Virtual Network Embedding with Formal Reachability Assurance

Guido Marchetto*, Riccardo Sisto*, Jalolliddin Yusupov*, Adlen Ksentini†
* Politecnico di Torino
Email: {name.surname}@polito.it
† Eurecom
Email: adlen.ksentini@eurecom.fr

***Abstract*—Networks are becoming increasingly software-defined and automated. In this context, SDN and NFV allow service providers to use the network infrastructure more efficiently with reduced cost and to develop secure services. This procedure of efficient mapping of virtual networks on the substrate network is delegated to an orchestrator component of NFV that automatically manages its constituent virtualized network functions (VNFs). However, incomplete or inconsistent configuration of VNFs and service graphs may be vulnerable to potential security threats and could cause breakdown of services and of the supporting infrastructure. The main purpose of this paper is to provide an approach for allocation and formal verification that can ensure at the same time that policies such as reachability or isolation are never violated and that optimization is achieved. This ability to orchestrate and automate service validation makes assurance of reliable service delivery possible and simplifies security management tasks for network administrators.**

I. Introduction

Many platforms have recently emerged for NFV management and orchestration (MANO). These are capable of setting up and configuring service chains on demand, mapping Virtual Network Functions (VNF) onto physical resources, and steering traffic according to chaining policies. However, wrong configuration of VNFs and service graphs (SGs, aka service chains) could cause service degradation and security issues or even the breakdown of the supporting infrastructure. Moreover, automation of the NFV process with integration of Software-Defined Networking (SDN) technologies may lead the network services to be even more error-prone.

This renewed scenario poses new challenges, because the output of management and mapping service chain components needs to be carefully verified in order to ensure network correctness, security and fault tolerance. With this respect, mechanized formal methods have proven to be powerful engines for a formal verification of the network behavior in many different contexts [1], [2], [3], [4], [5], but they all refer to the service graph or low level configuration correctness. For example, in our previous work [1] we modeled the network, the forwarding behavior of VNFs, taking configurations into account, and reachability policies, as sets of First Order Logic (FOL) formulas. These sets of formulas are fed as input to a Satisfiability Modulo Theories (SMT) solver, z3[6], to verify their satisfiability, i.e. to verify the reachability requirements are correctly satisfied by the network model.

To the best of our knowledge, a verified orchestration of network services, giving at the same time optimal placement and assurance about a number of safety and security-related properties of the orchestrated virtual networks, remains an open problem to be addressed by the research community. For example, optimal placement alone does not provide assurance that reachability properties requested by a user are valid. Only after the deployment, a reachability analysis can be performed by means of tools such as ”ping” or ”traceroute”. The integration of formal verification with the placement procedure would allow these analyses to be automated and performed before deployment, thus formally verifying that services work before their actual deployment.

To cover this gap, this paper presents an approach that merges the two operations in one step. In our solution, we generate an optimal placement plan on the basis of given performance parameters (e.g., CPU cycles, physical location, latency, bandwidth, etc.) and deliver a formal assurance of reachability policies (e.g. isolation) within the requested performance constraints (e.g., sufficient bandwidth is available and latency is within the recommended values). Overall, this method makes it safer and more reliable for network operators to compose their network services in an NFV environment, so that they can be assured that the properties are correctly enforced. This is not possible with the previously mentioned verification tools (e.g. [2], [1]), which offer the possibility to verify network functionality but without keeping performance indicators into account. We recently introduced the formulation of the joint virtual network embedding and formal verification problem in [7], where we also presented some preliminary results related to an IIoT use case. Here we generalize the problem to a more generic network scenario by considering a larger set of supported VNFs and presenting more experimental results related to real network topologies different from the IIoT use case. Also, we discuss the advantages of the proposed approach over the traditional techniques generally used for the placement problem.

The remainder of this paper is structured as follows. We first introduce the main concepts of formal techniques in Section II to provide the background information for our work. Section III presents related work and in Section IV we describe the proposed solution via an example. The obtained results are presented in Section V, while Section VI concludes the paper.

