Towards Reproducible Performance Studies Of Datacenter Network Architectures Using An Open-Source Simulation Approach

Daji Wong, Kiam Tian Seow
School of Computer Engineering
Nanyang Technological University
Singapore

Chuan Heng Foh
Center for Communication Systems Research
University of Surrey
Guildford, Surrey, United Kingdom

Renuga Kanagavelu
Data Storage Institute
A*STAR
Singapore

***Abstract*—In datacenter network (DCN) research, one key challenge is to reproduce the relative performances of different scalable DCN architectures in an unbiased and transparent way. Adequately addressing this challenge will support the validation of performance studies, and this is fundamental to building a strong research foundation and making wise datacenter investment decisions. In addressing this challenge, this paper presents the NTU-DSI-DCN initiative with a DCN simulation module based on an open-source platform called *ns-3*, on which the performance models of DCN topologies can be developed and made open source to support independent reproducibility of their performances. Advantages of the framework include the following: (1) it is low cost, (2) it provides transparent performance benchmarks of known DCN architectures and (3) it enables unbiased comparative performance simulations of DCN architectures, without the tedium of developing existing DCN models from scratch.**

In realizing this NTU-DSI-DCN initiative, the open-source performance models of the Fat tree and the BCube architectures have been implemented on the *ns-3* platform. A comparative performance study between Fat tree and BCube is reported, along with a performance reproducibility study of Fat tree. The documentation and source codes for our simulation set-ups are publicly available at http://code.google.com/p/ntu-dsi-dcn/. In continually adding new DCN architectural models or their variants in future work, the NTU-DSI-DCN is a promising initiative that can evolve into a well-documented open-source simulation platform supporting quality research in DCNs.

Keywords*—Datacenter network architectures, open-source simulation framework, reproducible performance studies

I. Introduction

A. Current trends in datacenter network (DCN) research

With the rise in demand for cloud computing services such as hypertextual web search engines [1], data processing platforms for large clusters [2] and social networking websites [3], a scalable datacenter is required to support these services [4]. Monetary costs become another important issue, as the estimated cost of around $125 million is required to realize a fully functional datacenter network of 50,000 servers [5], with approximately 45% of the costs going into the CPU, memory and storage systems, 25% into power distribution and cooling, 15% into electrical utility costs and another 15% into network equipment [6]. Motivated by economic considerations, several DCN architectures have been proposed to realize a scalable datacenter based on low-cost commodity hardware that can be easily purchased [7]–[14].

For a field to qualify as a science, it is imperative that published results are reproducible by others [15], [16]. However, this does not seem to be the standard practice in DCN research. Currently, existing performance results for various DCN architectures do not appear to be unbiased and transparent, as they seem to be produced from simulators mostly developed in a close-source or ad-hoc manner. Although some comparison work [17] has been made for the Fat tree [7], [18] and the DCell [12] architectures under an open-source platform called *ns-3* [19], these simulators are not readily available to the research community, and, in any case, are not implemented in a common open-source simulation environment to facilitate reproducibility of their experimental results. Investing in the necessary hardware to build a computer cluster just to manually reproduce the performance results would incur high monetary and administration costs. Moreover, to the best of our knowledge, there is little work in the literature on comparative performance studies of different DCN architectures for reference.

B. The need to reproduce performance studies of existing DCN architectures

Presented with different architectural solutions, there comes a need to evaluate them, either to help improve the architectures, to introduce an entirely new architecture, or to make wise business decisions. However, independently reproducing and ascertaining the performances of these solutions can be challenging. Reproducing the performance results of these architectural models for analysis is practically difficult as one would often have to develop the models from scratch, either on a software network simulator or a hardware testbed. An ideal approach is to have a common open-source simulation framework, to which researchers can contribute their performance models of new or existing DCN architectures, and from where they can find and readily use the DCN performance models contributed by others. A performance model (or simulator) consists of a datacenter network topology of interest, along with a configuration of its network devices and protocols and a traffic flow generator to facilitate performance evaluation.

