Comparison of Virtualization Algorithms and Topologies for Data Center Networks

Hanene Ben Yedder, Qingye Ding, Umme Zakia, Zhida Li, Soroush Haeri, and Ljiljana Trajković
Simon Fraser University
Vancouver, British Columbia, Canada
Email: {hbenyedd, qingyed, uzakia, zhidal, shaeri, ljilja}@sfu.ca

***Abstract*—Data centers are the core infrastructure of cloud computing. Network virtualization in these centers is a promising solution that enables coexistence of multiple virtual networks on a shared infrastructure. It offers flexible management, lower implementation cost, higher network scalability, increased resource utilization, and improved energy efficiency. In this paper, we consider switch-centric data center network topologies and evaluate their use for network virtualization by comparing Deterministic (D-ViNE) and Randomized (R-ViNE) Virtual Network Embedding, Global Resource Capacity (GRC), and Global Resource Capacity-Multicommodity (GRC-M) Flow algorithms.**

I. Introduction

With the continuous increase of the Internet traffic and the migration of computations to the cloud, service providers rely on data centers as a cost-effective infrastructure for storing data and hosting large-scale service applications. Data center networks (DCNs) have become the core infrastructure of the Internet application providers such as Google, Amazon, and Facebook.

Efficient resource provisioning is a common challenge in deployment of cloud computing services because applications often have diverse resource requirements (storage, computing power, bandwidth, and latency) from the underlying infrastructure. Therefore, architecture of DCNs needs better flexibility to effectively support such applications. The current operational cost and utilization of DCNs as well as the delivered quality of service (QoS) are still inadequate [1], [2].

Network virtualization is a promising solution that enables coexistence of multiple virtual networks (VNs) on a shared physical substrate network while allowing each VN to be independently implemented and managed. Network virtualization offers lower cost, better scalability, and additional management flexibility while meeting QoS requirements. Accommodating a large number of virtual network requests (VNRs) requires an efficient mapping of VNs onto the substrate network resources. This problem, commonly known as virtual network embedding (VNE), has been the subject of recent research interest [3], [4], with emphasis on its application to DCNs [2]. Resource-provisioning problem such as network virtualization has a large solution space and high computational complexity.

Virtualization techniques for DCNs have been proposed to solve the problems of virtual machine (VM) placement and bandwidth allocation while satisfying topology constraints. Various algorithms have been proposed to effectively partition resources among requests with different requirements in order to achieve the optimal tradeoff between performance and operational cost. The Deterministic (D-ViNE) and Randomized (R-ViNE) Virtual Network Embedding [5], Global Resource Capacity (GRC) [3], and Global Resource Capacity-Multicommodity Flow (GRC-M) [6] are general purpose VNE algorithms. We compare their performance in terms of VNR acceptance ratio, revenue to cost ratio, and substrate node and link utilizations on a number of switch-centric data center topologies.

Comparison of switch-centric and server-centric architectures for virtual network embedding indicated that the switch-centric architecture exhibits better performance [4]. In this paper, we compare more recent switch-centric topologies that offer better performance and easier upgrades [7]. Simulations were performed using VNE-Sim, a discrete event simulator for virtual network embedding [8]. The data centers used by Amazon, Facebook, and Google have millions of hosts and servers [9]. Data center topologies simulated in this paper are much smaller than the deployed networks due to high memory requirements and long simulation times. However, large data centers possess similar structures as those considered in this paper.

The remainder of this paper is organized as follows. In Section II, we provide an overview of selected VNE algorithms. Architectures and characteristics of evaluated DCNs are described in Section III. In Section IV, performance metrics for evaluating the VNE performance are presented. Description of simulation environment and simulation results are given in Section V. We conclude with Section VI.

II. VNE Algorithms

VNE algorithms deal with resources allocation challenges for both virtual nodes and links. VNE resource allocation problems are NP-hard [5] and may be formulated as a mixed-integer program (MIP) or reduced to the multiway separator problem. MIPs have been one of the mainstream approaches to solve the VNE [5], [10], [11]. While these algorithms perform well, finding a near optimal solution for an MIP is often time consuming, which may become a bottleneck when the arrival rate of the virtual network requests is high. Node-ranking-based algorithms are among the recent approaches to solve the VNE [3], [12], [13]. This family of algorithms computes a score/rank for substrate and virtual nodes based on various heuristics. Then, using the computed rank, a mapping scheme [12] is employed to map the virtual nodes to substrate nodes. While these algorithms find solutions faster than MIP-based algorithms, the quality of the MIP-based solutions is often higher. Recently, Monte-Carlo tree search algorithm

has been employed for solving the VNE problem [14]. This approach allows adjustable execution time based on the traffic load and offers a balance between execution time and quality of the solution.

