A Reliability Analysis of Datacenter Topologies

Rodrigo S. Couto, Miguel Elias M. Campista, and Luís Henrique M. K. Costa
Universidade Federal do Rio de Janeiro - PEE/COPPE/GTA - DEL/POLI
Email:{souza,miguel,luish}@gta.ufrj.br

Abstract—**The network infrastructure plays an important role for datacenter applications. Therefore, datacenter network architectures are designed with three main goals: bandwidth, latency and reliability. This work focuses on the last goal and provides a comparative analysis of the topologies of prevalent datacenter architectures. Those architectures use a network based only on switches or a hybrid scheme of servers and switches to perform packet forwarding. We analyze failures of the main networking elements (link, server, and switch) to evaluate the tradeoffs of the different datacenter topologies. Considering only the network topology, our analysis provides a baseline study to the choice or design of a datacenter network with regard to reliability. Our results show that, as the number of failures increases, the considered hybrid topologies can substantially increase the path length, whereas servers on the switch-only topology tend to disconnect more quickly from the main network.**

I. Introduction

Currently, the time needed to complete an Internet transaction is becoming a competitive factor among companies offering online services, such as web search, home banking, and shopping. The typical solution to reduce the response time of these services is distributed processing (e.g., MapReduce). This strategy is more efficient if more servers in the datacenter execute the parts of a single task. As a consequence, the number of servers in datacenters is growing steadily fast. Google, for instance, has a computing infrastructure of almost 1 million servers spread in datacenters around the world [1].

Distributed processing incurs in communication between servers, which adds latency to the completion of a distributed task. Moreover, high communication between servers cause high link utilization, which may lead to buffer congestion in switches, adding to latency. As data transfer is a potential slowdown for datacenter operations, distributed programming models use locality properties to choose the most appropriate server to store data. Ideally, one would plan for limiting data transfers to servers in a single rack. However, choosing the best server to store a specific piece of data is a difficult task, especially if we consider the ever increasing number of servers in datacenter networks. Thus, significant effort has been devoted to the development of new datacenter architectures which improve networking performance, while keeping the economical aspect in mind. One of the earliest datacenter architecture is Fat-Tree [2], which focuses on the utilization of off-the-shelf switches to avoid high costs. BCube [3] and DCell [4] are examples of architectures that use a combination of servers and switches to perform packet forwarding. The server-based forwarding allows those architectures to use switches with lower port density than Fat-Tree. Each architecture uses specific topologies and routing protocols.

For datacenters, networking performance is a function of three main metrics: bandwidth, latency, and reliability. Despite the high available bandwidth achieved by these architectures, datacenters are composed of tens of thousands of servers, which are prone to failures as well as the networking elements [5]. On the other hand, the datacenter must remain operational and present minimal impact to the user experience. To date, few studies compare existent architectures considering failures on each one of the main networking elements, namely, servers, switches, and physical links. Popa *et al.* [6] compare the different architectures in terms of cost and energy consumption, considering similar configurations to yield compatible performance. By analyzing the network capacity and maximum latency, they conclude that hybrid topologies (e.g. BCube) are cheaper than switch-only topologies (e.g. Fat-Tree). However, they foresee that switch-only topologies will become more cost-effective with the appearance of very low-cost switches in a near future. Guo *et al.* [3] address the reliability of the different topologies for specific traffic patterns and protocols, concluding that BCube is the most reliable.

In this work, we analyze the network topologies of three of the main existent datacenter architectures (Fat-Tree, BCube, and DCell) in terms of reliability, adding to the cost and bandwidth comparisons found in the literature. The present reliability analysis does not depend on applications, routing algorithms, or traffic engineering strategies used by each architecture. Instead, it provides a baseline study by using metrics to quantify the reliability of the datacenter network. These metrics can be combined with cost and available bandwidth metrics to help the datacenter designer. For example, the framework proposed by Curtis *et. al.* [7] proposes a datacenter topology optimizing metrics as available bandwidth and latency. However, it can be improved by using our definition of reliability evaluated in this work. In our analysis, we model the datacenter topology as a graph with servers and switches as nodes with network links connecting them. Using this model, we evaluate the impact of each networking component (server, switch and link) failure to the entire network.

