Dynamic Migration of Real-Time Traffic Flows in SDN-Enabled Networks

Peter Danielis*, György Dán†, James Gross† and André Berger‡
*Department of Computer Science, University of Rostock, Germany
†School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Sweden
‡School of Business and Economics, Maastricht University, The Netherlands
E-mail: peter.danielis@uni-rostock.de,{gyuri,james.gross}@kth.se, a.berger@maastrichtuniversity.nl

***Abstract*—In this paper, we investigate the problem of dynamic migration for realtime traffic flows, which consists in accommodating new flows at runtime in SDN-enabled networks. We show results for two algorithms that can calculate direct and indirect flow migrations at runtime. Numerical results obtained on a FatTree network topology show that flow migration is typically required for networks with a modest number of flows, while direct flow migration is possible in about 60% of the cases.**

I. Introduction

New application requirements create new challenges for networks. These new requirements arise in particular from the progressing digitalization in the prospective Smart Factory and tactile applications that imply networking with low latencies [1]. In this context, some works have presented new solutions for real-time communication that achieve latencies of less than 1 ms and are independent of the network topology used [2], [3]. By using software-defined networking (SDN) technology, these approaches are able to utilize the SDN controller to explore the topology and query all application requirements (e.g., bandwidth, maximum latency). However, these systems are only able to consider the requirements at design time for a defined network configuration and to set up corresponding paths between network participants.

In this paper, we address the problem of flow migration, which enforces non-overlapping paths of flows when an additional real-time flow is integrated into a network at runtime. We summarize the algorithms for dynamic flow migration that have already been published in [4]. These migrate paths at runtime to integrate the new real-time flow without interrupting the ongoing communication of existing flows. Our work extends existing approaches by an algorithm with a polynomial runtime, which can decide whether direct flow migration is possible. We also present another algorithm that finds indirect flow migrations as well. We use the algorithms to investigate in which scenarios a flow migration is necessary to integrate a new flow and we determine the number of migration steps if a flow migration is necessary.

II. System Model and Problem Formulation

We assume an SDN-enabled network below and model it with a directed graph $\mathcal{G} = (V, E)$, where V is the set of all vertices that represent switches and hosts, and E is the set of all edges. There is a set of $\mathcal{C} = \{(s_i, t_i) : 1 \leq i \leq N, s_i, t_i \in V\}$ with N source-destination host pairs that want to exchange data over the network. We call a sequence of edges in $\mathcal{G}$ from s_i to t_i a path between a source-destination pair (s_i, t_i), $s_i, t_i \in V$ and define the path length P, which is expressed as $l(P)$, as the number of edges contained. We assume that the traffic from s_i to t_i is subject to a certain constraint with regard to the delay and that the constraint can be met if the path from s_i to t_i has a maximum length $L_i \in \mathbb{N}^+$. Furthermore, for a pair (s_i, t_i) we define that there is a set $\mathcal{P}_i = \{P_i | l(P) \leq L_i\}$ with length-constrained paths and $P_i \in \mathcal{P}_i$ denotes an element of this set. Any path $P_i \in \mathcal{P}_i$ allows s_i to send data to t_i. For a given set $\mathcal{C}$ with source-destination pairs, there is a path collection $R = \{P_1, \ldots, P_N\}$ that is valid if P_i and P_j do not have overlapping edges. Such a path collection enables the source-destination pairs in $\mathcal{C}$ to communicate simultaneously. For the host pairs in $\mathcal{C}$, we call the set of all possible path collections $\mathcal{R}_{\mathcal{C}}$. We assume that the SDN controller has cross-network information available that it can use to configure the forwarding rules of one switch at a time via the SDN southbound API, e.g., by using an atomic commit protocol coordinator as described in [5]. We call this step-by-step configuration process *atomic forwarding table configuration*, which could be implemented by using OpenFlow Bundles [6]. The restriction that only one switch can be configured at a time avoids unpredictable network behavior due to race conditions caused by poor synchronization of the switches. Atomic forwarding table configurations can thus be used as atomic operations for path migration, as the following definition describes.

Definition 1. *A flow migration (FM) from path collection R to path collection R' is a sequence $\mathcal{S} = (R^{(0)}, \ldots, R^{(K)})$ of $K \geq 1$ path collections $R^{(k)}$, such that $R^{(0)} = R$ and $R^{(K)} = R'$, and path collection $R^{(k)}$ can be obtained from path collection $R^{(k-1)}$ by a sequence of atomic forwarding table configurations.*

In the following, we focus on the feasibility of FM as the set $\mathcal{C}$ with source-destination pairs changes. Since the problem is trivial when a source-destination pair is removed, let us take a closer look at the case where a flow is added for a new source-destination pair. For a given set $\mathcal{C}$ with source-destination pairs, a path collection $R = \{P_1, \ldots, P_N\}$ for $\mathcal{C}$, and a source-destination pair (s_{N+1}, t_{N+1}), we want to find an FM $\mathcal{S}$ from path collection R to path collection $R' = \{P'_1, \ldots, P'_N\}$ so that there is a path P'_{N+1} for which $\{P'_1, \ldots, P'_N, P'_{N+1}\} \in \mathcal{R}_{\mathcal{C}'}$ is a path collection for the set $\mathcal{C}' = \mathcal{C} \cup \{(s_{N+1}, t_{N+1})\}$ with source-destination pairs. We call this the Flow Migration Problem (FMP).