The VNE problem may be divided into two stages: Virtual Node Mapping (VNoM) and Virtual Link Mapping (VLiM). The D-ViNE, R-ViNE, GRC, and GRC-M algorithms are coordinated two-stage algorithms, where VNoM is solved in the first stage and the solution is then used for solving VLiM.

We consider the D-ViNE, R-ViNE, GRC, and GRC-M algorithms:

A. D-ViNE and *R-ViNE Algorithms* employ MIP and Multicommodity Flow (MCF) algorithms to solve VNE problems [5]. D-ViNE and R-ViNE apply MIP that deterministic and random rounding, respectively. VNoM is solved using both MIP and MCF. The MCF or the shortest path algorithm may be applied then for VLiM to address various requests for resources of the substrate network.

B. GRC Algorithm computes the available resource capacities of substrate nodes that could accept virtual requests [3]. It provides a comprehensive view of the overall resource capacity for each substrate graph because it computes both the capacity of a node and its neighbors. GRC solves VNoM first, followed by employing a slightly modified Dijkstra's shortest path algorithm for VLiM.

C. GRC-M Algorithm is a recently proposed algorithm [6] that combines GRC and MCF [15], [16] algorithms to solve VNE problems. GRC-M improves the profit by increasing acceptance ratio of VNRs. Similar to the GRC algorithm, GRC-M algorithm ranks the capacities of substrate nodes and embeds the virtual nodes according to "large to large and small to small" principle when solving VNoM. The MCF algorithm is used in the VLiM stage. MCF treats virtual link request, virtual node, and substrate node as commodity, source, and destination, respectively. Hence, it allows path splitting to fully utilize the substrate resources.

III. Topologies of Data Center Networks

Data centers have a variety of topologies consisting of switches and routers. They may be classified as enhanced, server-centric, or switch-centric topology.

Enhanced topologies may be implemented as pure optical, hybrid (electrical and optical), or wireless networks. Examples of architectures are: OSA, c-Through, and Wireless Fat-Tree [17], [18]. Optical networks offer high bandwidth and significant flexibility in reconfiguring topology during operations.

Server-centric DCNs use multi-level recursively defined structures [17]. This recursive design interconnects individual cells where each cell includes a single switch and a number of servers (hosts) [19]. Switches are not directly connected to each other and hosts perform packet forwarding functions. Servers with multiple network interface cards (NICs) are used as relay nodes and are responsible for routing while commodity switches are used for forwarding packets. An example is the BCube(n, k) DCN topology that consists of n levels and k hosts in level-0 connected to a k-port switch [4].

Most current data centers are based on switch-centric architectures [19]. Switch-centric DCNs rely on switches for interconnection and traffic routing while hosts provide the service. Switches are the dominant components while hosts include only one NIC. They preserve the layered approach of the conventional data centers and offer easy upgrades. In this paper, we consider the switch-centric DCNs.

A. Switch-Centric DCNs

Conventional switch-centric DCNs are based on two-tier or three-tier tree topologies and rely on enterprise switches that may have different number of ports. Recent DCNs employ more elaborate topologies such as Fat-Tree, F^{2}Tree, and Diamond. They rely on inexpensive commodity switches that have the same number of switch ports for interconnections and traffic routing. These topologies enhance traffic load balancing, fault-tolerance, and multi-path routing [17]. However, they suffer from increased complexity and limited scalability. The number of switches and hosts for various switch-centric DCN topologies are listed in Table I, where k represents the number of switch ports. We compare the following topologies:

TABLE I. Number of Switches and Hosts for Three DCN Topologies

	Fat-Tree	F^{2}Tree	Diamond
Core switches	$k^2/4$	$k^2/4 - k/2$	$k^2/4$
Aggregation switches	$k^2/2$	$k^2/2 - k$	$k^2/2$
Edge switches	$k^2/2$	$k^2/2 - 2k + 2$	$k^2/2$
Hosts	$k^3/4$	$k^3/4 - k^2 + k$	$k^3/4$
Links	$3k^3/4$	$7k^3/8 - 13k^2/4 + 3k$	$3k^3/4$

1) Two-Tier topology is shown Fig. 1. The generic top-down tree architecture has the core layer (layer-3) of switches (routers), the aggregation layer (layer-2) of switches, and the edge layer (layer-1) of top-of-rack (ToR) edge switches. The Two-Tier DCNs have only the core and edge layers because the aggregation and edge layers are merged. The core layer switches are connected via high-speed links and control incoming and outgoing DCN traffic. This layer is responsible for routing and balancing traffic load between the core and the aggregation layer. The aggregation layer supports various functions such as server-to-server traffic, default gateway redundancy, spanning tree processing, load balancing, and firewall. At the edge layer, each ToR switch is connected to two core switches for redundancy. Servers are connected via the edge switches.

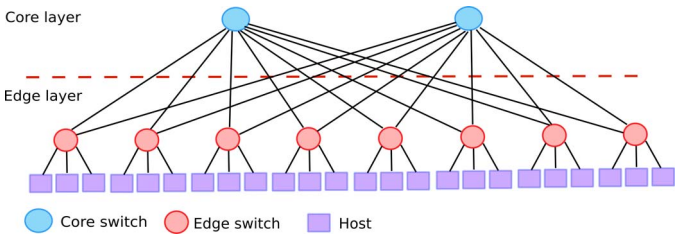

Fig. 1. An example of a Two-Tier DCN with only core and edge layers. It consists of 10 switches, 24 hosts, and 40 links.

2) F^{2}Tree topology is an enhancement of Fat-Tree [20], which is based on Clos architecture [21] and was proposed as an early enhancement for conventional DCNs. The Fat-Tree

topology is shown in Fig. 2. The F^{2}Tree topology has two additional links added to each core and aggregation switch thus forming rings in each pod [22], as shown in Fig. 3. It allows immediate access and backup route to the neighbors of the same pod while keeping all benefits of Fat-Tree. F^{2}Tree is designed to ensure fault-tolerance and to maximize bandwidth even though it has $(k^2 - k)$ fewer number of hosts than the Diamond topology.

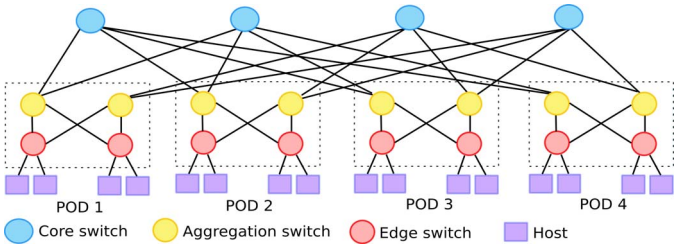

Fig. 2. An example of a Fat-Tree DCN with $k = 4$. It consists of 20 switches, 16 hosts, and 48 links.

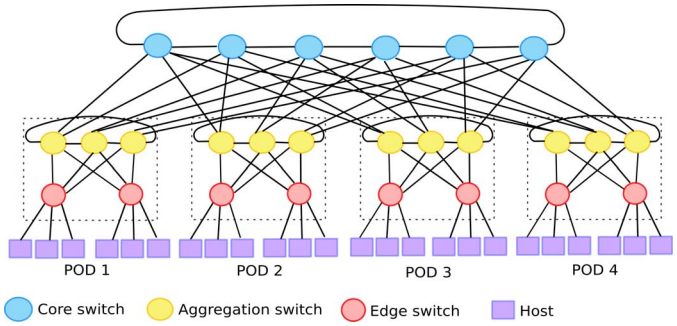

Fig. 3. An example of F^{2}Tree DCN with $k = 6$. It consists of 26 switches, 24 hosts, and 90 links.

3) Diamond topology shown in Fig. 4 is a variation of the Fat-Tree. It exhibits a symmetrical architecture where the core switches are placed in two layers [23]. There is no aggregation layer and the core switches are connected directly to the edge layer switches. Consequently, the average path length of inter-pod routes is shorter than in case of the Fat-Tree topology, which reduces the end-to-end delay.