The results of our analysis show that the network degrades with the removal of connected components with a relatively small number of servers as the number of failures increases, for all considered topologies. We also show that hybrid topologies as BCube and DCell can substantially increase the average path length as the failures increase, whereas in Fat-Tree servers tend to disconnect more quickly.

This paper is organized as follows. Section II details the

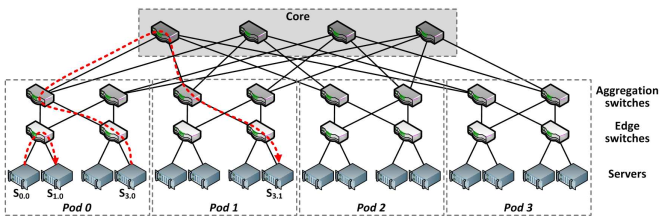


Fig. 1. Fat-Tree topology.

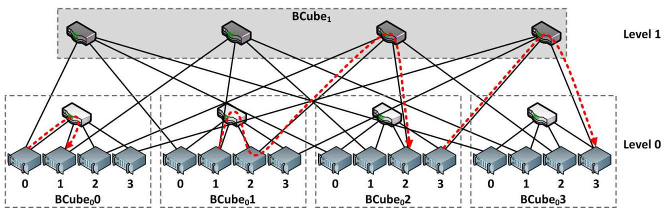


Fig. 2. BCube topology.

topology used in each architecture. Section III describes the methodology and the metrics used. Section IV shows the obtained results and Section V concludes this work.

II. Datacenter network topologies

In this work, we consider three representative datacenter topologies found on the current literature, Fat-Tree, BCube, and DCell. Their main characteristics are explained below.

A. Fat-Tree

We refer to Fat-Tree as the topology proposed by Al-Fares *et al.* in [2]. The authors use the concept of fat-tree, which is a special case of a Clos network, to define a datacenter topology organized as a k-ary tree. VL2 [8] also uses a Clos network and is not considered in our analysis due to its similarity to Fat-Tree. As shown in Figure 1 the topology has two sets of elements, the core and the pods. The first set is composed of switches that interconnect the pods. Each port of each switch in the core is connected to a different pod. A pod is composed of aggregation and edge switches, and datacenter servers. Aggregation switches connect the pod to the core by linking edge and core switches. Finally, each edge switch is connected to a different set of servers.

All switches are identical and have k ports. Consequently, the network has k pods, and each pod has $\frac{k}{2}$ aggregation switches and $\frac{k}{2}$ edge switches. In a single pod, each aggregation switch is connected to all edge switches, which are individually connected to $\frac{k}{2}$ different servers. Thus, the Fat-Tree topology can have $\frac{k}{2} * \frac{k}{2} * k = \frac{k^3}{4}$ servers. Figure 1 shows a Fat-Tree for $k = 4$. Note, as an example, that the server with index 0 in Pod 0 ($S_{0.0}$) communicates with Server 1 ($S_{1.0}$) in the same pod and both of them are connected through the same edge switch. On the other hand, Server 3 from Pod 0 ($S_{3.0}$) communicates with a server in a different pod, $S_{3.1}$, requiring the use of core switches. Fat-Tree allows all servers to communicate at the same time using the total capacity of their network interfaces. In this topology, all networking elements are identical, avoiding expensive switches with high port density in higher topology levels.