978-3-903176-14-0

II. Background

Verigraph [1] is a network reachability verifier that models the forwarding behavior of a virtual network, possibly including stateful middleboxes, in a formal way. Verigraph can verify reachability properties, such as the possibility or impossibility for a certain flow of packets originated from a certain VNF of the network to reach a certain destination. Verigraph models the network and its functions as a set of logical formulas and reduces the verification of reachability properties to a SMT problem. Our work is based on the same approach used by Verigraph. However, while Verigraph exploits an SMT solver (z3), we exploit the possibilities offered by z3Opt [8] which can solve the Maximum Satisfiability (MaxSAT) problem.

Maximum satisfiability problem: The MaxSAT problem is the optimization version of the satisfiability (SAT) problem. The goal is to state and solve optimization objectives in the context of logical constraints by maximizing the weight of satisfied clauses in a SAT formula. The clauses can be divided into hard and soft clauses, depending on whether they must be satisfied (hard) or they may or may not be satisfied (soft). In our approach, reachability properties between nodes in the network that must be satisfied during the verification process are modeled as hard clauses, while the choices of how to allocate the VNFs of the SG onto the substrate nodes are modeled using soft clauses. It is possible to assign weights to soft clauses, which introduces differentiated penalties for falsifying them. The MaxSAT solver tries to find an assignment that satisfies all the hard clauses, and ensures the sum of the weights of the falsified clauses is minimal.

III. Related work

The classical literature on Virtual Network Embedding (VNE) is based on Integer Programming (IP) formulation of mapping each VNF to specific nodes and links in the substrate network, but it does not take into consideration reachability analysis during optimization. A number of NFV placement or orchestration frameworks have been studied in the literature. Some approaches, such as PACE [9], propose smart VM placement to deploy VNFs without considering reachability properties at all, so they cannot optimize or control the way packets are forwarded. Other state-of-the art approaches, such as APPLE [10], also consider reachability policies while providing VNF placement, where policies describe the sequence of VNFs that each class of flows needs to traverse in order. However, these approaches only assure that traffic is forwarded by the SDN switches according to the policies while they do not provide formal assurance that reachability policies will really hold, because they do not use precise models of the forwarding behavior of middleboxes.

In contrast to traditional methods for solving VNE problem based on mathematical programming and heuristic methods, which are limited to a set of constraints over binary, integer, or real variables, the problem addressed in this paper is to allocate VNFs while also formally checking that the desired reachability policies will hold, considering formal models of the forwarding behavior of all the VNFs involved, including their configurations. Even though the transformation of propositional calculus statements into integer and mixed integer programs is possible [11], combinatorial encoding is impractical in most cases and we were often not able to generate MaxSAT encodings for many of the instances when using it. The reason is that the formulation we have adopted in this paper is the first-order logic - an extension of propositional logic that covers variables for individual objects, quantifiers, symbols for functions, and symbols for relations.

Table I
Summary of key notations

Symbols	Notations
$G^s = (N^s, L^s, A^s_V, A^s_L)$	substrate network
$G^s = (N^v, L^v, A^v_V, A^v_L)$	virtual network
N^s, E^s, L^s	set of substrate nodes/endpoints/links
N^v, E^v, L^v	set of VNFs to be allocated/endpoints/links
A^s_N, A^s_L	attributes of substrate nodes/links
A^v_N	attributes of virtual functions
$l^s_{j,k}$	link between substrate nodes indexed by j and k
$n^v_i \uparrow n^s_j$	VNF n^v_i is hosted on substrate node n^s_j
$x_{i,j}$	boolean variable, true if a virtual function x_i is mapped onto substrate node j
y_i	boolean variable, true if substrate node is in use
$Soft(c, w)$	clause c is a soft clause with weight w
$route(v^v_i, v^v_{adj}, l^s)$	true if the adjacent neighbor of v^v_i is v^v_{adj} and it is reached via link l^s

How to verify security holes in SGs is another important consideration. During our research, we observed a number of existing approaches ([12], [13]) on the problem of security-aware optimal VNE. These approaches make an assumption that each virtual network request has a set of security requirements and enumerate them in a virtual network. These requirements only comprise constraints on confidentiality levels of the substrate nodes and isolation of the resources. In terms of security services, authentication, data integrity, confidentiality, and replay protection should be provided [14]. On the other hand, several VNFs (e.g., NAT) can modify or update packet headers and payload. In these environments, it is difficult to protect the integrity of flows traversing such VNFs and reason about reachability properties without using precise behavioral models of VNFs.