C. Contributions

We feel that simulation studies should be done in an unbiased, transparent and cost-effective manner. In this direction, and in recognizing the fundamental importance of the Fat tree and the BCube architectures, this paper presents an open-source simulation module on DCN, using which a comparative performance study of the two DCN architectures is carried out. A performance reproducibility study of Fat tree investigated in [7], [17] is conducted as well. By this study, we mean to investigate the extent that the efficiency performance (measured in terms of throughput and delay) is consistent with (or deviant from) the results reported or claimed in the literature. The current module contains the Fat tree and BCube performance models developed on the open-source *ns-3* platform [19]. The module developed has been made publicly available at *http://code.google.com/p/ntu-dsi-dcn/*. Importantly, in a common open-source simulation framework, we demonstrate that the experimental results can be easily produced or reproduced by other researchers, which can in turn better serve as unbiased performance benchmarks against other variants or new topologies proposed in their future work.

We believe that this NTU-DSI-DCN initiative serves as a good starting point for researchers to reproduce the performance studies of two important DCN architectures for further understanding. It reduces the tedium of developing performance models and carrying out simulation work. Researchers can add performance models of new DCN architectures or their variants for comparative studies, which could in turn inspire improvements to existing work, leading to even higher quality research and datacenter investment decisions.

D. Paper organization

The rest of the paper is organized as follows. In Section II, we review the Fat tree and the BCube architectures. In Section III, we present a reproducible performance study of these two architectures. The experimental setup procedures and the simulation results are detailed. Section IV demonstrates and discusses the reproducibility of performance studies for the Fat tree architecture under the module. A discussion on related work is presented in Section V, before the paper is concluded in Section VI.

II. Fundamental DCN Architectures

A. Fat tree Architecture

A fundamental class of universal routing network called the Fat tree [18] allows for the interconnection of commodity switches and hosts. Several decades ago, Charles Clos designed a network topology called the Clos network, which ensures high bandwidth for end devices using smaller commodity switches that are interconnected together [20]. The Fat tree architecture adopted by *Al-Fares et al.* [7] can be seen as an interplay of these two fundamental concepts.

The Fat tree architecture of [7] can be seen as a hierarchical network topology consisting of four layers: The core switch layer, aggregation switch layer, edge switch layer and the end-hosts layer. In a k-ary Fat tree, there are $(k/2)^2$ k-port switches in the core switch layer. Below the core switch layer, there are k pods, enumerated from pod 0 to pod $k-1$. Each core switch has one port that is connected to each of the k pods. Within each pod, there is an aggregation switch layer and an edge layer, each containing $k/2$ switches. Each k-port switch in the edge layer is connected to $k/2$ end hosts. The remaining $k/2$ ports of each edge switch is connected to $k/2$ of the k ports of each aggregation switch. The number of end hosts that the architecture can support is $k^3/4$. The Fat tree architecture also incorporates several improvements to achieve better performance and fault-tolerance, such as the two-level table routing, flow scheduling, flow classification and *Bidirectional Forwarding Detection*. An illustration is depicted in Fig. 1, with $k = 4$.

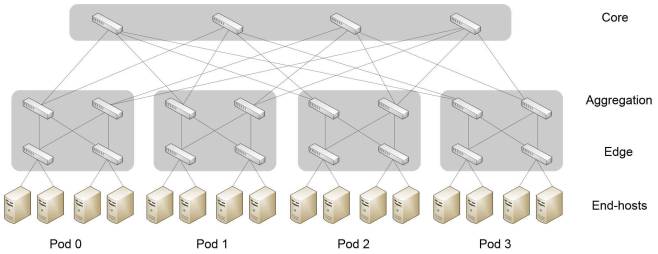

Fig. 1: A Fat tree DCN architecture with $k = 4$

B. BCube Architecture

In recent years, modular datacenter (MDC) has become a new approach to building datacenter [21] [22]. The BCube architecture serves as a suitable fundamental building block to realize a MDC design, which requires special features such as as higher network capacity and graceful performance degradation in the event of hardware failures.