B. BCube

BCube [3] topology was proposed to be used in a Modular Data Center (MDC), which is a datacenter built inside shipping containers to permit simpler installation and physical migration procedures as compared with regular datacenters. Datacenter migration is useful for energy saving, because it becomes easier to move the datacenter to regions with lower energy costs, and for strategic positioning, allowing the placement close to regions with high service demands. As MDCs are built in sealed containers with a high equipment density, they need to be highly reliable. Furthermore, the performance of these networks has to slowly degrade as equipment failures occur. Also, as in the case of Fat-Tree, the network have a high transmission rate capacity and low cost. To this end, the BCube topology has layers of COTS (commodity off-the-shelf) mini-switches and servers, which participate in packet forwarding. These servers thus have several network interfaces, usually no more than five [3]. The main module of a BCube topology is $BCube_0$, which consists of a single switch with n ports connected to n servers. A $BCube_1$, on the other hand, is constructed using n $BCube_0$ networks and n switches. Each switch is connected to all $BCube_0$ networks through its connection with one server of each $BCube_0$. Figure 2 shows a $BCube_1$ network. More generally, a $BCube_k$ ($k \geq 1$) network consists of n $BCube_{k-1}$s and n^k switches of n ports. To build a $BCube_k$, the n $Bcube_{k-1}$s are numbered from 0 to $n-1$ and the servers of each one from 0 to $n^k - 1$. Next, the level k port of the i-th server ($i \in [0, n^k - 1]$) of the j-th $BCube_k$ ($j \in [0, n-1]$) is connected to the j-th port of the i-th level k switch. A $BCube_k$ network can have n^{k+1} servers. Figure 2 shows that, in $BCube_0 0$, Server 0 communicates through a switch to Server 1. On the other hand, Server 1 from $BCube_0 1$ uses its local switch to forward its packets to Server 2, which can forward the packet to the destination, in this case the Server 2 in $BCube_0 2$ network. However, the communication between different $BCube$s in the same level may occur by only using a higher level switch, as in the case of Server 3 of $BCube_0 2$ with Server 3 in $BCube_0 3$. In a nutshell, BCube servers can participate in packet forwarding depending on the communicating pair.

C. DCell

Similarly to BCube, DCell is defined recursively and uses servers and mini-switches for packet forwarding. The main module of this topology is $DCell_0$ which, as $BCube_0$, is composed of a switch connected to n servers. A $DCell_1$ is built by connecting $n + 1$ $DCell_0$ networks and a $DCell_0$ is connected to all other $DCell_0$ cells by one link from one of its servers to a server in another $DCell_0$. A $DCell_1$ network is illustrated in Figure 3. Note that communication inside a cell is performed locally using a switch, as shown in the communication between Server 2 and Server 3 from $DCell_0 0$. The communication between servers from different cells is performed directly, as the one between Server 1 in $DCell_0 2$

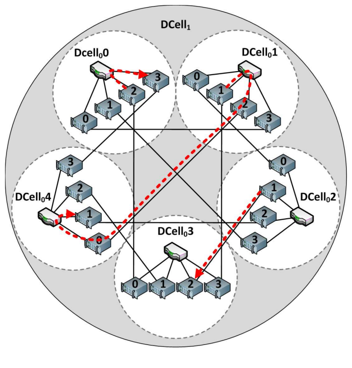


Fig. 3. DCell topology.

and Server 2 in $DCell_03$, or using a combination of servers and switches. This last is shown in the communication between Server 1 from $DCell_01$ and Server 1 from $DCell_04$.

Note that in a DCell, differently from BCube, the switches are connected only to servers in its same DCell and the connection between different DCell networks is always done using servers. To build a $DCell_k$, $n+1$ $DCell_{k-1}$ networks are needed. Each server in a $DCell_k$ has $k+1$ links. On each server the first link (level 0 link) is connected to the switch of its $DCell_0$ and the second link connects the server to a node on its $DCell_1$, but in another $DCell_0$. Generically, the level i link of a server connects it to a different $DCell_{i-1}$ in the same $DCell_i$. The procedure to construct a DCell is more complex than that of a BCube, and is executed by an algorithm proposed by Guo *et al.* [4].

