the destination of migration m_i, and 0 otherwise.
Controller Resource Constraints. Each controller instance has resource constraints (memory and computational capacity) limiting the maximum number of concurrent switch migrations it can handle in each round. We denote this limitation by a_i for each controller instance c_i.[1]
QoS Constraints. To satisfy QoS constraints, we need to ensure switches associated with a specific service (or a critical network path) are not all migrated concurrently. To model these constraints, we consider l groups of switches $g_1, g_2, \ldots, g_l$. Each group g_i is also associated with a positive

[1]We model controller resource constraints as a one-dimensional value. It is straight forward to extend this model to multiple dimensions.

number α_i, that indicates the maximum number of concurrent migrations that we can handle in this group without major service interruptions. Each group has at least two switches, and different groups can have non-empty intersections. We define a binary matrix W with W_{ij} equal to 1 when migration m_i is part of group g_j.

Variables. Let us assume we have an upper bound, R, on the number of rounds for executing all the migrations[2]. We use a $k \times R$ binary matrix A to represent our schedule: A_{ij} is set to 1 if migration m_i happens during round j, and $A_{ij} = 0$ otherwise.

ILP Formulation. We define our problem as an instance of an integer linear program as follows.

Objective Function: $min \sum_{i=1}^{k} \sum_{j=1}^{R} A_{ij} \times e^j$. Here e is a constant. We are taking the weighted some of the elements of the matrix A. By exponentially growing the weights for each round, we ensure there is pressure towards reducing the number of rounds in the schedule.

Constraints:

1) $\forall i, 1 \leq i \leq k : \sum_{r=1}^{R} A_{ir} = 1$. This constraint ensures that each switch migration happens once and only once in the schedule.
2) $\forall i, 1 \leq i \leq n, \forall j, 1 \leq j \leq R : \sum_{t=1}^{k} A_{tj} \times m_t^w \times Q_{ti} \leq a_i$. This is for the destination controller's resource limitation. At each round, the sum of weights of migrations associated with the same destination should not exceed each destination controller's available capacity.
3) $\forall i, 1 \leq i \leq n, \forall j, 1 \leq j \leq R : \sum_{t=1}^{k} A_{tj} \times m_t^w \times (P_{ti} + Q_{ti}) \leq a_i$. If we need failure resiliency, the limitation of the source of migration will also matter. In this case, this constraint will replace Constraint 2. We note that a controller cannot simultaneously be a source and destination of a particular migration, so P_{ti} and Q_{ti} cannot both be equal to 1.
4) $\forall j, 1 \leq j \leq R, \forall u, 1 \leq u \leq |G| : \sum_{i=1}^{k} A_{ij} \times W_{iu} \leq \alpha_u$. This ensures QoS constraints, i.e. for each group of conflicts, g_u, the number of migrations of this group does not exceed the predefined limit of α_u.
5) $\forall i, 1 \leq i \leq k, \forall j, 1 \leq j \leq R : A_{ij} \in \{0, 1\}$.

If there is a feasible solution, our ILP will return a scheduling plan with the minimum number of rounds.

IV. Evaluation

To evaluate our proposed scheduling scheme, we created a sample scenario with 5 controller instances and 15 switches. The topology, controller resource constraints, and QoS constraints are shown in Figure 1. Before any migration happens, each controller instance is responsible for three switches (dashed lines represent this relationship). Our goal is to perform 10 switch migrations $(m_1, m_2, ..., m_{10})$, represented by arrows pointing towards the destination in Figure 1. We assume all of these migrations have the same weight equal to 1.

[2]Clearly, the number of switch migrations is a bound, but in practice, we expect a much lower bound depending on the constraints.

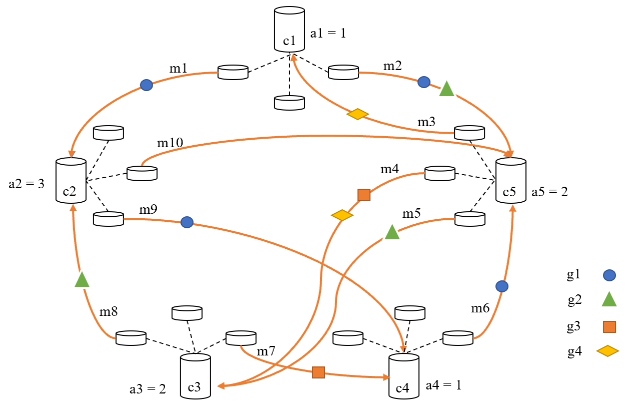

Fig. 1. Network topology, as well as resource and Qos constraints.

We also consider 4 different groups for QoS constraints, each group is represented by a different shape in the figure. Migrations with the same shape belong to the same group of constraints. In this example, $\forall i, \alpha_i$ is set to 1.

Existing solutions that do not consider migration scheduling will resort to executing one migration at a time to avoid conflicts and resource limitations. In this example, this will require 10 rounds to complete all switch migrations. Using a simple ILP solver, we were able to find a schedule that completes these 10 switch migrations in only 4 rounds. Here, the optimally of 4 rounds is obvious, as g_1 has 4 members and $\alpha_1 = 1$ ensures we cannot schedule any of these 4 switch migrations in the same round. This is meant to serve as a preliminary example that shows the potential benefits of the proposed scheme.

V. Conclusion

Reducing the total migration time can have a major impact on the efficiency of the SDN control plane. In this poster, we presented a model and a solution for switch migration in a distributed SDN control platform. This complements existing work on single switch migration by carefully planning switch migrations. We presented a simple example to highlight the potential benefits of the proposed scheme. Further enhancements to run-time (including techniques to solve the problem in a distributed manner, or heuristics for faster convergence) are left as future work.

References

[1] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. R. Kompella, "ElastiCon; an elastic distributed SDN controller," in *2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. New York, NY, USA: IEEE, 2014, pp. 17–27.

[2] M. Beiruti and Y. Ganjali, "Load Migration in Distributed SDN Controllers," Systems and Networking Group, University of Toronto, Tech. Rep. TR19-SN-UT-0801-01, Aug. 2019. [Online]. Available: http://www.cs.toronto.edu/ yganjali/TR-SN-UT/TR19-SN-UT-0801-01.pdf

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.