The BCube architecture is a recursive structure in nature [8]. For example, a BCube_1 is constructed from n BCube_0 and n n-port switches. In a more general form, a BCube_k ($k \geq 1$) is constructed from n BCube_{k-1}s and n^k n-port switches. A BCube_0 contains n servers that are connected to an n-port switch. For each server in a BCube_k, it has $k+1$ ports, enumerated from level-0 to level-k. A BCube_k has $N = n^{k+1}$ servers and $k+1$ level of switches. Each level contains n^k n-port switches. The switches are denoted in the form of $<l, s_{k-1}s_{k-2}...s_0>$, where $l (0 \leq l \leq k)$ is the level of the switch and ($s_j \in [0, n-1]$, $j \in [0, k-1]$) allows the enumeration of a unique address for each switch. Each server in a BCube_k is enumerated using the addressing scheme as $a_k a_{k-1}...a_0 (a_i \in [0, n-1], i \in [0, k])$. To construct a BCube_k, we first enumerate the n BCube_{k-1}s from 0 to $n-1$. Next, we enumerate the servers in each BCube_{k-1} from 0 to $n^k - 1$, as the number of servers equals to n^k. To connect the switches to the servers, the level-k port of the i-th server ($i \in [0, n^k - 1]$) in the j-th BCube_{k-1} ($j \in [0, n-1]$) is connected to the j-th port of the i-th level-k switch. An illustration is given in Fig. 2, with $n = 4$ and $k = 1$.

III. A Reproducible Performance Study

A. Experimental Setup

Using a computer with Intel Core2 Duo 2.8GHz CPU and 4GB of RAM running CentOS 5.8, we have simulated the

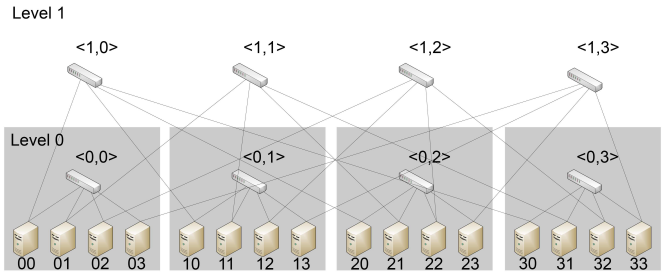

Fig. 2: A BCube DCN architecture with $n = 4$ and $k = 1$

performance models of the Fat tree and the BCube architectures on *ns-3* for comparison, which has been widely used by the research community for network simulation work [23] [24]. To the best of our knowledge, this is a new comparative performance study. Done under the NTU-DSI-DCN initiative, this study is reproducible and should serve as a useful benchmark. For a fair comparison, the size of both architectures, the configurations of the network devices and protocols, and the traffic flow generation have to be consistent. The only variable that should be changed is the network topology. Table I show the simulation settings used for the performance models of the Fat tree and the BCube architectures.

The size for both architectures ranges from 16 to 3,456 nodes. The 1,000 Mbps ethernet switches with realistic IP addresses and Nix-Vector routing [25] protocol are used. To generate traffic flow in them, we randomly select communication pairs across all the nodes, with each pair consisting of a sender and a receiver. Each pair would simultaneously send a 1 Mbps flow of data from its sender to its receiver and the simulation for both architectures are run for 100 seconds. The traffic flow pattern for both architectures follows an on-off behaviour with exponential random distribution, which has been shown to reasonably model traffic flows in real-world datacenters [26]. With the consistent simulation settings decided, we proceed to construct the Fat tree and the BCube network topology. The performance statistics are then gathered and analyzed using the Flow Monitor module [27] in *ns-3*.

For the performance study, we focus on two important performance metrics: Average packet delay and average throughput. The average packet delay can be calculated as follows:

$$D_{avg} = \frac{1}{n}\sum_{i=1}^{n} d_i \tag{1}$$

where D_{avg} refers to the average packet delay, n is the total number of packets received in the network and d_i is the delay of packet i.