The DCell capacity in number of servers can be evaluated recursively, using the following equations: $g_k = t_{k-1} + 1$ and $t_k = g_k \times t_{k-1}$, where g_k is the number of $DCell_{k-1}$ networks in a $DCell_k$ and t_k is the number of servers in a $DCell_k$. A $DCell_0$ network is a special case in which $g_0 = 1$ e $t_0 = n$.

III. Reliability Analysis

Our analysis considers failures in switches, servers, and links in a datacenter. To this end, a topology is modeled as an undirected graph $G = (V, E)$, where V is the set of vertices, modeling the nodes, and E is the set of edges, modeling the links. The set V is given by $V = S \cup C$, where S is the server set and C the switch set. Every edge weight in G is unitary because all topologies use only a single type of physical link. To analyze the reliability of graph G of each topology, we remove either S', C', or E' from G, where $S' \subset S$, $C' \subset C$ and $E' \subset E$, generating the subgraph G'. The removal of different elements from the network allows to analyze the influence of each failure type. The vertices or edges removed from G are randomly chosen from only one of the given sets (switches, servers, or links), generating the subgraph G'. The graphs were generated and analyzed using NetworkX [9].

The main goal of our work is to quantify how performance degrades when a fraction of malfunctioned network devices are not repaired. Also, as we consider only the topology, our analysis represent the network behavior on a steady-state, i.e., when the routing protocol has finished to compute all routes after a failure event. We evaluate the reliability in terms of the network size and the variation of path length when the datacenter network is subject to failures. The first one identifies the number of servers interconnected, whereas the second one is related to the number of hops between the network servers.

A. Metrics Related to Network Size

The metrics related to network size are the maximum and average relative sizes of the connected components after failures in network devices, using metrics similar to those used by Albert and Barabási [10], that provides a study about reliability in complex networks. The relative size of the largest connected component in relation to the number of existent servers, called RS_{max}, is given by Equation 1:

$$RS_{max} = \frac{\max_{1 \le i \le n} |s_i|}{\sum_{i=1}^{n} |s_i|}, \tag{1}$$

where $|s_i|$ is the number of servers in each connected component i, and n is the total number of connected components in the resulting graph G'. The second metric is the average size of isolated connected components of G', given by S_{avg} in Equation 2. In this work we define as isolated components all of the network partitions, except the largest component.

$$S_{avg} = \frac{1}{n-1}\left(\sum_{i=1}^{n} |s_i| - \max_{1 \le i \le n} |s_i|\right). \tag{2}$$

B. Metrics Related to Path Length

The first metric used for path length analysis takes into account the relationship between the diameter of the largest connected component of G', given by D'_{max}, and the diameter of the original graph G, given by D_{max}. This metric is defined as the diameter stretch, given by DS below:

$$DS = \frac{D'_{max}}{D_{max}}. \tag{3}$$

We also evaluate the path stretch. This metric quantifies the increase on the average path lenght between all server pairs in G', given by L'_{avg}, in relation to this average path length on G, given by L_{avg}. Thus, this metric is defined as:

$$PS = \frac{L'_{avg}}{L_{avg}}. \tag{4}$$

The stretch metrics consider only distances between servers, by analyzing the shortest path between each pair. The shortest paths is a good base measure to evaluate the behavior of path quality on the network, since it is used by most routing mechanisms that can be used in a data center, such as TRILL, IEEE 802.1aq and SPAIN [11]. Note that these mechanisms can also use multipath routing, which is generally based on the shortest paths. In this way, our future work includes the reliability analysis considering path diversity, which impacts the multipath routing schemes.

TABLE I
PROPERTIES OF THE ANALYZED TOPOLOGIES.