IV. Proposed solution

Similar to previous works in [15], [16], [7], the substrate network is modeled as a weighted undirected graph and denoted by $G^s = (V^s, L^s, A^s_V, A^s_L)$. We illustrate our methodology with a simple example of a substrate network with two nodes, three endpoints, and five links, as shown in Figure 1. The notation used in our derivation is summarized in Table I (detailed description can be found in [7]). We model a virtual network service request as another weighted directed graph denoted $G^v = (V^v, L^v, A^v_V, A^v_L)$. In this paper we are considering only chains of VNFs, and we assume the ordering in the chain is given by the indexes, i.e. v^v_i is the ith VNF in the chain. Figure 1 shows an example of a service request graph with 3 VNFs and 2 endpoint VNFs.

Upon the arrival of a service request G^v, an orchestrator component has to decide how to optimally allocate the VNFs of G^v onto the substrate network nodes. This is known as the Virtual Network Embedding (VNE) problem. In our case, this problem is combined with the problem of verifying that a number of reachability properties are satisfied by the virtual network, with a given configuration of the VNFs.

The mapping of the endpoint VNFs is assumed to be already specified in the service request. In the example, e_0^v and e_4^v are assumed to be mapped onto endpoints e_0^s and e_4^s respectively. In Figure 1, c is a shorthand for the storage attribute while l is a shorthand for the latency attribute. The storage requirement is 10 for all VNFs, to be installed in Docker containers. The figure depicts the SDN paradigm where all physical nodes that host VNFs are connected to one of the SDN-enabled switches. The orchestrator obtains network information and delegates forwarding rules to a central controller. In our example, the chain of the service request is assumed to be composed of the following types of functions: a web client (e_0^v), a firewall (n_1^v), a NAT (n_2^v), a DPI (n_3^v), and a web server(e_4^v).

The joint VNE and verification problem is formulated as a set of clauses for an SMT solver as follows. We use an initial set of hard clauses representing the VNF forwarding models and the reachability properties we want to ensure. These clauses are the same ones used for formal verification by Verigraph. The reachability properties for our example can be that the HTTP packets with allowed payloads can reach e_4^v from e_0^v while the other packets cannot. If the current configurations of the involved VNFs in the chain don't satisfy the reachability property, the solver returns UNSAT (unsatisfiable) and it will not produce a placement plan. In our example, the firewall is configured to allow only HTTP packets from e_0^v to e_4^v while the DPI is configured to drop HTTP packets with certain contents in the payload to satisfy the reachability property. In presence of these configurations we check the reachability property between the endpoints and we obtain SAT (satisfiable) result with an optimal placement plan. Analogously, we can assert an isolation property between the endpoints (e.g., if we want to be assured that affected endpoints are isolated in the network).

Resource requirements: We assume that VNFs from the same service request can share the same substrate node, which is common in Data Center networks, e.g. in order to reduce latency. For our example, the sum of all storage required by VNFs allocated on a substrate node should be less than or equal to the storage available on that substrate node and expressed as:

$$10 * x_{11} + 10 * x_{21} + 10 * x_{31} \leq 20 * y_1$$
$$10 * x_{12} + 10 * x_{22} + 10 * x_{32} \leq 10 * y_2$$

In addition to these inequalities, we need to represent explicitly that each VNF can be mapped onto exactly one node. For our example, such constraints take the following

