On Random Wiring in Practicable Folded Clos Networks for Modern Datacenters

postprint version

C. Camarero, C. Martınez and R. Beivide, "On Random Wiring in Practicable Folded Clos Networks for Modern Datacenters,"

in IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 8, pp. 1780-1793, 1 Aug. 2018.

`https://doi.org/10.1109/TPDS.2018.2805344`

This is the author's version of the work. It is openly available to comply with institutional requirements. Not for redistribution.

Esta versión esta disponible para cumplir exigencias institucionales, no para su distribución.

Cristóbal Camarero, Carmen Martínez, and Ramón Beivide*

December 17, 2021

Abstract

Big scale, high performance and fault-tolerance, low-cost and graceful expandability are pursued features in current datacenter networks (DCN). Although there have been many proposals for DCNs, most modern installations are equipped with classical folded Clos networks. Recently, regular random topologies, as the Jellyfish, have been proposed for DCNs. However, their completely unstructured nature entails serious design problems. In this paper we propose Random Folded Clos (RFC) and Hydra networks in which the interconnection between certain switches levels is made randomly. Both RFCs and Hydras preserve important properties of Clos networks that provide a straightforward deadlock-free multi-path routing. The proposed networks leverage randomness to be gracefully expandable, thereby allowing for fine grain upgrading. RFCs and Hydras are compared in the paper, in topological and cost terms, against fat-trees, orthogonal fat-trees and random regular networks. Also, experiments are carried out to simulate their performance under synthetic traffic patterns emulating common loads present in warehouse scale computers. These theoretical and empirical studies reveal the interest of these topologies, concluding that Hydra constitutes a practicable alternative to current datacenter networks since it appropriately balance all the main design requirements. Moreover, Hydras perform better than the fat-trees, their natural competitor, being able to connect the same or more computing nodes with significant lower cost and latency while exhibiting comparable throughput.

1 Introduction

Datacenters are becoming critical components in modern industry and society. To be adequately supported, current Internet services and cloud computing require extremely powerful datacenters. Computing and storage systems in modern warehouse scale computers heavily rely on exploiting massive parallelism. Some recent installations contain around 10,000 racks comprising tens of thousands servers [12]. Having to manage such amount of communicating components, the datacenter network (DCN) plays a critical and pivotal role. In addition, according to [33], bandwidth demands in the datacenter are doubling every 12–15 months. Thus, that work and many others in the field, have highlighted the necessity of gracefully expandable topologies.

Although there have been many recent proposals for DCN topologies such as Dcell [19], BCube [18], HyperX [1], Jellyfish [34], and Space Shuffle [40], the reality is that most modern datacenters are equipped with a folded Clos network. Clos topologies [13] were proposed more than sixty years ago and have been extensively used in telephony and high-performance computing (HPC). After the Leiserson's work [27] and the Connection Machine commercialization [28], a specific family of folded Clos networks has been known as *fat-tree networks*. Many current supercomputers on the Top 500 list use fat-tree networks, or some of its variants [29]. Such non-blocking topologies can be built to an

*Some of the results in this paper were previously announced at 23rd IEEE Symposium on High Performance Computer Architecture. C. Camarero, C. Martínez, and R. Beivide are with the Department of Computer Science and Electronics, Universidad de Cantabria, Santander, Spain. email: {cristobal.camarero, carmen.martinez, ramon.beivide}@unican.es.

Table 1: Practical properties of the DCNs considered in this work.

Topology	**Routing**	**Expandability**	**Deployment**
fat-tree	very easy	poor	very easy
random graph	hard	extreme	very hard
RFC	easy	extreme	hard
Hydra	easy	enough	easy

arbitrary scale using regular switches with affordable radices (number of ports). Nowadays, big companies build their own fat-tree network fabrics, using general purpose merchant silicon instead of commercial switches [16]. This allows to have custom packaging, wiring and networks protocols.

Topologies based on random regular graphs have been considered long time ago for interconnection networks. The Jellyfish topology was recently proposed in [34] for DCN design. This network is based on building a random regular graph (RRG) on top of the rack switch layer, aiming to facilitate graceful datacenter expansion. Compared to a fat-tree, the Jellyfish can support 25% more servers at full bandwidth with the same switching equipment. The Jellyfish is a direct network while indirect ones such as fat-trees are predominately used in the datacenter industry. Although very interesting, the Jellyfish could be considered as a quite long and disruptive step forward in the design of networks.

A network without any structure and random links could greatly increase the complexity of cabling and routing when deploying a large-scale DCN. In addition, direct topologies, as the Jellyfish, are deadlock prone as they embed cycles and hence, must be equipped with an efficient deadlock-avoidance mechanism. This translates into higher cost and complexity. In the case of lossy networks, deadlock and congestion management can imply more packet losses and retransmissions which highly degrades performance. Although evolved network protocols, such as Shortest Path Bridging (IEEE 802.1aq), allows these networks to contain cycles, their presence always imply higher complexity and cost. On the contrary, one important advantage of fat-trees is their acyclic nature when using the standard up/down routing. This easily avoids severe problems as packet deadlock and broadcast storms and allows for extremely simple shortest multi-path routing.