Name	Switch ports	Server ports	Servers	Switches	Diameter	Avg. length
Fat-Tree	24	1	3456	720	6	5.9
BCube2	58	2	3364	116	4	3.9
BCube3	15	3	3375	670	6	5.6
BCube5	5	5	3125	3125	10	8.0
DCell2	58	2	3422	59	5	4.9
DCell3	7	3	3192	456	11	8.2

IV. RESULTS

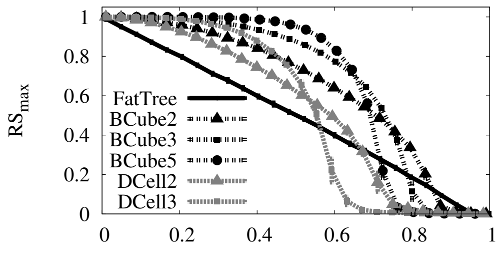
Our results are obtained using the topologies detailed in Section II, with some parameter variations to achieve a number of servers close to 3,400 for all topologies for fair comparison. On the other hand, we observed that this number is sufficiently large to disclose the differences between the topologies. As these topologies have a regular structure, our results can be extended to a higher number of servers. Table I shows the name associated to each considered topology and their respective parameters: the number of switch ports and server ports. Table I also gives the following properties of each topology: the number of servers, the number of switches, the network diameter, and the average path length between all pairs of servers. Using each topology of Table I, we average the outcomes when repeating the methodology of Section III several times, using a confidence level of 95%.

It is worth mentioning that although some of these topologies can be incrementally deployed, we only consider *complete* topologies where all network interfaces of servers and switches are used. Furthermore, the number of switch ports was not limited to the number of ports often seen in commercially available equipment, to provide similar number of servers for all topologies. As one of the key goals of the network topology of a datacenter is to provide processing capacity or storage redundancy, which increases with a higher number of servers, balancing the number of server per topology is an attempt to provide an analysis as fair as possible.

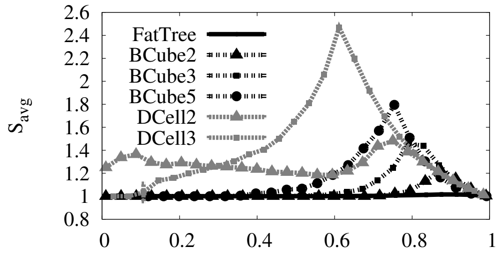
A. Link failures

Figure 4 shows the RS_{max} and S_{avg} metrics according to the link failure ratio in G. As given by RS_{max}, the network size of hybrid topologies has a smoother decay than that of Fat-Tree. Nevertheless, after a certain failure ratio on, all of the hybrid topologies present a point of inflexion, starting to reduce more sharply. Hence, for higher failure ratios the Fat-tree topology becomes more reliable. Nonetheless, a high failure ratio is hopefully not a realistic scenario.

Comparing topologies of the same type but with different parameters, such as DCell2 and DCell3, we observe that the ones with higher number of ports per server have a smoother degradation. Also, Fat-Tree, which only uses one-port servers, presents the worst performance. In an hybrid topology, when a link directly attached to a server breaks, the server can use the extra ports to connect to the other nodes. As a server uses more ports, the probability of this server being disconnected from the main component is smaller. The tradeoff, however, is a high path stretch, as we show later in this section. In Fat-Tree, the failure on a link directly connected to a server always disconnects this node from the network.

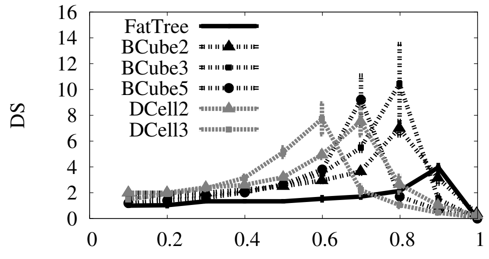


(a) Largest connected component.

