Customizable real-time service graph mapping algorithm in carrier grade networks

Balázs Németh, János Czentye, Gábor Vaszkun, Levente Csikor, Balázs Sonkoly
Budapest University of Technology and Economics
Email: {nemeth.balazs, czentye, vaszkun, csikor, sonkoly}@tmit.bme.hu

***Abstract*—Today the notion of Service Function Chaining (SFC) returns to the focus of technological development in the area of network management and design. Researchers focus on exploiting the advantages of Network Function Virtualization (NFV) and Software Defined Networking (SDN) to bring SFC to a brand new level of fast and flexible, modern end-to-end service orchestration. The core of SFC architecture is the orchestration algorithm, which has ultimate decision making responsibility over compute and networking resource reservation. There are many sophisticated solutions for this task with long running times, so those cannot be executed real-time for every new request arriving within a few seconds. There is a lack of very fast (meta)heuristic orchestration algorithms to deal with enormous amount of service request. The goal of this paper is to demonstrate novel approaches to designing, evaluating and fine-tuning of real-time parameterizable orchestration algorithms for carrier grade networks.**

***Index Terms*—SDN, NFV, SFC, resource orchestration, Unify, mapping**

I. Introduction

Service Function Chaining (SFC) is not new, but the evolution driven by Software Defined Networking (SDN) and Network Function Virtualization (NFV) has re-spotted the importance of the concept. A Service Chain (SC) or more generally a Service Graph (SG) is a set of network functions, such as firewall, NAT, etc. interconnected in a given order to support an application. Formerly, building and deploying a service chain took a huge amount of time and effort. The network functions were expensive off-the-shelf devices, and cabling and connecting them in the right order required careful attention. Moreover, in order to induce these devices to function in the right manner, each device needed to be manually configured using their own management interface and language they "spoke". The chance for error was high, and a problem in one component could disrupt the entire network. In case of a new demand, or just the replacement of a device to support a higher scale often required a complete reorganizing of the whole placement. Since chains were constructed in a way to support multiple applications, it was often the case that data passed through unnecessary functions and servers slowing down its own end-to-end delay and consuming needless bandwidth and CPU cycles of other devices.

SDN and NFV simplified and fastened the provisioning of service chains, however, proper deployment of the functioning elements is still not an obvious task. Since NFV enables the network functions to move from the dedicated, special-purpose hardwares into softwares running in different environments (e.g., virtualized) over a general purpose, x86 based platform, the number of properties and coefficients has significantly raised. In order to deploy a SC, we need to collect many resource related properties, e.g., unused CPU cores, memory, space, capabilities and accordingly we need to prepare to the case when an unexpectedly high traffic demand influences the packet processing capabilities of other service chains and consumes more resource than it normally needs.

So far, several research have been arisen but most of them need to be supplemented to be a perfect solution. For instance, a similar problem, called Virtual Network Embedding (VNE) aims at finding a possible or in some sense optimal mapping of multiple given logical networks on a shared substrate network. In VNE [1], only the resources of the nodes are taken into account, therefore requirements for links such as end-to-end delay cannot be considered. Nevertheless, the problem itself proved to be NP-hard, and a mixed-integer-programming (MIP) was given to solve it. However, the algorithm had a significant drawback: collocation of nodes (e.g., placing/running NFs on the same server) is not supported and its running time is not fast enough for real-time execution. In a fairly different approach [2], a preprocessing algorithm was proposed to solve a network embedding problem, in particular, determining the best collocation in advance. A fast and efficient algorithm, called LoCo, was also proposed, however, it considers only a few important parameters, for instance, no latency, NF type, placement criteria could be given as an input. Moreover, one of the most important and fundamental property, the end-to-end quality requirement, is also not considered.

In this paper, we propose a novel fast algorithm that maps efficiently an enormous amount of constantly arriving service graphs in huge network consisting tens-to-hundreds of thousands of nodes. Our algorithm is implemented as an add-on to an emerging prototyping environment, called ESCAPEv2 (Extensible Service ChAin Prototyping Environment, [3]), wherein it could become an important supplement.

II. Architecture and Environment

A desirable architecture would unify the control planes of cloud and telecommunication providers to enable the visualized SFC capabilities. Principles and prototypes for such a future architecture is being developed by quite a few research groups and consortia nowadays.