The average throughput of the network can be calculated as follows:

$$\tau = \frac{\sum_{i=1}^{n}(p_i \times \delta_i)}{\sum_{i=1}^{n} d_i} \tag{2}$$

TABLE I: Simulation settings for the Fat tree and the BCube architectures

	Fat tree	BCube
Number of pods (k)	4-24	-
Number of BCube levels	-	3 ($k = 2$)
Number of nodes in BCube$_0$ (n)	-	4-15
Number of nodes	16-3,456	64-3,375
Simulation running time	100s	100s
Packet size	1024 bytes	1024 bytes
Data rate for packet sending	1 Mbps	1 Mbps
Data rate for device channel	1000 Mbps	1000 Mbps
Communication pairs selection	Random selection with uniform probability	Random selection with uniform probability
Traffic flow pattern	Exponential random traffic	Exponential random traffic
Routing protocol	Nix-Vector	Nix-Vector

where τ refers to the average throughput in the network, $p_i \in [0, 1]$ with $p_i = 0$ representing the loss of packet i and $p_i = 1$ representing the reception of packet i, δ_i as the size of packet i in bits, d_i as the delay of packet i and n is the total number of packets received in the network.

B. Experimental Results and Analysis

Based on the average throughput results in Fig. 3, we can observe that the BCube architecture consistently offers much more throughput performance than the Fat tree architecture, even as the number of nodes increases from 16 to 3,456. There is a slight degradation in the network throughput as the BCube architecture scales from 64 to 3,375 nodes, down from 237 Mbps to 174 Mbps. The Fat tree architecture network throughput remains steady within the range of 117 Mbps to 126 Mbps. For the average packet delay results in Fig. 4, we can see that the BCube architecture has slightly lower delay than the Fat tree architecture, with the BCube architecture having a range from 0.036 ms to 0.048 ms and the Fat tree architecture having a range from 0.066 ms to 0.072 ms. Both architectures experienced a fairly gradual increase in packet delay as the number of nodes increased from 64 to 3,456.

One of the reasons why BCube performs better in both metrics is due to its network architecture having multiple possible paths to send a traffic flow from point A to point B. This would bring about lesser traffic congestion and more available bandwidth. Moreover, servers in the BCube architecture act as relay nodes and help one another to speed up packet routing and traffic flow, thus resulting in improved throughput and packet delay performance over the Fat tree architecture. As for the Fat tree architecture, the reason for the consistent overall performance lies in the fundamental properties of Fat tree networks, which have been theoretically proven to be very efficient for interconnecting networks [18]. One can also argue that to a large extent, the packet delay performance of both architectures is independent of the size of the network. The results here have shown that the BCube architecture performs slightly better than the Fat tree architecture, in terms of average throughput and packet delay.

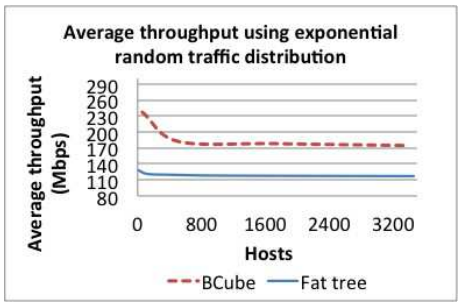

Fig. 3: Average throughput of the Fat tree and the BCube architectures

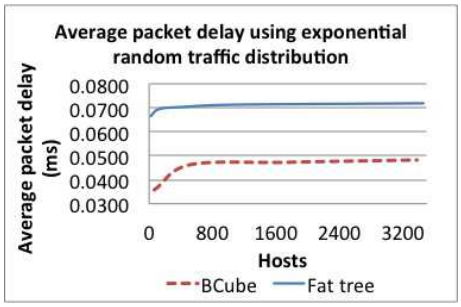

Fig. 4: Average packet delay of the Fat tree and the BCube architectures

IV. Towards Reproducible Performance Studies

A. Experimental Setup

We proceed to investigate the reproducibility of existing performance models in the literature through our simulation framework. As our simulation framework is in its infancy, we attempt to first reproduce the performance results of Fat tree that are available in [7], [17] for a start. In [17], the Fat tree performance model is simulated based on the *ns-3* platform and in [7], it is based on a hardware testbed. Table II shows the simulation settings as reportedly used in [7], [17].

Due to the close-source nature of the experiments in [17]. we are unable to accurately implement the two-level table routing protocol. Nix-Vector routing is used instead for the replicated performance model of [17]. For the replicated performance model of [7], we used 1,536 Mbps ethernet switches to model the ideal bisection bandwidth as specified by the authors. Various levels of k have been simulated to illustrate the scalability of the replicated performance models. The two-level table routing, flow classification and flow scheduling mechanisms have been omitted, as we could not reliably model their implementations which are close-source. Instead, we have substituted the two-level table routing with the Nix-Vector routing. For both replicated models, each host tries to send a traffic flow of 96 Mbps to its receivers and the strategy to select each communication pair follows a uniform random distribution. Apart from these variations, the simulation settings for the two replicated performance models follow what has been described by the authors.