For these reasons, this work explores an intermediate evolutive step based on the natural idea of randomizing the interconnection pattern between switch layers of folded Clos topologies; we have denoted them as *random folded Clos networks* (RFC) and they were announced in [10]. Similar and related structures have been known long ago, mainly associated with the introduction of expander graphs [4, 36, 15]. In the same way that fat-trees, RFCs conserve a structure based on different levels of switches which, under certain conditions studied in this paper, allows for the existence of common ancestor switches for every pair of communicating servers (computing nodes or network terminals). RFCs leverage this property for providing a simple deadlock-free shortest multi-path routing mechanism, identical to the one employed in standard fat-trees. Moreover, this work also explores a more conservative evolution of RFCs, denoted as *Hydra*, in which the random interconnection between some switches layers are substituted by traditional ones, *a la* fat-tree. As it will be shown, RFCs and Hydras constitute a compromise among cost, performance, scalability and expandability inside the class of the indirect topologies.

When looking for efficient DCNs, it is critically important to know in advance the expected traffic to achieve judicious designs. There are some studies suggesting that in certain datacenters, traffic is mostly uniform, [33, 21]. For example, in [33] the authors show that in Google datacenters, a big proportion of traffic is uniform, with small deviations. Specifically, [33] states "Recent work on alternate network topologies [...] deliver more efficient bandwidth for uniform random communication patterns. However, to date, we have found that the benefits of these topologies do not make up for the cabling, management, and routing challenges and complexity." Thus, in those cases, the design of DCNs should be driven by the most frequent uniform traffic scenario providing good enough performance for other more rare traffic patterns. The RFC and Hydra networks proposed in this research follow this design principle. Table 1 summarizes how the datacenter topologies considered in this work balance their fundamental design requirements.

To show the benefits of RFCs and Hydras, they are deeply studied in this paper to bring to light their outstanding properties, and to prove that they deserve to be considered for forthcoming DCNs. Summarizing, the main outcomes of this research are:

- A unified definition for RFCs and Hydras.
- A characterization of the diameter, the bisection bandwidth and the scalability of such topologies.
- A comparison, in terms of cost, expandability and deployment, of Hydras against other topologies (including random and non-random ones).
- An empirical performance evaluation of Hydras, comparing them against their natural competitor, *i.e.*, fat-trees, under different representative synthetic datacenter traffic patterns.

The rest of the paper is organized as follows. Section 2 reviews the most related proposals appeared in the technical literature. Section 3 defines folded Clos networks and reviews some of their topological properties. Section 4 defines RFCs and Hydras, and provides sufficient conditions for equipping them with the standard up/down routing. Also, a diameter, bandwidth and scalability characterization is done. Section 5 compares RFCs and Hydras with other topologies such as fat-trees, orthogonal fat-trees and random regular graphs in terms of expansion. Next, in Section 6 the physical deployment of these networks is considered. In Section 7, RFC and Hydra performance is evaluated under different synthetic traffic patterns. Finally, Section 8 concludes the paper summarizing its main findings.

2 Related Work

Although random graphs have been previously considered for interconnection networks, the recently proposed Jellyfish topology [34] has motivated the current work. Ten years before the Jellyfish was introduced, Lakamraju *et al.* proposed in [26] to randomly generate interconnection networks for parallel computers. Their idea was to generate many random graphs and to choose the best ones in terms of low diameter, high fault-tolerance and good embeddability of common applications.

Koibuchi *et al.* [25] put forward the use of random shortcuts in topologies for low-latency DCNs because they can be implemented for any size and their suitability for faulty scenarios. Later, Fujiwara *et al.* [24, 17] complete this research by adding random swaps between links. Although their methods and results mostly apply to direct topologies, they provide a first approach to indirect networks, considering wiring permutation in Myrinet-Clos networks.

Space Shuffle is another topology proposed for datacenter networks [40]. It connects routers in several cycles randomly, which allows for an easy non-minimal routing.

Scafida, a DCN architecture based on scale-free networks, reduces the average path lengths compared to other topologies with the same servers [20]. Small-World is another recent topology for DCNs, where several random links are added to ring, 2D torus, or 3D hexagon torus, while limiting the degree of each node [32]. These two architectures both employ random links. However, they require to manage the correlation among links, which is unknown when the networks expand.

Most of these studies focus on direct networks and much less attention has been paid to indirect networks. Nevertheless, it must be noticed that proposals which randomize the wiring between levels of multistage interconnection networks are not new. Around the 70's of the past century there was a considerable effort devoted to this kind of graphs with different applications. It seems that it was in [4] where random topologies for multistage networks appeared first. The authors proposed a method based on randomization to asymptotically find optimum nonblocking switching networks. That paper, although it seems to be almost unknown among the current interconnection network community, has had a great impact on Graph Theory, since expander graphs appeared firstly there. A related construction was considered independently by Upfal in [36], where splitter networks are defined to obtain low complexity deterministic packet routing schemes. Moreover, the Hashnet interconnection scheme proposed in [15] can be seen as an unfolded random Clos network. In that paper, its author already pointed out a possible application to parallel computer systems.

Related to expander graphs, an interesting novel result appeared in [37]. Their authors acknowledge that datacenters with direct random topologies outperform more sophisticated designs, achieving near-optimal throughput and bisection bandwidth, high resiliency to failures, incremental expandability, high cost efficiency, and more. Nevertheless, they complain about their unstructured nature and look for equivalent structured deployments. Through a combination of theoretical analyses and simulations, they conjecture that any expander topology comes with some of these benefits (random graphs are just an example of the expander family).

3 Folded Clos Networks

Since the original article by Clos [13] many authors have dealt with such networks. This paper considers Definition 3.1, which has been stated taken into account the original one given by Clos and also the one in the well-known book by Dally and Towles [14]. Table 2 contains the notation used in this article when dealing with folded Clos networks.