In order to reveal the possibilities of the revisited approach of SFC for end-to-end service delivery and management, Internet Engineering Task Force (IETF) has created a working group (SFC WG). The goal is to define the problem statement,

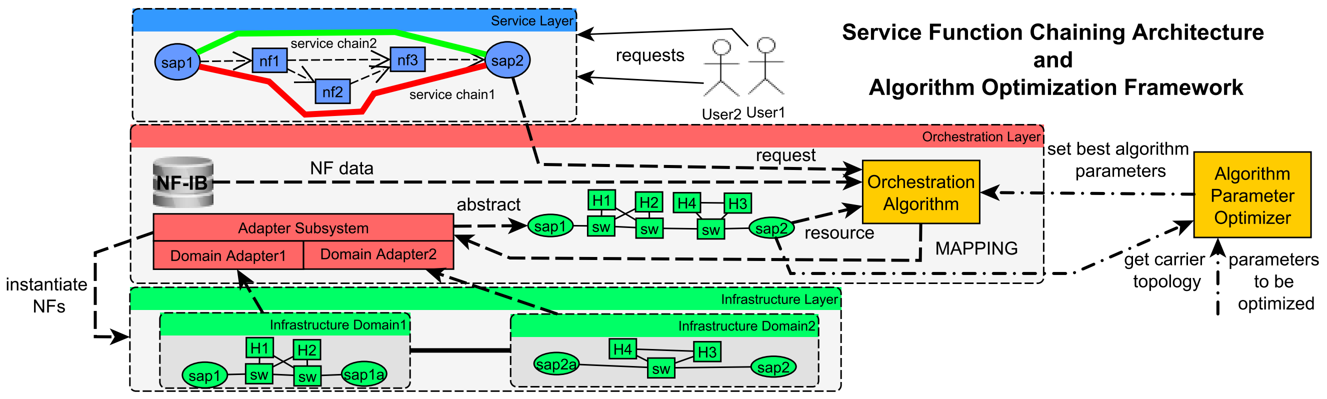

Fig. 1. A possible layered SFC architecture extended with the orchestration Algorithm Optimization Framework.

determine the core use cases and propose a possible realization of the SFC architecture [4].

The goal of the UNIFY project is to create a common service provision architecture for cloud and telecommunication providers. The UNIFY architecture, depicted in Fig. 1, has three main layers. The Infrastructure layer (denoted by color green) is responsible for the underlying, multi-domain, multi-layered network infrastructure. The red Orchestration layer takes care of efficient orchestration and manages the whole system. The new demands can be requested through the Service layer, where the given service chains are processed to the Orchestration layer.

A prototype implementation of the UNIFY architecture is ESCAPEv2, mentioned in Sec. I, which enables recursive resource orchestration[1]; interconnects different technological domains by adapters; is extensible with orchestration algorithms; operates on an abstract resource and service description model. The core of the framework is implemented on top of POX (OpenFlow controller) platform. It supports several infrastructure domains, such as Mininet (extended with the notion of NF and execution environment), legacy cloud domains managed by OpenStack and the novel *Universal Node* with hardware accelerated packet processing defined by the UNIFY project.

ESCAPEv2 framework eases the implementation of service chaining architectures and enables its users to experiment with resource orchestration algorithms. It offers an environment close to an envisioned real one based on NF virtualization, where services can be deployed to multiple domains. The novel Network Function Forwarding Graph (NFFG) model is implemented for the joint resource abstraction of network and compute resources, and also for service chain requests. Furthermore, it offers a well defined interface in order to easily replace and evaluate several mapping algorithms (denoted by Orchestration Algorithm box in Fig. 1).

III. The Mapping Algorithm

A. *Preference value based greedy backtrack algorithm*

We have designed and implemented a fast and efficient resource orchestration algorithm in order to handle the full mapping of enormous amount of service graphs arriving within few seconds to provider networks in the scale of several tens-to-hundreds of thousands of substrate nodes. It is a preference value based greedy backtrack algorithm based on graph pattern matching and bounded graph simulation. We have added this algorithm as a new mapping strategy to ESCAPEv2.

Besides the physical nodes and NFs, the substrate and the service graph have Service Access Points (SAPs), where the user connects to the network. These endpoints can be mapped unambiguously between the two graphs. Service chains are expressed as end-to-end paths in the service graph (see Service Layer of Fig. 1), defining which NFs should a specific subset of network traffic traverse between the appropriate SAPs, meanwhile the given requirements should hold on the path.

A simplified pseudo code of the algorithm can be found in Alg. 1. Line 8 in the pseudo code basically says that the service graph is partitioned into small paths derived from the end-to-end service chain requirements, so that, the subchains are disjoint on the edges of the service graph. This aids the greedy mapping process to find an appropriate host for NFs which are used by multiple service chains. In line 13, the variable k indicates the branching factor of the backtrack process.

The substrate nodes are ordered by a composite preference value function, which takes notice of (i) the weighted sum of the substrate node's utilizations of the resource components, (ii) the path length measured in latency and (iii) the average link utilization on the path leading to the node. Furthermore, the importance among the three components is also an adjustable parameter. This preference value defines the best candidates among the substrate nodes, where nf could be mapped.

The placement criteria in line 15 is determined by the NF sharing between end-to-end chains. In line 18, the algorithm subtracts the resources from the substrate graph reserved for the allocation of nf. If nf could not be mapped and backtrack is required, the resource reservation is undone in line 22.