III. Flow Migration Algorithms

An FMP can be infeasible if no set $\mathcal{R}_{\mathcal{C}'}$ can be found or if there is no valid FM to another path collection R'. In addition, it may be that the FMP is solvable, but that it does not require FM. In the simplest case, there is a path $P'_{N+1} \in \mathcal{P}_{N+1}$, so that $\{P_1, \ldots, P_N, P'_{N+1}\}$ is a path collection and therefore no FM is required. If such a path does not exist, i.e., none of the path $P'_{N+1} \in \mathcal{P}_{N+1}$ has non-overlapping edges with the paths in R, then the path collection R must be reconfigured to another path collection $R' \in \mathcal{R}^* = \{(P'_1, \ldots, P'_N) | \exists P'_{N+1} s.t. (P'_1, \ldots, P'_N, P'_{N+1}) \in \mathcal{R}_{\mathcal{C}'}\}$.

Below we outline two algorithms for the solution of the FMP, which assume that the set $\mathcal{R}^*$ of path collections can be calculated. The two algorithms compute a sequence of FMs. Each FM changes a single path at a time which we call elementary FM. Although the first algorithm for the Direct Flow Migration (DFM) has polynomial runtime, it can migrate each path only once, i.e., a DFM is a permutation of N source-destination host pairs, so individual paths from P_i to P'_i migrate stepwise with elementary FMs such that there is a sequence of path collections. In contrast to the DFM algorithm, the Generic Flow Migration (GFM) algorithm can migrate any path several times. This enables the GFM to determine Indirect Flow Migrations (IFMs) in addition to DFMs at the expense of increased runtime. To insert a new flow between the source-destination pair (s_{N+1}, t_{N+1}), an SDN controller executes the algorithms. If the new flow cannot be inserted without migrating the paths of the existing flows, it tries to find a DFM from the initial path collection R to the desired path collection R'. If there is a DFM, the controller installs appropriate atomic forwarding configurations (rules) on the switches via the SDN southbound API. If DFM is not possible, the controller uses the GFM algorithm to find an IFM and installs corresponding rules on the switches.

IV. Summary of Performance Evaluation

We perform the evaluation of our algorithms on the FatTree topology. It consists of 16 hosts connected by a network of 20 switches [7]. In the evaluation, we consider $N = \{1, ..., 7\}$ source-destination pairs with a length constraint of 6 edges. For each value of N, we generated up to 5000 different sets $\mathcal{C}$ of source-target pairs by randomly selecting the source and destination host without replacement. For each set $\mathcal{C}$, we use the All Paths and Cycle (APAC) algorithms from [8] to determine all possible path collections $\mathcal{R}^*$. For every set $\mathcal{C}$ and path collection R, we randomly choose the $N+1$-st host pair to create an instance of the FMP. Therefore we consider thousands of FMPs for every value of N.

It can be observed that the number of infeasible FMPs increases with increasing N, due to increasing probability that most of the paths on the FatTree topology are already occupied and no further path can be found. FM is typically not necessary for low values of N since there are enough free paths. However, if we count up N, the proportion of instances for which an FM is necessary increases and the proportion reaches its highest level for $N = 5$ at 30%, before declining again. This is because non-overlapping paths become available for medium values of N if existing paths are migrated and the migration is also possible with a high degree of probability. Nevertheless, the FM hardly helps for high values of N, since additional paths can either be found without any FM or, which is more often the case, the FMP is infeasible. The number of elementary FMs rises to a value of 5 for N approaching 5 and then remains constant. The results also show that for $62-92\%$ of all feasible FMPs requiring FM, the DFM is sufficient. This is important as the DFM can be calculated in polynomial time.

To summarize, we can conclude that for weakly and heavily loaded topologies an FM is either not necessary or the FMP is not feasible and therefore the FM cannot help. Nevertheless, for moderately loaded topologies (4 to 6 flows), a significant proportion of FMPs requires FMs (24% to 30%). In roughly 60% of these cases, a DFM solves the FMP.

V. Conclusion

In this paper, we investigated the problem of dynamic migration of real-time flows in SDN-enabled networks. We have used two algorithms that enable flow migration at runtime. The direct flow migration algorithm has polynomial runtime, but does not always find a solution to the flow migration problem. Another algorithm finds all solutions, if existing, but at the cost of an increased runtime. Our results show that in $24-30\%$ of the cases FM is necessary on a FatTree topology and typically requires multiple migration steps. It was also determined that direct FM is sufficient in $62-92\%$ of the cases.

References

[1] G. P. Fettweis, "The tactile internet: Applications and challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, March 2014.

[2] J. W. Guck and W. Kellerer, "Achieving end-to-end real-time quality of service with software defined networking," in *IEEE Intl. Conf. on Cloud Networking (CloudNet)*, Oct. 2014, pp. 70–76.

[3] E. Schweissguth, P. Danielis, C. Niemann, and D. Timmermann, "Application-aware industrial ethernet based on an sdn-supported tdma approach," in *IEEE World Conference on Factory Communication Systems (WFCS)*, May 2016, pp. 1–8.

[4] P. Danielis, G. Dán, J. Gross, and A. Berger, "Dynamic flow migration for delay constrained traffic in software-defined networks," in *GLOBECOM 2017*. IEEE, 2017, pp. 1–7.

[5] M. Curic, Z. Despotovic, A. Hecker, and G. Carle, "Fitsdn: Flexible integrated transactional sdn," in *2019 IEEE 44th LCN Symposium on Emerging Topics in Networking (LCN Symposium)*, Oct 2019, pp. 1–9.

[6] T. Ren and Y. Xu, "Analysis of the new features of openflow 1.4," in *2nd International Conference on Information, Electronics and Computer*. Atlantis Press, 2014.

[7] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008.

[8] R. Simões, "Apac: An exact algorithm for retrieving cycles and paths in all kinds of graphs," *Tékhne-Revista de Estudos Politécnicos*, no. 12, pp. 39–55, 2009.