Definition 3.1. *A l-level folded Clos network is a topology in which switches are divided into l levels, where:*

- *Level 1 switches connect to compute nodes using down-links and to level 2 switches using up-links. These switches are called leaf switches.*
- *Level i switches, $1 < i < l$ connect to level $i-1$ switches using down-links and $i+1$ switches using up-links.*
- *Level l switches only connect to level $l-1$ switches. These switches are called root switches.*

A radix-regular folded Clos network is a folded Clos network in which all switches have radix R and i-level routers have $R/2$ down-links and $R/2$ up-links for every $1 \leq i < l$.

Table 2: Folded Clos Parameters, $\mathcal{P}$.

Parameter	**Definition**
T	Number of compute nodes or terminals
R	Switches radix (number of ports)
l	Number of levels
N_i	Number of switches at level i
k_i	Arity of the i-th level (as a tree)

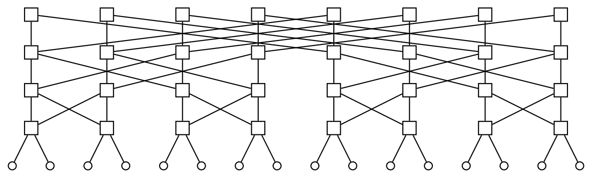


Figure 1: The 2-ary 4-tree.

In this paper, we are going to compare random folded Clos topologies against the popular fat-tree. The name fat-tree [27] originally made reference to a tree in which the edges closer to the root are 'fatter'—using 2^i parallel wires to connect a $i-1$ level switch to a i level switch. In most situations a root switch with so many ports is not feasible. Thus, the networking community uses this name to refer to a realization of the fat-tree as a folded Clos network with multiple roots. Next, the following recursive definition of fat-trees is considered, which has been taken slightly adapted from the paper by Petrini and Vanneschi [31]. A remarkable introduction to fat-trees can be found in [6].

Definition 3.2. *A l-level folded Clos network is called a fat-tree if the following recursive condition holds:*

- *there is a unique switch or,*
- *there is some integer $k_l > 1$ such that the switches up to level $l-1$ induce k_l disjoint fat-trees. The number k_l is called the* arity *at level l.*

If the recursive arities k_i have the same value k then the fat-tree is called a k-ary l-tree.

A 2-ary 4-tree, equipping 16 servers with diameter 6, is depicted in Figure 1; from here onwards, in the figures, boxes represent switches and circles represent servers or compute nodes. The fat-trees considered in [2] are another interesting case. A *R-commodity fat-tree* (CFT) is a radix-regular fat-tree whose arities are all $R/2$ except $k_l = R$. Figure 2 represents a four levels CFT for switches of radix 4, servicing 32 compute nodes with diameter 6. Note that the 4-level CFT connects twice number of computing nodes than the 2-ary 4-tree. Thus, for the same number of levels, a CFT doubles the number of nodes of the k-ary l-tree.

One of the advantages of CFTs is the simplicity of their deadlock-free routing. A switch is an *ancestor* of another one if it is possible to reach it by using only up-links. When two switches have a *common ancestor*, there is a path that connects them beginning with up-links until achieving the ancestor, followed by only down-links. If this happens for every pair of leaf switches, the folded Clos will be said to be *up/down-connected*. In such a case, its diameter D fulfills $D = 2(l-1)$, which bounds maximum latencies.

Finally, let us introduce *Orthogonal fat-trees* (OFT) [38], which are fat-trees inspired in the projective plane. They have recently deserved attention since they constitute a highly scalable cost-optimal topology for indirect networks [22]. Let q be a power of a prime number, then the l-level OFT of order q is a radix-regular fat-tree of radix $R = 2(q+1)$ and arities $k_1 = \cdots = k_{l-1} = q^2+q+1$ and $k_l = 2(q^2+q+1)$. The 2-level OFT is able to connect the maximum number of compute nodes for a given radix. Figure 3 shows a 2-level OFT; it equips 42 servers with diameter 2 but using, in this case, radix-6 switches. Minimal routes in the 2-level OFT are unique, which reduces worst-case performance.

As stated before, it is critical to know the expected traffic to achieve judicious network designs. There are evidences suggesting that datacenter traffic is random uniform in a great proportion, but it is not clear up to what extent. In the

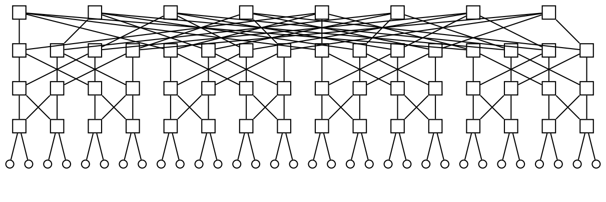


Figure 2: The 4-commodity fat-tree (CFT) or 4-port 4-tree.

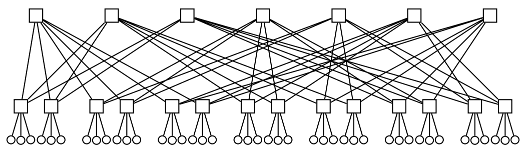


Figure 3: The 2-level orthogonal fat-tree (OFT).

HPC domain, however, there are many algorithms using some data permutations that must be routed at full rate to obtain competitive performance. This scenario suggests the use of full-bisection networks; a network is said to be *full bisection bandwidth* if its bisection bandwidth is enough to transmit all possible generated data in one of the halves to the other. For this reason, a CFT, which is full-bisection bandwidth, is worthwhile in HPC as it courses traffic at full rate regardless if it is uniform or any specific data permutation. Notwithstanding, there exist other networks (OFTs included) that, even not being full bisection bandwidth, can route uniform traffic at full rate. This is the case, for example, of dragonflies [23] which are gaining momentum in the HPC industry. Such networks manage adversarial traffic patterns by using Valiant random routing [39], which allows to achieve half of their maximum performance.