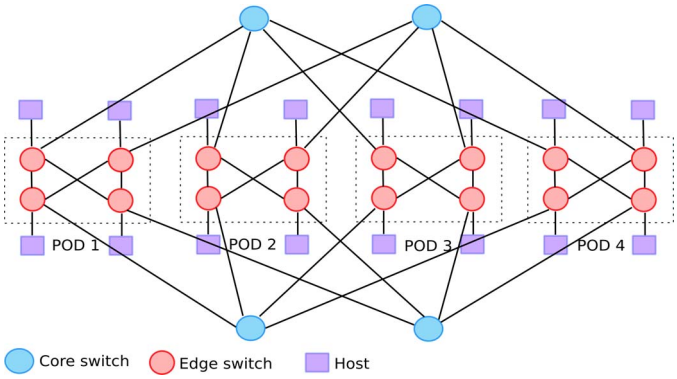

Fig. 4. An example of Diamond DCN with $k = 4$. It consists of 20 switches, 16 hosts, and 48 links.

B. Properties of Switch-centric DCNs

Performance of DCN topologies is compared based on metrics listed in Table II, where:

1) Load balancing addresses distribution of traffic through network paths.

2) Robustness (fault tolerance) is the capacity to handle processes in case of hardware failures.

3) Oversubscription ratio (OSR) is the ratio of the smallest aggregate bandwidth among the end hosts to the total bisection bandwidth [20]. Typical designs have OSR of 2.5:1 (400 Mbps) to 8:1 (125 Mbps). An OSR of 1:1 implies that all hosts may communicate with other hosts at the full bandwidth [7].

4) Scalability is the ability to increase the number of network elements. For Two-Tier topology, it is the number of edge switch ports (k_e) to the power of topology levels (2). For F^{2}Tree and Diamond topologies, the scalability is equal to the number of hosts [17].

5) Bisection bandwidth (BiW) is the bandwidth available between two network partitions of equal size. It is calculated by equally splitting the network into two parts and by counting the number of links crossing the dividing line [18].

The Two-Tier DCNs suffer from lack of robustness, limited bandwidth (oversubscription), poor scalability, low utilization, and high cost because of expensive enterprise-level switches [17]. Many recently proposed data center architectures have addressed these shortcomings and have offered performance enhancements. F^{2}Tree and Diamond topologies offer higher OSR, scalability, and bisection bandwidth compared to the Two-Tier topology. The F^{2}Tree topology exhibits higher load balancing and fault tolerance. It increases path redundancy by sacrificing some bisection bandwidth and scalability.

TABLE II. QUALITATIVE AND QUANTITATIVE COMPARISON OF THREE DCN TOPOLOGIES

	Load balancing; Robustness	**OSR**	**Scalability**	**BiW**
Two-Tier	Low	2.5:1–8:1	${k_e}^2$	$k_c/2$
F^{2}Tree	High	1:1	$k^3/4 - k^2 + k$	$(k^3/4 - k^2 + k)/2$
Diamond	Medium	1:1	$k^3/4$	$k^3/8$

IV. PERFORMANCE METRICS

We denote a substrate graph as $G^s(N^s, E^s)$, a substrate node as n_k^s, and a substrate link as e_k^s. CPU in a substrate node and bandwidth resources in a substrate link are represented as $\mathcal{C}(n^s)$ and $\mathcal{B}(e^s)$, respectively. $G^{\Psi_i}(N^{\Psi_i}, E^{\Psi_i})$ denotes a virtual graph, $n_k^{\Psi_i}$ a virtual node, and $e_k^{\Psi_i}$ a virtual link. CPU and bandwidth requirements of a virtual node and link are denoted as $\mathcal{C}(n^{\Psi_i})$and $\mathcal{B}(e^{\Psi_i})$, respectively.

Performance metrics used to evaluate the simulation results are acceptance ratio, revenue to cost ratio, average node utilization, and average link utilization [4]:

1) Acceptance ratio is the ratio of accepted requests among all requests:

$$p_a^\tau = \frac{|\Psi^a(\tau)|}{|\Psi(\tau)|}, \tag{1}$$

where $|\Psi^a(\tau)|$ and $|\Psi(\tau)|$ denote the number of accepted and arrived VNRs in a time slot τ, respectively.