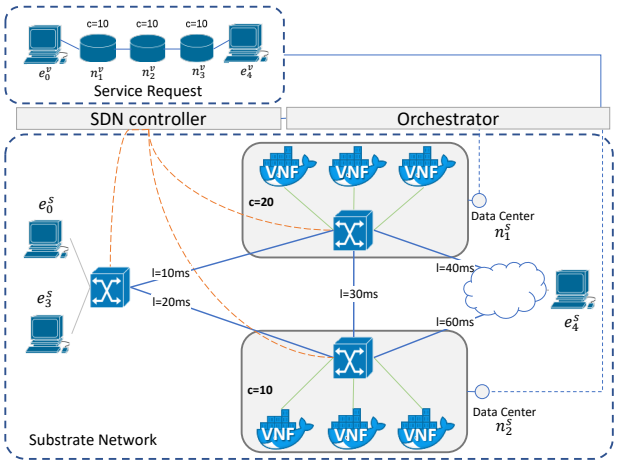

Figure 1. An example of VNE problem with verification process.

form:

$$x_{11} + x_{12} = 1 \quad x_{21} + x_{22} = 1 \quad x_{31} + x_{32} = 1$$

Finally, when substrate node is in use, there is at least one VNF deployed on this node and for our example we have:

$$y_1 \implies x_{11} \vee x_{21} \vee x_{31}$$
$$y_2 \implies x_{12} \vee x_{22} \vee x_{32}$$

Routing tables: The network behavior of the virtual service is modeled by a set of formulas that represent the routing tables of each network function involved in the service request. In order to see the formulas related to the forwarding behavior of network functions, readers are encouraged to see [7]. These formulas express the next hops - next gateways to which packets have to be forwarded along the path to their final destination. For each VNF v_i^v and next hop v_{i+1}^v, we define a predicate $route(v_i^v, v_{i+1}^v, l^s)$ which is true if the next hop of v_i^v is v_{i+1}^v and it is reached via link l^s.

The routing table of the first VNF e_0^v in the chain is formulated as a set of soft clauses, with the opposite of the link latency as the weight. In this way, the MaxSAT solver will minimize the overall latency of the chosen path in the infrastructure. As the location of e_0^v is fixed in the substrate endpoint e_0^s, we generate the following soft constraint for each possible substrate node n_k^s onto which n_1^v (the next VNF in the chain) can be allocated:

$$Soft((route(e_0^v, n_1^v, l_{0k}^s) \implies x_{1k}), -latency(l_{0k}^s)) \quad (1)$$

where the notation $Soft(c, w)$ specifies that clause c is a soft clause with weight w.

In practice, the routing table of the endpoint VNF specifies to which substrate node k a packet is forwarded depending on the allocation of the next VNF in the chain. For the example illustrated in Figure 1, which involves two endpoints in the substrate network, the following soft constraints are generated for the first endpoint of the chain:

$$\begin{aligned} &Soft((route(e_0^v, n_1^v, l_{01}^s) \implies x_{11}), -10) \\ &Soft((route(e_0^v, n_1^v, l_{02}^s) \implies x_{12}), -20) \end{aligned} \quad (2)$$

The soft clauses of the other VNFs $n_i^v \in N^v$ in the chain, with $i > 0$, are formulated similarly:

$$Soft((route(n_i^v, n_{i+1}^v, l_{jk}^s) \implies x_{ij} \wedge x_{(i+1)k}), \\ -latency(l_{jk}^s))$$

i.e., if VNF i forwards packets to the next VNF $i+1$ in the service graph through link l_{jk}, then the corresponding boolean variables x_{ij} and $x_{(i+1)k}$, which indicate the locations of the VNFs must be true. If two VNFs are allocated onto the same substrate node, i.e. $j = k$, we have $latency(l_{jk}^s) = 0$, and a soft clause with weight equal to zero is added to the set. For the example in Figure 1 the soft constraints are formulated as follows:

For n_1^v:

$$\begin{aligned} Soft((route(n_1^v, n_2^v, l_{11}^s) &\implies x_{11} \wedge x_{21}), 0) \\ Soft((route(n_1^v, n_2^v, l_{12}^s) &\implies x_{11} \wedge x_{22}), -30) \\ Soft((route(n_1^v, n_2^v, l_{21}^s) &\implies x_{12} \wedge x_{21}), -30) \\ Soft((route(n_1^v, n_2^v, l_{22}^s) &\implies x_{12} \wedge x_{22}), 0) \end{aligned} \tag{3}$$