Coming back to modern datacenters, it would be critical to know the proportion of non-uniform traffic for proposing the more adequate network architecture. For example, if almost all the traffic were uniform, it could be considered to switch from a CFT to an OFT, which would imply trading worst case performance for more scalability. OFTs manage uniform random traffic at full rate, but are extremely poor dealing with data permutations. The RFCs and Hydras, introduced next, constitute a much more sensible alternative. They course at full rate uniform traffic while adversarial traffic can be routed with much more than 50% performance. This can be done even without using any randomization mechanism, as needed in dragonflies, which reduces complexity, cost, latency and energy.

4 Random Folded Clos and Hydra Networks

Random graphs are widely known in the mathematical literature [8]. The direct interconnection networks based on random graphs proposed in [26] and [34] select a randomly chosen graph among all the regular graphs with the same number of nodes and degree. It is very hard to quickly generate with uniform probability random regular graphs; thus, it is better to sacrifice uniformity a little bit. As far as we know, the best result for random graph generation is the one by Steger and Wormald [35]. An example of a random regular network (RRN) of degree 4 obtained with this algorithm can be seen in Figure 4; the degree of this graph is $\Delta = 4$,, the router radix R is 6 to accommodate 2 servers per switch and its diameter is $D = 4$. It is known that a Δ-regular random graph with N vertices of diameter D can be obtained with high probability if $\Delta^D \approx 2N \ln N$, [8]; thus, $\Delta \approx (2N \ln N)^{1/D}$.

Our proposal, introduced in next definition, is to force the random interconnection network to be a folded Clos.

Definition 4.1. *A random folded Clos (RFC) network with parameters $\mathcal{P}$, as those listed in Table 2, is a topology chosen randomly with uniform probability from all possible folded Clos networks with parameters $\mathcal{P}$.*

A random l-level folded Clos network can be generated using $l - 1$ random bipartite graphs. An algorithm to easily generate such graphs can be found in [10]. There are some old network proposals that can be related to (or included in) the previous definition. As an example, the Hashnet interconnect in [15] can be obtained by unfolding the RFC of the previous definition but with the same number of switches in all the levels. Other examples are random k-ary l-trees, which are really close to the definitions given in [4] and [36], although with more restrictions. For practical reasons, in this paper our interest is focused on radix-regular RFCs. In Figure 5 a 4-level RFC with radix 4 connecting 48 computing nodes is shown. It is important to highlight that this RFC, although having the same radix and number of levels as the CFT in Figure 2, it is able to connect 50% more computing nodes. It should be noted that a CFT connecting 48 servers requires not just 4 levels but 5, thereby with higher cost and increased latency as its diameter scales from 6 to 8.

4.1 Hydra Network

It can be argued that one of the most important drawbacks of a RFC would be its physical deployment due to the random wiring between levels. Therefore, this subsection considers how to balance randomness, which is beneficial for graceful expandability, while preserving some structure. As it will be shown, this will simplify its physical deployment, and will have impact on its scalability and performance.

The Hydra network is a topology built by adding to a RFC several k-ary l-trees. Its name comes from its structure: the RFC resembles the body of the Hydra while the multiple k-ary l-trees resemble the heads of the mythological creature. Figure 6 represents a specific Hydra. It should be highlighted that Hydra architectures are specially suited for a typical industrial deployment based on commodity pods or containers. The Hydra's heads represent, in the figure, such standard components. Next, the definition of these networks is given.

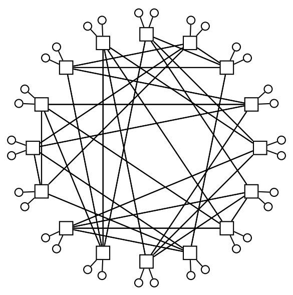


Figure 4: A random regular network (RRN) with 16 routers of radix 6.

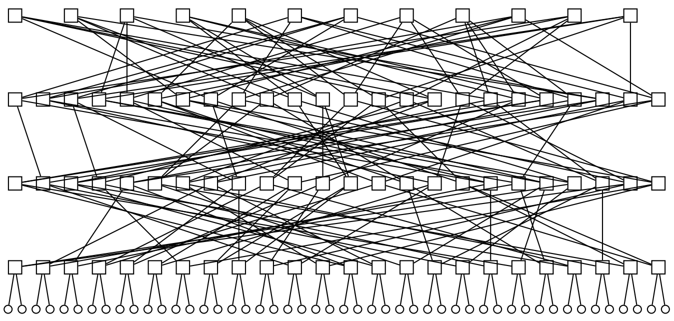


Figure 5: RFC of radix 4, $N_1 = 24$ and 4 levels.

Definition 4.2. *Let R be the router radix. A (R, l_{FT}, H, l_{RAN})-Hydra (of radix R, $1 \leq l_{FT}$ fat-tree levels, H heads and $2 \leq l_{RAN}$ random levels) is defined as an indirect folded Clos topology with $l = l_{FT} + l_{RAN} - 1$ levels. The number of switches in each level N_i is the same as in a radix R folded Clos network:*

- *$N_i = N$, for levels $1 \leq i \leq l - 1$,*
- *$N_l = N/2$,*

where $N = H(\frac{R}{2})^{l_{FT}-1}$. Each switch is labeled with a (i, j) pair, where $1 \leq i \leq l$ denotes the level and $1 \leq j \leq N_i$ denotes the position in the row level. The adjacency between switches is done as follows:

- *Switches of level $l - 1$ are connected with switches of level l as a random $(R/2, R)$-biregular bipartite graph.*
- *Switches of level i, for every $l_{FT} \leq i \leq l - 2$, are connected with switches of level $i + 1$ as a random $R/2$-regular bipartite graph.*
- *Switches $H_k = \{(i, j) \mid 1 \leq i \leq l_{FT}, \frac{N}{H}(k - 1) + 1 \leq j \leq \frac{N}{H}k\}$ are connected as a $R/2$-ary l_{FT}-tree, for every $1 \leq k \leq H$.*

Finally, every switch $(1, j)$ has $R/2$ processing nodes connected, for $1 \leq j \leq N$.