2) Revenue and *cost* are calculated as:

$$\mathbf{R}(G^{\Psi_i}) = \omega_c \sum_{n^{\Psi_i} \in N^{\Psi_i}} \mathcal{C}(n^{\Psi_i}) + \omega_b \sum_{e^{\Psi_i} \in E^{\Psi_i}} \mathcal{B}(e^{\Psi_i}), \quad (2)$$

where ω_c and ω_b are the weights of CPU and bandwidth requests, respectively.

$$\mathbf{C}(G^{\Psi_i}) = \sum_{n^{\Psi_i} \in N^{\Psi_i}} \mathcal{C}(n^{\Psi_i}) + \sum_{e^{\Psi_i} \in E^{\Psi_i}} \sum_{e^s \in E^s} f_{e^s}^{e^{\Psi_i}}, \quad (3)$$

where $f_{e^s}^{e^{\Psi_i}}$ is the allocated bandwidth resource of a substrate link.

3) Node utilization is calculated as:

$$\mathcal{U}(N^s) = 1 - \frac{\sum_{n^s \in N^s} \mathcal{C}(n^s)}{\sum_{n^s \in N^s} \mathcal{C}_{max}(n^s)}, \quad (4)$$

where $\mathcal{C}_{max}(n^s)$ is the maximum available CPU in a substrate node.

4) Link utilization is calculated as:

$$\mathcal{U}(E^s) = 1 - \frac{\sum_{e^s \in E^s} \mathcal{B}(e^s)}{\sum_{e^s \in E^s} \mathcal{B}_{max}(e^s)}, \quad (5)$$

where $\mathcal{B}_{max}(e^s)$ is the maximum available bandwidth of a substrate link.

V. Simulation Scenarios and Results

VNE-Sim is a discrete event simulator for virtual network embedding [8]. It has been used for performance evaluation of a number of VNE algorithms [4], [6], [14]. It is publicly available under the terms of the MIT License. At its core, VNE-Sim contains classes and interfaces required for implementing various VNE algorithms. It also contains the simulation system that models the process of embedding virtual networks as a discrete event. VNE-Sim provides extensible interfaces for users to implement algorithms and customizable network elements. Written in C++, VNE-Sim offers good memory management and enables performing batch simulations using any scripting language.

We implement Two-Tier, F^{2}Tree, and Diamond data center topologies in VNE-Sim using the Fast Network Simulation Setup (FNSS) [24]. We generated Two-Tier and F^{2}Tree topologies with $k = 10$ and Diamond topology with $k = 8$. They have comparable number of hosts, switches, and links. Simulations were performed using a Dell Optiplex-790 with 16 GB memory and the Intel Core i7 2600 processor. The simulated substrate network topologies are listed in Table III.

The VNR requests are generated using the Boston University Representative Topology Generator [25]. Connections between the hosts are generated using the Waxman algorithm ($\alpha = 0.5$ and $\beta = 0.2$) [8], [26]. The number of virtual hosts is uniformly distributed between 3 and 10 [5] where each virtual host is connected to maximum three hosts. Initial substrate networks (SNs) resource capacities and VNR requests resource requirements are shown in Table IV. The VNR arrival rates are varied between 1 and 8 requests per 100 time units. Generated traffic loads range from 10 to 80 Erlangs [3]. Duration of each simulation scenario is 50,000 time units [3], [4], [6].

TABLE III. Simulated Substrate Network Topologies

Topology	No. of switches	No. of links	No. of hosts
Two-Tier	30	260	156
F^{2}Tree	92	580	160
Diamond	80	384	128

TABLE IV. Parameters of Substrate and Virtual Networks

SNs bandwidth	10 Gbps
Initial SNs host CPU capacity	100 units
Initial SNs link bandwidth	100 units
Virtual host CPU requirement	Uniformly distributed between 2 and 20 units
Virtual link bandwidth requirement	Uniformly distributed between 1 and 10 units
VN bandwidth	100 Mbps to 1 Gbps
VNRs arrival	Poisson distribution with mean arrival rate λ of requests per unit time
VNR life time	Exponentially distributed with a mean $1/\mu$ = 1,000
VNR traffic	λ/μ Erlangs

We compare D-ViNE, R-ViNE [5], GRC [3], and GRC-M [6] algorithms using four performance metrics: acceptance ratio, revenue to cost ratio, and substrate node and link utilizations, as shown in Table V. Performance of these algorithms depends on traffic loads and substrate network topologies. Based on the presented simulation results, the R-ViNE and GRC-M algorithms exhibit comparable behavior in terms of acceptance ratio and link and node utilizations. Thus, they may prove beneficial to the service providers. The D-ViNE algorithm exhibits inferior performance for low traffic loads. It offers comparable performance to the R-ViNE algorithm in case of smaller DCN topologies.