For n_2^v:

$$\begin{aligned} Soft((route(n_2^v, n_3^v, l_{11}^s) &\implies x_{21} \wedge x_{31}), 0) \\ Soft((route(n_2^v, n_3^v, l_{12}^s) &\implies x_{21} \wedge x_{32}), -30) \\ Soft((route(n_2^v, n_3^v, l_{21}^s) &\implies x_{22} \wedge x_{31}), -30) \\ Soft((route(n_2^v, n_3^v, l_{22}^s) &\implies x_{22} \wedge x_{32}), 0) \end{aligned} \tag{4}$$

For n_3^v:

$$\begin{aligned} Soft((route(n_3^v, e_4^v, l_{14}^s) &\implies x_{31}), -40) \\ Soft((route(n_3^v, e_4^v, l_{24}^s) &\implies x_{32}), -60) \end{aligned} \tag{5}$$

where e_4^v represents the last VNF in the chain. As the location of the endpoint VNF is fixed in the substrate endpoint e_4^s, the implication is similar to Equation 2, where we reason only about the location of VNF n_3^v.

Optimization Objectives: VNE is a multi-objective optimization problem. From an infrastructure perspective, as many service requests as possible should be mapped onto the substrate network, making efficient use of the substrate network resources and minimizing link propagation delay (especially for communications that require low latency [17]). Accordingly, the objective function of our formulation has two goals: to minimize the number of substrate nodes in use and to minimize network latency. By feeding these objectives along with the formulas defined so far to the MaxSAT solver, we obtain, if possible, a model that satisfies all hard clauses, including the ones about reachability, while minimizing latency and the number of nodes in use. In the case of our example, the solver says the model is satisfiable with value *true* given to the following variables:

$$x_{1,2}, x_{2,1}, x_{3,1}, y_1, y_2$$

We can conclude from the output that the firewall VNF n_1^v in the chain is placed on the substrate node n_2^s, the NAT n_2^v and the DPI n_3^v on the substrate n_1^s. This allocation of the VNFs on the infrastructure introduces a link latency of 90 ms.

Table II
COMPUTATION TIME OF DIFFERENT TOPOLOGIES

Topology	Nodes	Links	Time (s)
Internet2[18]	12	15	0.551
GEANT[18]	23	74	17.674
UNIV1[19]	23	43	20.684
AS-3679[20]	79	147	31.454

V. Experimental Results

The joint VNE and verification approach based on MaxSAT presented in this paper has been evaluated by performing a number of experiments with real data sets. Each of these scenarios consists of a substrate network with a number of service requests to be allocated, with related reachability properties to be verified. All experiments have been executed on a workstation with 32GB RAM and an Intel i7-6700 CPU. As we can see from Table II, for small and medium topologies the placement and verification tool is fast. However, for the largest scenario where there are 79 hosts and 147 links, which involves 4 VNFs, the tool requires average of 31 seconds.

Algorithms solving the VNE problem come in two forms: offline algorithms and online algorithms. Online algorithms are better suited to deal with high dynamicity, which however comes at the cost of less optimal solutions, relying on heuristics. Moreover the online VNE problem is more difficult as we need to consider the arrival times of the requests and there are more possibilities of inefficient resource utilization due to time gaps created by earlier mappings. Our version allows to tackle the cases where the service requests are issued well ahead of the time when their service will be activated, thus allowing for sufficient time for offline planning. Taking into account that these calculations are performed with an exact method to obtain an optimal solution in offline mode, the computation time is acceptable. As the initial results show promises in smaller instances, we plan to improve our abstract model to cope with bigger instances and use them to further scale our tool.