Example 4.3. *The network in Figure 6 is a $(4, 3, 8, 2)$-Hydra of 4 levels, a small example trying to illustrate the definition provided above. The topmost levels are randomly connected and the others are organized in 8 heads, each one composed of a 2-ary 3-tree. Each head has 4×3 switches and connects 8 computing nodes, which means that the whole datacenter has $8 \times 8 = 64$ computing nodes. Note that, this Hydra doubles the computing capacity of the CFT in Figure 2 with the same diameter ($D = 6$). A CFT accommodating 64 servers, requires one more level, hence $D = 8$.*

A Hydra then, is seen as an indirect topology in which the interconnection in a number of the top levels is random (like in the RFC) while the rest of the bottom levels are wired with the k-ary l-tree pattern. As it will be shown later, Hydras allow for a practicable physical deployment with a wiring complexity comparable to CFTs. The proportion of random and structured levels has implications in the performance and the scalability of the datacenter. In [10], it was already proved that the CFT always provides (in most cases, at a higher cost) the same or better performance than the RFC. Later in this paper, it will be proved that, the more the number of random levels a Hydra contains, the better is its performance, being the RFC—that is, a Hydra with all its wiring levels done randomly—the one with best performance. In terms of network size, the inverse relation is fulfilled; the more the number of fat-tree levels a Hydra contains, the better is its scalability.

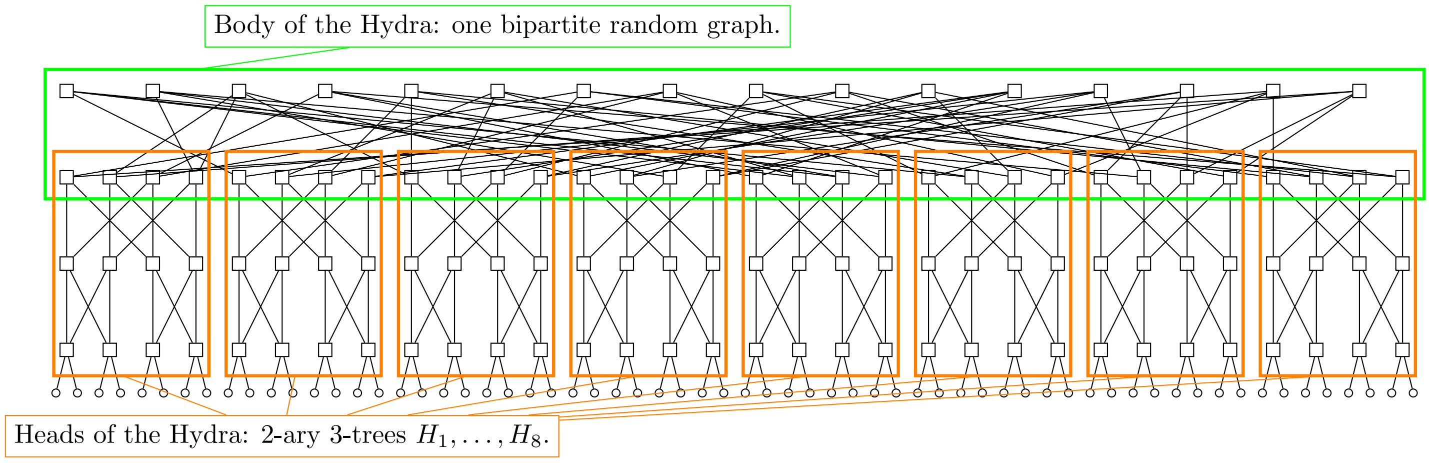

Body of the Hydra: one bipartite random graph.
Heads of the Hydra: 2-ary 3-trees $H_1, \ldots, H_8$.

Figure 6: A $(4,3,8,2)$-Hydra with radix $R = 4$, $l_{FT} = 3$, 8 heads and $l_{RAN} = 2$.

4.2 Existence of Common Ancestors and Routing

One important feature in a folded Clos network is being up/down-connected, which allows for the application of a simple deadlock-free routing. The following theorem states the probabilistic condition for a RFC network to be up/down-connected.

Theorem 4.4. *If* $\frac{R}{2} = \left(N_l(\ln\binom{N_1}{2} + x)\right)^{\frac{1}{2(l-1)}}$*, then the probability of each pair of leaf switches to have, at least, a common ancestor in a* l*-level* R*-radix regular random folded Clos network tends to* $e^{-e^{-x}}$*.*

Sketch of the Proof. Let λ be the expected number of pairs of leaf switches with disjoint ancestors set. Next, it will be shown some intermediate steps to prove that x equals $\ln\lambda$ in the limit (when $N_l \to \infty$). Thus, the threshold is obtained when $\lambda = 1$, that is, $x = 0$ or equivalently, when exactly one pair without common ancestors is expected. This is sharp threshold, which means that the probability of having common ancestors, abruptly changes from 0 to 1, except in a small region surrounding $x = 0$.