TABLE V. Performance of VNE Algorithms

DCN topology	Acceptance ratio	Revenue to cost ratio	Node utilization	Link utilization
Two-Tier	R-ViNE (max)	D-ViNE (max)	ViNE (max)	ViNE (max)
		GRC-M (min)		
F^{2}Tree		GRC (max)		
		D-ViNE (min)		
Diamond	D-ViNE (min)	D-ViNE (max)	GRC (min)	GRC (min)
		GRC-M (min)		

The GRC algorithm requires the lowest processing time for VNRs embeddings because it ranks resource capacities for substrate nodes. By permitting path splitting in the VLiM stage, the GRC-M algorithm achieves better embedding performance by trading some processing time. Both ViNE algorithms are based on MIP and, hence, require higher processing time while having better resources utilization. For all algorithms, Two-Tier and Diamond DCNs require half processing time of F^{2}Tree network, as shown in Fig. 5.

Simulation results shown in Fig. 6 illustrate that Two-Tier topology has the highest link utilization. F^{2}Tree topology exhibits higher acceptance and revenue to cost ratios compared to Two-Tier and Diamond topologies. It may accept additional VNRs since it provides multiple paths between hosts. F^{2}Tree and Diamond topologies show the lowest and the highest

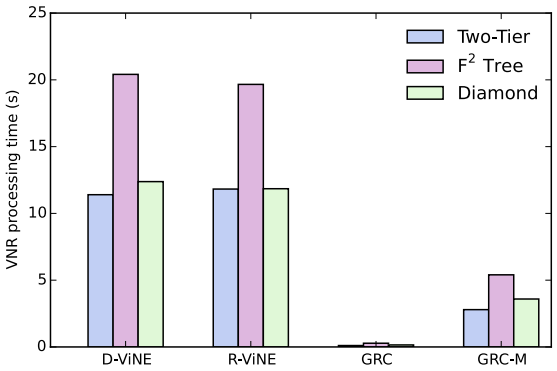

Fig. 5. Comparison of processing times for various VNE algorithms and DCN topologies.

node utilizations, respectively. Summary of simulation results is shown in Table VI.

TABLE VI. PERFORMANCE SUMMARY FOR SIMULATED TOPOLOGIES

Performance metrics	Two-Tier	F^{2}Tree	Diamond
Acceptance ratio		Highest	Lowest
Revenue to cost ratio	Highest		Lowest
Node utilization		Lowest	Highest
Link utilization	Highest	Lowest	

F^{2}Tree topology has higher wiring density and number of nodes and, hence, the embedding cost is higher while the link and node utilizations are lower. While Diamond topology has smaller number of nodes than F^{2}Tree, it still exhibits higher node utilization and has comparable performance. Two-Tier topology has the lowest switch to host ratio (0.192) compared to F^{2}Tree (0.575) and Diamond (0.625) topologies. It is also a preferred architecture for embedding in terms of revenue to cost ratio. However, Two-Tier topology has very low multi-path capabilities that may cause traffic congestion and delay. Furthermore, its performance degrades in case of hardware failure affecting the QoS of embedded VNRs.

VI. CONCLUSION

In this paper, we compare performance of various VNE algorithms in a number of switch-centric DCN topologies. The choice of the embedding algorithm depends on its performance and the topology of a data center. Simulation results indicate that GRC-M and R-ViNE algorithms exhibit better performance for embedding while GRC-M requires less processing time. The F^{2}Tree topology may accept additional VNRs, which is significant for virtualization while Diamond exhibits higher node utilization even with simpler topology and lower number of hosts.

REFERENCES

[1] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, and M. F. Zhani, "Data center network virtualization: a survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 909–928, Jan. 2013.