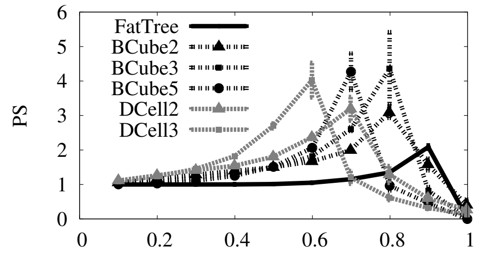


(b) Other components.

Fig. 4. Metrics related to network size considering link failures.



(a) Diameter Stretch.



(b) Path Stretch.

Fig. 5. Metrics related to path length considering link failures.

Moreover, RS_{max} shows that, for the same number of ports per server, BCube outperforms DCell. S_{avg} shows that failures in DCell tend to disconnect a larger number of servers than in BCube with the same configuration of ports per server. We also observe that S_{avg} is low for all topologies. This result indicates that the largest connected component degrades due to the removal of connected components with a small number of servers. Also, S_{avg} increases to a maximum then steadily decreases. These results also occur for exponential networks in the aforementioned work of Albert and Barabási [10]. The decreasing in S_{avg} occurs because as the number of connected components increases, the size of the largest component is reduced, increasing the removal probability of edges in the isolated components. Therefore, the peak value matches the inflection point of RS_{max} because in this point the size of the largest component decreases faster.

Figure 5 shows the impact of link failures on the network path length. DS and PS presents the same behavior differing only in their absolute values. Results show that all curves have a peak stretch value, from which we can conclude that link failures remove shortest paths until a point where the paths are shortened due to the decreasing network size. Also, path length increases fast, becoming as high as four times the original average length. The best performance in this analysis is achieved by Fat-Tree, despite the worst performance considering the RS_{max}. However, the number of servers in the main component of Fat-Tree is smaller than on the other topologies for a given failure ratio. Consequently, it is important to evaluate the reliability considering more than one metric.

B. *Switch Failures*

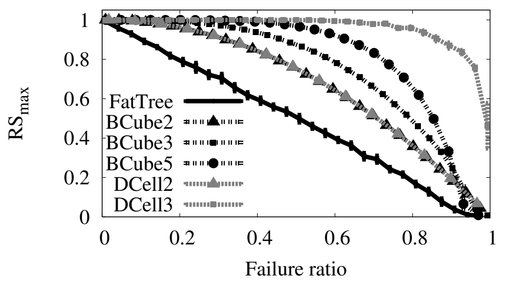
Figure 6 plots the behavior of network size considering switch failures. The curves of RS_{max} have a behavior similar to the case of link failures. However, the region after the inflection point is negligible. Although the curves of S_{avg} have no relevant peaks, the S_{avg} of DCell3 increases approximately 20 times after the elimination of most of its switches. Ignoring this behavior, as it represents an unreal failure ratio, we observe that switch failures produce small isolated components.

It is interesting to note the reliability of DCell3 with respect to the network size. The RS_{max} of DCell3 decreases only for a failure ratio greater than 60%. Comparing to Fat-Tree and BCube3 topologies, which have a close number of switches in relation to the number of servers (Table I), DCell3 has a better performance. Also, Figure 6(b) shows that DCell3 has no points for small failure ratios due to the lack of isolated components. This superior performance of DCell3 is related to its high dependence on servers, which is analyzed in the next section. As in our previous results, Figure 6 shows that the number of ports used by servers increases the reliability for a same topology type. This is because, as shown in Table I, topologies with higher number of ports per server have a lower switch port density, being less dependent on switches. Finally, DCell2 has the same performance of Bcube2.