Next, the steps followed in the proof are summarized. Let $\Delta = R/2$. The number of ancestors of a given leaf is $\Delta^{l-1}(1 + o(1))$. First, it is shown that given two leaves a and b, the probability that they have disjoint set of ancestors is

$$P(\text{anc}(a) \cap \text{anc}(b) = \emptyset) = \frac{\binom{N_l - |\text{anc}(a)|}{|\text{anc}(b)|}}{\binom{N_l}{|\text{anc}(b)|}}$$

$$= \left(1 - \frac{|\text{anc}(a)|}{N_l}\right)^{|\text{anc}(b)|}(1 + o(1)).$$

Then, applying some technical lemmas, the logarithm of the expectation λ can be calculated as

$$\ln\lambda = \ln\binom{N_1}{2} + \ln\left(1 - \frac{|\text{anc}(a)|}{N_l}\right)^{|\text{anc}(b)|} + \ln(1 + o(1))$$

$$= \ln\binom{N_1}{2} - \frac{\Delta^{2(l-1)}(1 + o(1))}{N_l} + o(1)$$

Thus, replacing Δ by the expression in the statement it is obtained

$$\ln\lambda = -x(1 + o(1)) + o(1).$$

Therefore, the expectation λ tends to e^{-x}. The probability that every pair of leaf switches has at least a common ancestor is the probability of having exactly 0 pairs with disjoint ancestors set, so it is $(1 - \frac{\lambda}{\binom{N_1}{2}})^{\binom{N_1}{2}}$, which tends to $e^{-\lambda}$. □

Theorem 4.4 implies that $2(N_l \ln\binom{N_1}{2})^{1/(2(l-1))}$ is a sharp threshold for finding a RFC network being up/down-connected. For simplicity and resemblance with the direct random regular networks, we rewrite such a threshold as $2(N_1 \ln N_1)^{1/(2(l-1))}$. Note that $N_l(\ln\binom{N_1}{2}) \approx N_1(\ln N_1 - \frac{\ln 2}{2})$ and that $D = 2(l-1)$ would be the diameter in an up/down-connected RFC. When $x = 0$, the equality $R = 2(N_1 \ln N_1)^{1/(2(l-1))}$ implies that the probability converges to

Table 3: Probability of having common ancestors in 36- radix RFCs.

l	D	x	$e^{-e^{-x}}$
2	2	≈ -17.88	≈ 0
3	4	≈ 0.95	≈ 0.68
4	6	≈ 6104.26	≈ 1

Table 4: Probability of having common ancestors in Hydras.

l_{FT}	l_{RAN}	D	x	$e^{-e^{-x}}$
2	2	4	≈ -17.27	≈ 0
3	2	6	≈ 0.63	≈ 0.58
2	3	6	≈ -3.77	≈ 0
3	3	8	≈ 1093.88	≈ 1
1	4	6	≈ -8.16	≈ 0
1	5	8	≈ 1085.08	$\approx 1.$

a constant; so it is easy—after a few random graph generations—to build an up/down-connected RFC. In this case, $e^{-e^{-x}}$ becomes $1/e$ which means that, in average, an up/down-connected RFC is obtained, more or less, every three times the generating algorithm is executed [10]. Small positive or negative values of x quickly impact on the probability of finding RFCs being up/down-connected. For example, if $R = 2(N_1 \ln N_1 + \ln\ln N_1)^{1/(2(l-1))}$ then the probability of having common ancestors tends to 1 and if $R = 2(N_1 \ln N_1 - \ln\ln N_1)^{1/(2(l-1))}$, such probability tends to 0.

Example 4.5. *Let us fix $R = 36$ and $T = N_1\frac{R}{2} = 200,000$ servers to illustrate Theorem 4.4 with a specific example. The limit of the probability changes with the number of levels (or equivalently the diameter), as shown in Table 3. If $l = 2$ there is no up/down-connected RFC for such a number of servers. If we increase the number of levels to $l = 3$ then, most of the times, the RFC is up/down-connected. Finally, for any number of levels greater than 3, the RFC is always up/down-connected. Note that a CFT would require 4 levels for this number of servers.*

As done previously for RFCs, Theorem 4.6 bellow, establishes a sufficient condition under which a Hydra is up/down-connected.

Theorem 4.6. *Let $l = l_{FT} + l_{RAN} - 1$. If $\frac{R}{2} = \left(N_l(\ln\binom{H}{2} + x)\right)^{\frac{1}{2(l-1)}}$, then the probability of each pair of leaf switches to have a common ancestor in a (R, l_{FT}, H, l_{RAN})-Hydra network tends to $e^{-e^{-x}}$.*

Example 4.7. *Let us illustrate Theorem 4.6 using another example. As different configurations of random and structured levels are needed, let us fix the radix $R = 18$ and the number of servers $T = N_1\frac{R}{2} \approx 700,000$. In Table 4 it is shown how the probability of being up/down-connected changes when the number of levels is varied. First, there is no up/down-connected Hydra able to connect 700,000 computing nodes with just 3 levels. For 4 levels, only the Hydra network with the maximum number of structured levels, that is $l_{FT} = 3$, is up/down-connected with high probability. Note that, when $l_{FT} = 1$, the resulting Hydra is indeed a RFC. If the total number of levels is increased to 5, then, all the configurations are up/down-connected, even the RFC. Note that a CFT would require 6 levels for this number of servers.*

It is important to note that, an up/down-connected Hydra can be built using a RFC which is not up/down-connected. Consider a $(18, 3, 960, 3)$-Hydra included in the previous example; the contained 3-level RFC is not up/down-connected although the whole Hydra does.