TABLE II: Simulation settings used for the reproduction of the Fat tree performance models described in [17] and [7]

	Fat tree of [17]	Fat tree of [7]
Number of pods (k)	4-72	4, 16, 24
Number of nodes	16-93,312	16-3,456
Simulation running time	10-1,000s	100s
Packet size	1024 bytes	1024 bytes
Data rate for packet sending	1 Mbps	96 Mbps
Data rate for device channel	1000 Mbps	1536 Mbps
Communication pairs selection	Random selection with uniform probability	Random selection with uniform probability
Traffic flow pattern	Exponential random traffic	Exponential random traffic
Routing protocol	Nix-Vector	Nix-Vector

B. Experimental Evaluation

1) On Reproducing the Results in [17]: Fig. 5 shows the results of the replicated Fat tree performance model of up to 3,456 nodes for illustrative purposes, though it can generalize up to 93,312 nodes. The average throughput falls steadily from 127 Mbps to 117 Mbps and the average packet delay rises fairly from 0.066 ms to 0.071 ms, as the number of nodes increases from 16 to 3,456. Both results follow the same steady state trend with slight degradation as the number of nodes increases, and they are fairly consistent with the findings presented by the authors of [17]. There are some deviation in terms of the absolute performance. The reported average throughput and packet delay in [17] lie between 160 Mbps to 175 Mbps and between 0.040 ms to 0.050 ms respectively. The results of the replicated model show an approximate 25% drop in performance as compared to the results reported in [17]. One of the reasons might be due to the lack of the two-level table routing protocol in place, which has been shown empirically to increase the performance of the network [7]. Another reason could be due to the performance model differences at the implementation level.

With contributions from researchers in the community to extend and refine the performance models, we expect that the performance results stated by the authors of [17] would be reproducible to a large extent in the near term.

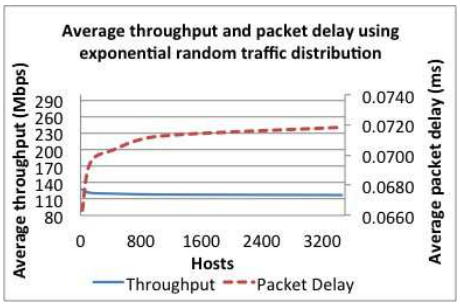

Fig. 5: Average throughput and packet delay of the Fat tree architecture using the simulation settings described in [17]

2) On Reproducing the Results in [7]: Table III shows the results of the replicated performance model as described above. While the network devices, protocols and applications used in the replicated model and the hardware testbed are

TABLE III: Results of the replicated Fat tree performance model described in [7]

Number of pods (k)	Hosts	Percentage of Ideal Bandwidth (1536 Mbps)	Average Packet Delay (ms)
4	16	12.4% (190 Mbps)	0.043
16	1024	11.5% (176 Mbps)	0.048
24	3456	11.2% (173 Mbps)	0.050

similar in behaviour qualitatively, for $k = 4$ with 16 nodes, we are only able to achieve 12.4% of the ideal bandwidth, as compared to 53.4% reported by the authors of [7]. The average packet delay is 0.043 ms. It seems that the results as reported in [7] are largely dependent on the absolute performance of the hardware testbed. However, without much investment into cabling and network equipment costs, we are able to gain some insights into the behaviour and conservative performance of Fat tree based on the simulation outcome. Without this study, one would have no idea of how close or far the simulation results are from the reported hardware experiments [7]. In any case, as we scale up the number of nodes to 3,456, the percentage value gradually drops to 11.2% of the ideal bandwidth, with the average packet delay increased to 0.050 ms. This observed downward trend is consistent with the findings presented in the previous sections.

Our common simulation framework is capable of reproducing the performance results obtained from the hardware testbed in [7] feasibly at low cost, with the caveat of a performance gap in relation to real hardware.