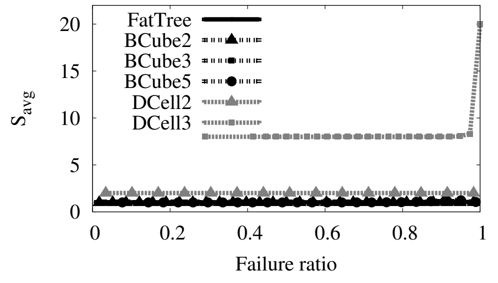
The results of path length in Figure 7 show that switch failures generally double the diameter size, having peaks only on values close to 90% of failure ratio for most of the topologies. Moreover, the path stretch changes slightly up to failure ratios of 90%. This indicates that switch failures have a lower impact on path stretch comparing to other failure types.

C. *Server Failures*

Figure 8 shows the results for server failures. It is important to note that, in our previous results, the term $\sum_{i=1}^{n} |s_i|$ of Equations 1 and 2 is constant and represents the total number of servers, because there is no removal of this type of element. In this analysis, however, $\sum_{i=1}^{n} |s_i|$ reduces of a unit for each server removal. Thus, as shown in Figure 8(a), Fat-Tree presents the maximum reliability ($RS_{max} = 1$). This happens because a server removal in this topology does not induce disconnected components since a Fat-Tree does not depend on servers to forward packets. The results also show that the other networks, except DCell3, are more reliable

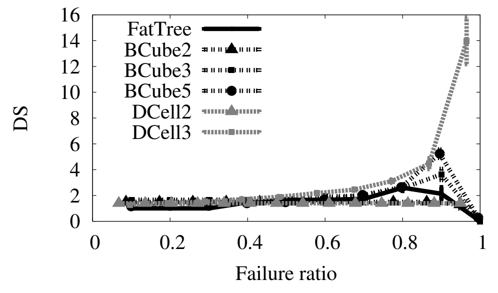


(a) Largest connected component.

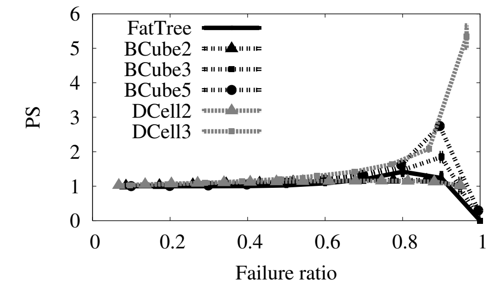


(b) Other components.

Fig. 6. Metrics related to network size considering switch failures.

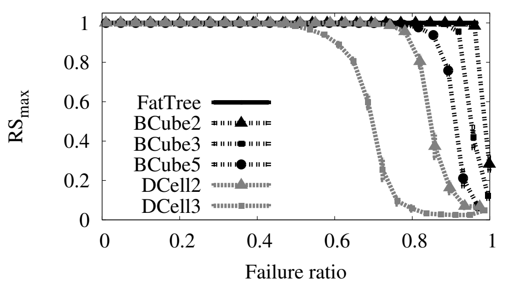


(a) Diameter Stretch.

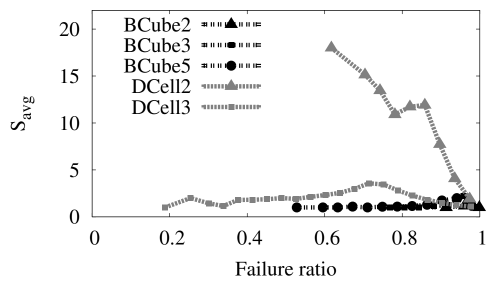


(b) Path Stretch.

Fig. 7. Metrics related to path length considering switch failures.

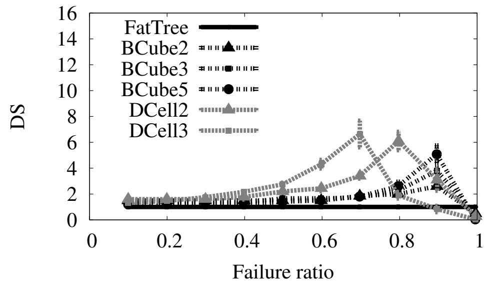


(a) Largest connected component.