VI. Conclusion

As todays networks are becoming more virtualized, it is important that network wide invariants are carefully verified before service deployment in production environments in order to detect and mitigate security attacks. This paper studied an approach for solving the VNE problem jointly with a formal analysis of reachability. It relies on the notion of maximum satisfiability and models the embedding problem as a set of clauses to be fed to a MaxSAT tool, along with other clauses that model the VNF forwarding behavior and the desired reachability policies. Instead of providing two different components for an orchestrator that performs allocation and formal verification of network wide invariants separately, our approach achieves this in one step. The results from our experiments show that the computational cost for providing formal assurance about reachability in addition to optimal embedding of virtual functions is adequate for offline modes of operation.

REFERENCES

[1] S. Spinoso, M. Virgilio, W. John, A. Manzalini, G. Marchetto, and R. Sisto, "Formal Verification of Virtual Network Function Graphs in an SP-DevOps Context," in *Service Oriented and Cloud Computing - 4th European Conference, ESOCC 2015, Taormina, Italy, September 15-17, 2015. Proceedings*, 2015, pp. 253–262.

[2] R. Stoenescu, M. Popovici, L. Negreanu, and C. Raiciu, "Scalable Symbolic Execution for Modern Networks," *Proceedings of the ACM SIGCOMM 2016 Conference*, pp. 314–327, 2016.

[3] N. P. Lopes, N. Bjørner, P. Godefroid, K. Jayaraman, and G. Varghese, "Checking beliefs in dynamic networks," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. Oakland, CA: USENIX Association, 2015, pp. 499–512.

[4] P. Kazemian, G. Varghese, and N. McKeown, "Header space analysis: Static checking for networks," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 9–9.

[5] S. K. Fayaz, T. Yu, Y. Tobioka, S. Chaki, and V. Sekar, "BUZZ: Testing Context-Dependent Policies in Stateful Networks," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. Santa Clara, CA: USENIX Association, 2016, pp. 275–289.

[6] L. De Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS'08/ETAPS'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 337–340.

[7] J. Y. Guido Marchetto, Riccardo Sisto and A. Ksentini, "Formally verified latency-aware vnf placement in industrial internet of things," in *14th IEEE International Workshop on Factory Communication Systems (WFCS), Imperia, Italy*, 2018, in press.

[8] N. Bjørner, A. Phan, and L. Fleckenstein, "νz - an optimizing SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, 2015, pp. 194–199.

[9] L. E. Li, V. Liaghat, H. Zhao, M. Hajiaghayi, D. Li, G. Wilfong, Y. R. Yang, and C. Guo, "Pace: Policy-aware application cloud embedding," in *2013 Proceedings IEEE INFOCOM*, April 2013, pp. 638–646.

[10] X. Li and C. Qian, "An nfv orchestration framework for interference-free policy enforcement," in *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, June 2016, pp. 649–658.

[11] E. Hadjiconstantinou, "Transformation of propositional calculus statements into integer and mixed integer programs: An approach towards automatic reformulation," *Brunel University Mathematics Technical Papers collection;*, 1990.

[12] S. Liu, Z. Cai, H. Xu, and M. Xu, "Towards security-aware virtual network embedding," *Computer Networks*, vol. 91, no. Supplement C, pp. 151 – 163, 2015.

[13] L. R. Bays, R. R. Oliveira, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Security-aware optimal resource allocation for virtual network embedding," in *2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualiztion management (svm)*, Oct 2012, pp. 378–384.

[14] B.-M. Chin, Y.-H. Choe, S.-U. Kim, and J.-I. Jung, "Automated test generation from specifications based on formal description techniques," *ETRI Journal*, vol. 19, no. 4, pp. 363–388, 1997.

[15] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, Mar. 2008.

[16] X. Cheng, S. Su, Z. Zhang, K. Shuang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology awareness and optimization," *Computer Networks*, vol. 56, no. 6, pp. 1797 – 1813, 2012.

[17] G. Chochlidakis and V. Friderikos, "Low latency virtual network embedding for mobile networks," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–6.

[18] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, April 2007, http://sndlib.zib.de, extended version accepted in Networks, 2009.

[19] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 267–280.

[20] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring isp topologies with rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, Feb. 2004.