V. Related Work

While several suitable simulation platforms are available [28] [29] [30] [31] [32], *ns-3* [19] is best suited for building performance models for DCN architectures without reinventing the wheel as it is open-source, mature and actively being developed since 1999 [33]. With a common open-source simulation framework based on *ns-3*, the simulation results produced by the performance models of two fundamental DCN architectures, namely the Fat tree [7] and the BCube [8] architectures, can serve as important unbiased benchmarks against the existing DCN variants [9]–[14], [17], [34], [35] and new DCN architectures. We briefly survey these variants and explain the reasons behind the selection of the Fat tree and the BCube architectures as the focus of our research.

The VL2 [9] architecture extends the Fat tree topology with Valiant Load Balancing (VLB) to reduce congestion, and a directory-based cache routing to ensure quick route mapping lookups. Portland [11] is a set of layer 2 routing procedures that does efficient routing, address resolution and fault-tolerant detection on an arbitrary datacenter topology. However, these two solutions are not architectures by nature, as they are dependent on a fundamental network topology such as Fat tree in order to be fully functional.

MDCube [10] tries to interconnect BCube-based networks into a giant datacenter, with the main purposes of reducing the cabling requirements and costs. MDCube and Hyper-BCube [14] are essentially an extension of the BCube architecture, which has already been chosen as the one of the fundamental architectures to investigate. DCell [12], [17] is rather similar to the BCube architecture, in the sense that a higher-level DCell network can be built by connecting many low-level DCell networks together. The architecture of DCell seems to be more targeted towards building traditional large scale datacenters and the Fat tree architecture already exists as a decent fundamental architecture to adopt.

The GreenCloud [34] architecture uses a migration manager layer to dynamically allocate computing resources across each Virtual Machine (VM), thus delivering the needed performance and reduces power consumption. Secondnet [35] uses an algorithm to allocate virtual machines to physical machines in a way that guarantees the provision of CPU, memory and disk resources, ensuring scalability and high utilization in the network. These virtualized solutions require the support of non-commodity high-end grade servers which could inflate costs.

Monsoon [13] leverages on programmable commodity switches to introduce VLB to do layer 2 traffic load balancing and a new directory service for layer 3 routing and address mapping lookups. Unfortunately, this approach requires the use of programmable switches in place of off-the-shelves normal switches to inject the modifications into the architecture.

VI. Conclusion and Future Work

This paper has presented the NTU-DSI-DCN initiative of proposing a common *ns-3* simulation module for evaluating DCNs. We have illustrated that the reproducibility of simulation results for DCN architectures can be reasonably achieved through open-source simulation development on *ns-3*. The framework currently contains the performance models of two important DCN architectures, namely, Fat tree and BCube. A comparative performance study between Fat tree and BCube is reported, along with a performance reproducibility study of Fat tree. The simulation framework has been made publicly available at http://code.google.com/p/ntu-dsi-dcn/.

Simulation will always be a low cost method that does not displace but often precedes hardware experimentation, and *ns-3* is expected to evolve into a network simulator of high fidelity [23]. We hope this would encourage the research community to actively contribute different *ns-3* based performance models of DCN architectures to our module. On our part, future work includes adding more DCN architectures such as VL2, DCell and Portland into our module, as well as refining the Fat tree and BCube performance models.

VII. Acknowledgements

This research work was supported by the Singapore Agency for Science, Technology and Research, under SERC Future Data Center Technologies Programme Grant No: 112-172-0012. The authors would also like to thank Ngoc Linh Vu for implementing the Fat tree and BCube performance models, and performing the simulation experiments.

References

[1] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, vol. 30, no. 1, pp. 107–117, 1998.

[2] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[3] D. M. Boyd and N. B. Ellison, "Social network sites: Definition, history, and scholarship," *Journal of Computer-Mediated Communication*, vol. 13, no. 1, pp. 210–230, 2007.

[4] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[5] L. Popa, S. Ratnasamy, G. Iannaccone, A. Krishnamurthy, and I. Stoica, "A cost comparison of datacenter network architectures," in *Proceedings of the 6th International Conference, ACM*, 2010, p. 16.

[6] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.

[7] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, 2008, pp. 63–74.

[8] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: A high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63–74, 2009.