Later it will be seen that Hydra networks are more scalable as the number of fat-tree levels is greater. That is, for some number of levels it is easy to find an up/down-connected Hydra but nearly impossible to find such a RFC. Remember also that $l_{RAN} \geq 2$; trying to have no randomness would result (after some adjustments in the definition) in a CFT, which is less scalable than a RFC.

4.3 Diameter and Bisection Bandwidth

As stated before, the diameter of a RFC or Hydra being up/down-connected is trivially upper bounded by $2(l-1)$. As a consequence of Theorems 4.4 and 4.6, the threshold in which the diameter changes is known. A RFC fulfilling equation $R = 2(N_1 \ln N_1)^{1/(2(l-1))}$ is, with high probability, able to connect the maximum number of compute nodes for a given diameter. The same happens if a Hydra network fulfills the equivalent expression in Theorem 4.6. For both networks, any other construction, not fulfilling these equalities, can be obtained by expanding the original network without adding a new level, as it will be shown in Section 5. After further expansions, it is necessary to add a new level. The expanded network can again connect the maximum number of compute nodes, but for the next possible diameter.

Figure 7 shows the diameter evolution of the topologies considered in this paper with $R = 36$. This value has been chosen since it probably represents the most common degree in current commodity switches. The CFT and OFT can connect only a certain number of processing nodes for a given diameter. Any CFT or OFT between these points can be built using the one with one more level but depopulated. On the contrary, RRNs, RFCs and Hydra networks, with a fixed diameter can be implemented for many other numbers of switches, which is represented by solid lines; note that the marks in the solid lines are only used to distinguish them. RFC and Hydra networks are rather similar to RRN with the major difference that the first ones can only have even diameters. The figure also indicates that random topologies admit an amount of compute nodes between the CFT and the OFT.

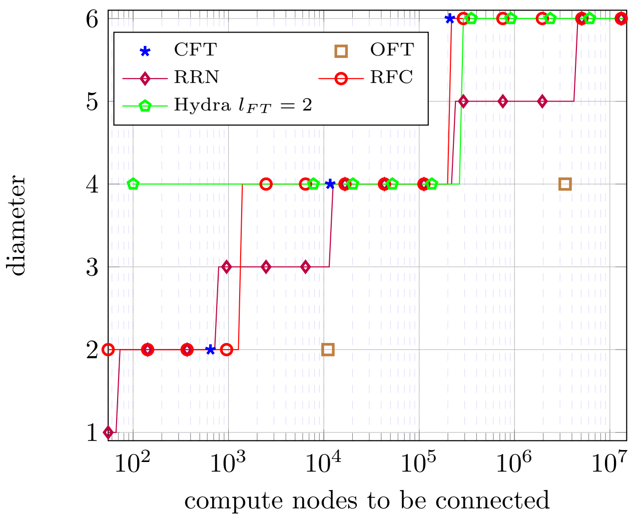


Figure 7: Diameter evolution for different networks with radix $R = 36$.

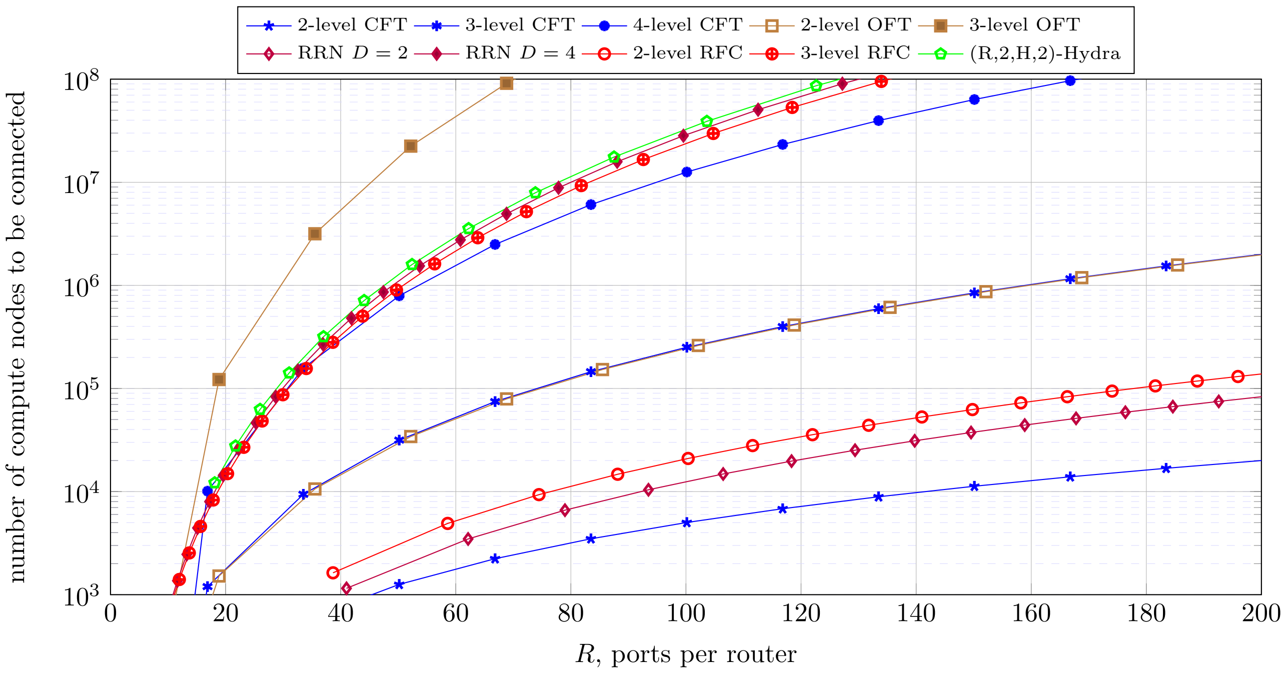


Figure 8: Network scalability comparison of the different topologies considered in the paper.