B. *Parameter fine-tuning survey*

The previously presented algorithm has an abundance of adjustable parameters, which cannot be set without correct examination of the quality of service requirements of the service graphs, and their correlation to the resources of the substrate

[1]Infrastructure Domains on Fig. 1 can have inner orchestrators

Algorithm 1 Resource Orchestration Algorithm

Input: *substrate graph (V, E) with available resources, bandwidths, latencies; service graph (V_p, E_p) with end-to-end service chains (SC), and its quality of service requirements.*
Output: *mapping of NFs to substrate nodes, mapping of logical NF connections to substrate paths.*

```
1:  procedure MAP($(V, E), (V_p, E_p), (SC)$)
2:      Basic preprocessing on request and substrate graph.
3:      for all $SAP_p \in V_p$ do
4:          Find $SAP$ from $V$ for $SAP_p$.
5:      end for
6:      Find helper subgraphs for each end-to-end chain in the substrate graph.
7:      for all $c \in SC$ do
8:          Divide $c$ into subchains $(subc)$ so every edge in $E_p$ is exactly in one subchain.
9:          Add $subc$ to $sSC$
10:     end for
11:     for all $subc \in sSC$ do
12:         for all $nf \in subc$ do
13:             Push $k$ best substrate nodes and paths to $backtrack(nf)$ stack.
14:             if $nf \in$ other subchains then
15:                 Set placement criteria for $nf$.
16:             end if
17:             Map $nf$ to best substrate node and path leading to it.
18:             Update graph resources.
19:         end for
20:         if $nf$ could not be mapped anywhere then
21:             Pop substrate node and path from $backtrack(nf)$ stack.
22:             Redo graph and path resources.
23:             Try allocating $nf$ again.
24:         end if
25:         Map last unmapped edge of $subc$.
26:     end for
27:     return Complete mapping
28: end procedure
```

graph. So a right parameter setting is highly dependent on the environment where the algorithm is to be applied.

How should the utilization of a substrate node's resource component (e.g. memory) be valued? What weighting should be used between the resource components? How should a substrate path leading to a candidate host node be selected? What should be the relative importance of the components of the substrate node preference value? What branching factor and depth should the backtrack process have?

These questions must be answered so that the acceptance ratio and possibly the quality of orchestration of the algorithm would be maximized (see Algorithm Parameter Optimizer (APO) in Fig. 1). The problem with the assessment of the mapping quality is that the optimal solution for an input is unknown due to the complexity of the problem. Acceptance ratio is tested with benchmark service graph requests, which is much easier to define.

As a part of the Algorithm Optimization Framework, an arbitrary, large scale service provider network can be defined with all of its resource parameters, where the fine-tuned algorithm will be applied. Using this target substrate network, a combinatorial optimization process[2] in the parameter space defined by a set of the mentioned tunable algorithm parameters is conducted to achieve the desired acceptance ratio and quality of orchestration of the algorithm. This method is summarized by the inputs and output of APO in Fig.1.

During the demo, we showcase two example setups of the algorithm: (i) Deployable service graph requests in the ESCAPEv2 framework, where a small substrate network is simulated by Mininet. Arbitrary service request can be given to ESCAPEv2 via a graphical interface and any parameter of the algorithm can be adjusted to demonstrate their effect on the mapping results. (ii) An all-round abstract description of a carrier grade network with several tens-to-hundreds of thousands of substrate nodes connected into a real world topology, where the already fine-tuned[3] resource orchestration algorithm can be executed with arbitrary service request. The results of the mapping can be examined on visualizations in both cases.

Furthermore, a slide show will be presented to explain the algorithm demonstration setups and to give an idea about the mapping process and the parameter fine-tuning survey.

ESCAPEv2 and the orchestration simulation to the carrier grade network will run on a notebook, which will be taken to the demo by the presenters. A projector or a large screen is required for the slides and the result visualizations. Approximately 40-60 minutes and a table are required for setting up the demo.

Acknowledgements - This work was conducted within the framework of the FP7 UNIFY project, which is partially funded by the Commission of the European Union.

References

[1] N. M. Mosharaf Kabir Chowdhury, Muntasir Raihan Rahman, Raouf Boutaba, *Virtual Network Embedding with Coordinated Node and Link Mapping*, in *IEEE INFOCOM 2009*

[2] Carlo Furst, Stefan Schmid, Anja Feldmann, *Virtual network embedding with collocation: Benefits and limitations of pre-clustering*, in *Cloud Networking (CloudNet), IEEE 2nd International Conference*, San Francisco, CA, USA, 2013.

[3] Balázs Sonkoly, János Czentye, Robert Szabó, Dávid Jocha, János Elek, Sahel Sahhaf, Wouter Tavernier, Fulvio Risso, *Multi-Domain Service Orchestration Over Networks and Clouds: A Unified Approach*, In *Proc. of ACM SIGCOMM 2015*

[4] Jim Guichard, Thomas Narten, Alia Atlas, *Service Function Chaining - Charter for Working Group*, https://datatracker.ietf.org/wg/sfc/charter/, accessed: 13.08.2015.

[2]MIP cannot be applied because the objective function which maps vectors from the algorithm parameter space to the scalar numbers (e.g. acceptance ratio) is nor linear.

[3]The fine-tuning process takes too much time to demonstrate its operation at the time and place of the demo.