[9] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," in *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4. ACM, 2009, pp. 51–62.

[10] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "MDCube: A high performance network structure for modular data center interconnection," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, pp. 25–36.

[11] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "PortLand: A scalable fault-tolerant layer 2 data center network fabric," in *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4. ACM, 2009, pp. 39–50.

[12] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "DCell: A scalable and fault-tolerant network structure for data centers," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 75–86.

[13] A. Greenberg, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Towards a next generation data center architecture: Scalability and commoditization," in *Proceedings of the ACM workshop on Programmable Routers for Extensible Services of Tomorrow*. ACM, 2008, pp. 57–62.

[14] D. Lin, Y. Liu, M. Hamdi, and J. Muppala, "Hyper-bcube: A scalable data center network," *IEEE*, vol. 201, pp. 2918–2923, 2012.

[15] J. B. Buckheit and D. L. Donoho, *Wavelab and reproducible research*. Springer, 1995.

[16] P. Vandewalle, J. Kovacevic, and M. Vetterli, "Reproducible research in signal processing," *Signal Processing Magazine, IEEE*, vol. 26, no. 3, pp. 37–47, 2009.

[17] K. Bilal, S. U. Khan, J. Kolodziej, L. Zhang, K. Hayat, S. A. Madani, N. Min-Allah, L. Wang, and D. Chen, "A comparative study of data center network architectures," in *26th EUROPEAN Conference on Modelling and Simulation, ECMS*, 2012.

[18] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Transactions on Computers*, vol. 100, no. 10, pp. 892–901, 1985.

[19] ns-3, "A discrete-event network simulator," http://www.nsnam.org, accessed: 30 Jan 2013.

[20] C. Clos, "A study of non-blocking switching networks," *Bell System Technical Journal*, vol. 32, no. 2, pp. 406–424, 1953.

[21] J. Hamilton, "Architecture for modular data centers," *In 3rd CIDR*, 2007.

[22] M. M. Waldrop, "Data center in a box," *Scientific American*, vol. 297, no. 2, pp. 90–93, 2007.

[23] E. Weingartner, H. vom Lehn, and K. Wehrle, "A performance comparison of recent network simulators," in *IEEE International Conference on Communications, ICC'09*. IEEE, 2009, pp. 1–5.

[24] T. R. Henderson, M. Lacage, and G. F. Riley, "Network simulations with the ns-3 simulator," *SIGCOMM Demonstration*, 2008.

[25] Nix-Vector Routing, "Nix-Vector Routing API Documentation," http://www.nsnam.org/doxygen/group__nixvectorrouting.html, accessed: 30 Jan 2013.

[26] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 92–99, 2010.

[27] G. Carneiro, P. Fortuna, and M. Ricardo, "Flowmonitor: a network monitoring framework for the network simulator 3 (ns-3)," in *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, p. 1.

[28] J. R. Jump and S. Lakshmanamurthy, "Netsim: A general-purpose interconnection network simulator," in *Proceedings of the International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems*. Society for Computer Simulation International, 1993, pp. 121–125.

[29] X. Chang, "Network simulations with opnet," in *Simulation Conference Proceedings, 1999 Winter*, vol. 1. IEEE, 1999, pp. 307–314.

[30] A. Varga, "The omnet++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference (ESM2001)*, vol. 9, 2001.

[31] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, L. K. T. Lim, Saidi, A. G. Saidi, and S. K. Reinhardt, "The m5 simulator: Modeling networked systems," *Micro, IEEE*, vol. 26, no. 4, pp. 52–60, 2006.

[32] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 19.

[33] K. Fall, "Network emulation in the vint/ns simulator," in *Proceedings. IEEE International Symposium on Computers and Communications, 1999*. IEEE, 1999, pp. 244–250.

[34] L. Liu, H. Wang, X. Liu, X. Jin, W. He, Q. Wang, and Y. Chen, "GreenCloud: A new architecture for green data center," in *Proceedings of the 6th International Conference Industry Session on Autonomic Computing and Communications Industry Session*. ACM, 2009, pp. 29–38.

[35] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "Secondnet: A data center network virtualization architecture with bandwidth guarantees," in *Proceedings of the 6th International Conference*. ACM, 2010, p. 15.