[2] J. Zhang, H. Huang, and X. Wang, "Resource provision algorithms in cloud computing: a survey," *J. Netw. Comput. Appl.*, vol. 64, pp. 23–42, Apr. 2016.

[3] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr. 2014, pp. 1–9.

[4] S. Haeri and Lj. Trajković, "Virtual network embeddings in data center networks," in *Proc. IEEE Int. Symp. Circuits and Systems*, Montreal, Canada, May 2016, pp. 874–877.

[5] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

[6] S. Haeri, Q. Ding, Z. Li, and Lj. Trajković, "Global resource capacity algorithm with path splitting for virtual network embedding," in *Proc. IEEE Int. Symp. Circuits and Systems*, Montreal, Canada, May 2016, pp. 666–669.

[7] (May 2017) Cisco Data Center Infrastructure 2.5 Design Guide. [Online]. Available: https://www.cisco.com/application/pdf/en/us/guest/netsol/ns107/c649/ccmigration_09186a008073377d.pdf.

[8] S. Haeri and Lj. Trajković, "VNE-Sim: a virtual network embedding simulator," in *Proc. SIMUTOOLS*, Prague, Czech Republic, Aug. 2016, pp. 112–117.

[9] (May 2017) Reports on data center networks. [Online]. Available: http://www.datacenterknowledge.com/.

[10] J. F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. de Meer, "Energy efficient virtual network embedding," *IEEE Commun. Lett.*, vol. 16, no. 5, pp. 756–759, May 2012.

[11] I. Houidi, W. Louati, W. Ben Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Comput. Netw.*, vol. 55, no. 4, pp. 1011–1023, Mar. 2011.

[12] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, Apr. 2011.

[13] S. Zhang, Y. Qian, J. Wu, and S. Lu, "An opportunistic resource sharing and topology- aware mapping framework for virtual networks," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 2408–2416.

[14] S. Haeri iand Lj. Trajković, "Virtual network embedding via Monte-Carlo tree search," *IEEE Trans. Cybern.*, vol. 47, no. 2, pp. 1–12, Feb. 2017.

[15] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener, "Provisioning a virtual private network: a network design problem for multi-commodity flow," in *Proc. ACM Symp. Theory of Comput.*, New York, NY, USA, July 2001, pp. 389–398.

[16] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *Comput. Commun. Rev.*, vol. 38, no. 2, pp. 19–29, Mar. 2008.

[17] T. Chen, X. Gao, and G. Chen, "The features, hardware, and architectures of data center networks: a survey," *J. Parallel Distrib. Comput.*, vol. 96, pp. 45–74, Oct. 2016.

[18] T. Wang, Z. Su, Y. Xia, and M. Hamdi, "Rethinking the data center networking: architecture, network protocols, and resource sharing," *IEEE Access*, vol. 2, pp. 1481–1496, Jan. 2014.

[19] A. Hammadi and L. Mhamdi, "A survey on architectures and energy efficiency in data center networks," *Comput. Commun.*, vol. 40, pp. 1–21, Mar. 2014.

[20] 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, Oct. 2008.

[21] C. E. Leiserson, "Fat-Trees: universal networks for hardware-efficient supercomputing," *IEEE Trans. Comput*, vol. 30, no. 10, pp. 892–901, Oct. 1985.

[22] G. Chen, Y. Zhao, D. Pei, and D. Li, "Rewiring 2 links is enough: Accelerating failure recovery in production data center networks," in *Proc. IEEE ICDCS*, Columbus, Ohio, USA, June 2015, pp. 569–578.

[23] Y. Sun, J. Chen, Q. Lu, and W. Fang, "Diamond: an improved Fat-Tree architecture for large-scale data centers," *J. Commun.*, vol. 9, no. 1, pp. 91–98, Jan. 2014.

[24] L. Saino, C. Cocora, and G. Pavlou, "A toolchain for simplifying network simulation setup," in *Proc. 6th Int. ICST Conf. Simulation Tools and Techn.*, Cannes, France, Mar. 2013, pp. 82–91.

[25] (May 2017) Boston University Representative Internet Topology Generator. [Online]. Available: http://www.cs.bu.edu/brite/.