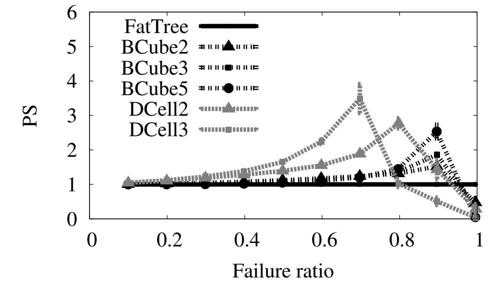


(b) Other components.

Fig. 8. Metrics related to network size considering server failures.



(a) Diameter Stretch.



(b) Path Stretch.

Fig. 9. Metrics related to path length considering server failures.

to server failures than to switch and link failures. In the case of DCell3, we show in Section IV-B its high reliability considering switch failures, because a significant part of this network remains connected through its servers. Figure 8(a) confirms that conclusion because DCell3 shows the earliest RS_{max} decrease when the server failure ratio increases. The results of S_{avg} show that the isolated components are also small comparing to the total number of servers, which is approximately 3,400. The absence of points in Figure 8(b) for failures up to a certain value, specific for each topology, shows that the topologies maintain one connected component for a long range of failures, except for DCell3 which has the worst performance. Figure 9 shows that DS and PS have the same behavior of those in Section IV-A, showing the significant impact that server failure produces on hybrid topologies.

V. Conclusions and Future Work

In this work we have evaluated the reliability of datacenter topologies proposed in the literature when subject to different element failures, revealing the tradeoffs of each topology design. We have observed that network degradation starts with the removal of connected components with a relatively small number of servers. Our results also have revealed that hybrid topologies degrades smoother than Fat-Tree with respect to the network size, in the case of link and switch failures. However, the reliability of BCube and DCell with respect to the network size is maintained at the cost of substantially increased path length. With these results we can also conclude that the network size or path stretch isolated does not provide an efficient way to evaluate the topology performance, and should probably be combined. Also, these metrics can be combined with other metrics like network cost and network capacity in a future work. Another future direction is to consider multipath routing, by analyzing path diversity.

Acknowledgement

This work was funded by CNPq, CAPES, CTIC/RNP, and FAPERJ. We thank the professors Stefano Secci and Daniel R. Figueiredo and the anonymous reviewers for their comments.

References

[1] R. Miller, "Google uses about 900,000 servers," http://www.datacenterknowledge.com/archives/2011/08/01/report-google-uses-about-900000-servers/, Aug. 2011.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM*, Aug. 2008, pp. 63–74.

[3] 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," in *ACM SIGCOMM*, Aug. 2009, pp. 63–74.

[4] 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*, Aug. 2008, pp. 75–86.

[5] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *ACM SIGCOMM*, Aug. 2011, pp. 350–361.

[6] L. Popa, S. Ratnasamy, G. Iannaccone, A. Krishnamurthy, and I. Stoica, "A cost comparison of datacenter network architectures," in *ACM Co-NEXT '10*, Dec. 2010, pp. 16:1–16:12.

[7] A. Curtis, T. Carpenter, M. Elsheikh, A. López-Ortiz, and S. Keshav, "REWIRE: An optimization-based framework for unstructured data center network design," in *IEEE INFOCOM*, Mar. 2012.

[8] 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*, Aug. 2009, pp. 51–62.

[9] A. Hagberg, P. Swart, and D S Chult, "Exploring network structure, dynamics, and function using NetworkX," (LANL), Tech. Rep., 2008.

[10] R. Albert, H. Jeong, and A. Barabási, "Error and attack tolerance of complex networks," *Letters to Nature*, vol. 406, no. 6794, pp. 378–382, 2000.

[11] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. Mogul, "SPAIN: COTS data-center ethernet for multipathing over arbitrary topologies," in *USENIX NSDI*, 2010, pp. 18–18.