[26] B. M. Waxman, "Routing of multipoint connections," *IEEE J. Sel. Areas Commun.*, vol. 6, no. 9, pp. 1617–1622, Dec. 1988.

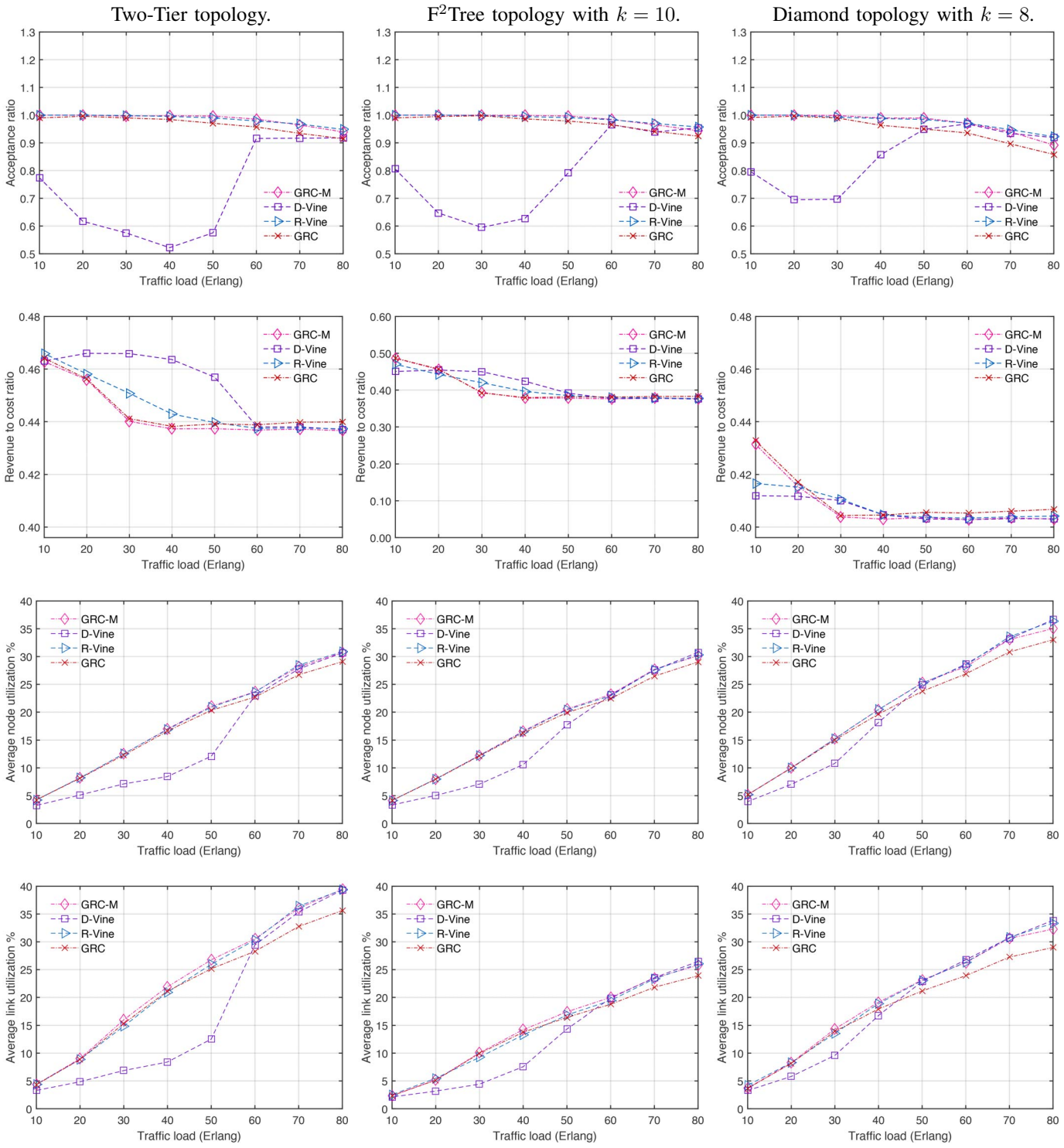

Fig. 6. Comparison of Two-Tier (left column), F^{2}Tree (middle column), and Diamond (right column) DCN topologies. Shown are acceptance ratio, revenue to cost ratio, and average node and link utilizations as functions of VNR traffic load.