As an example to illustrate the resemblance between direct and indirect random networks, consider diameter 4 and radix 36. Theorem 4.4 indicates that the limit of a realizable RFC is slightly above $N_1 \approx 11,254$, which implies about 202,554 compute nodes. In the case of RRNs, the radix shared between network ports, Δ, and server ports must be chosen *a priori*; let the degree be $\Delta = 26$ with 10 compute nodes per switch (optimized for an average distance around $26/10 = 2.6$). In this case, the RRN can be built with $N = 22,773$ switches (remember that $\Delta^D \approx 2N \ln N$), which can connect 227,730 compute nodes, 12% above the RFC. The corresponding CFT of diameter 4, has 3 levels (as the RFC) but it can only accommodate 11,664 computing nodes.

In respect to the bisection width, random regular graphs have been known to be good expanders for a long time [4], [3]. This implies a good isoperimetric constant, or *Cheeger constant*, which in turn implies large bisection. Bollobás [7] proved that the isoperimetric number of a Δ-regular random graph is at least $\frac{\Delta}{2} - \sqrt{\Delta \ln 2}$. As a consequence, a RRN has a bisection width of, at least,

$$BW_{RRN} \geq \frac{N}{2}\left(\frac{\Delta}{2} - \sqrt{\Delta \ln 2}\right)$$

links between the two halves of any cut of the network into two parts of the same size.

For a RFC it is possible to calculate such a value using the previous expression. Let S and $\bar{S}$ be the two halves of the worst possible—with least bandwidth—cut of the nodes into two sets with the same size. Now, let us make the following identification. For every switch of the upmost level, let us choose two switches in each level without repetition, and generate the graph identifying them. This is a random regular multigraph with $N_1/2$ vertices of degree $2(l-1)R$. Then, using the lower bound by Bollobás, the bisection width contains at least

$$BW_{RFC} \geq \frac{N_1}{4}\left((l-1)R - \sqrt{2(l-1)R \ln 2}\right)$$

links. Note that for the Hydra network, the expression is the same, but considering the number of random levels l_{RAN} instead of the total number of levels.

As an example, let us consider again networks with $R = 36$ and give their normalized bisection width, that is, the bisection divided between the number of compute nodes in one of the halves times the average number of traversals of the bisection. In CFTs, each path traverses the bisection at most once regardless the traffic pattern. However, it is easy to see that in RFCs the average number of traversals is $l-1$, equivalently $l_{RAN} - 1$ in Hydra networks. The CFT is full-bisection bandwidth and has normalized bisection 1. Bollobas' bound for a RRN gives 0.88. For a 2-level RFC this bound gives 0.80 and 0.86 for a 3-level RFC. Note that the bound gives the same value for $(36, 2, H, 2)$-Hydra than for 2-RFC. These four topologies perform almost at full rate under uniform traffic. However, there are adverse traffic patterns in which they perform slightly worse but, as it will be seen in Section 7, coursing such more rare traffic at a very good rate.

4.4 Scalability, Latency and Cost

Current datacenters aim to reach large sizes but if a bound is imposed in their DCN diameter, they are limited by the number of ports in their switches. Next, the term *scalability* is used as a measure of how many computing nodes can be arranged in a network given a router radix and a diameter.

In [9], it was shown that, if all compute nodes inject at the same rate as network links, then $\frac{T}{N} = \frac{\Delta u}{\bar{k}}$ results in a well balanced network, where $\bar{k}$ is the average distance, Δ is the graph degree and u is the average link utilization. Since in RRGs $\bar{k}$ is close to the diameter D and u is relatively large, which has been experimentally observed to be about 0.7, this expression can be approximated by Δ/D compute nodes per switch. Thus, the number of ports per router is $R = \Delta(1 + 1/D)$ and the total number of compute nodes is $T = N\Delta/D$. Now, it follows that

$$T = \frac{\Delta^{D+1}}{2D \ln N} = \frac{(R/(1+\frac{1}{D}))^{D+1}}{2D \ln N} = \frac{1}{2D(1+\frac{1}{D})^{D+1}} \frac{R^{D+1}}{\ln N}.$$

A RFC of radix R, with N_1 leaf switches and l levels has diameter $D = 2(l-1)$ if Δ satisfies $\Delta^D \approx N_1 \ln N_1$ (in the case of indirect networks, $\Delta = R/2$). The number of compute nodes per leaf switch is Δ and the total is $T = N_1\Delta$. Now, it follows that

$$T = \frac{\Delta^{D+1}}{\ln N_1} = \frac{(R/2)^{D+1}}{\ln N_1} = \frac{1}{2^{D+1}} \frac{R^{D+1}}{\ln N_1}.$$

Analogously, a Hydra in the same conditions with H heads fulfills that

$$T = \frac{1}{2^{D+1}} \frac{R^{D+1}}{\ln H}.$$

As stated in Section 3, a R-radix CFT has $T = 2(\frac{R}{2})^l$ compute nodes. Also, a l-level OFT of order a prime power q has radix $R = 2(q+1)$, $N_1 = 2(q^2+q+1)^{l-1}$ leaves and $T = 2(q+1)(q^2+q+1)^{l-1}$ compute nodes, so $T \approx R\left(\frac{R}{2}\right)^{2(l-1)}$.

Figure 8 shows the scalability of the different topologies. In abscissas the switch radix is represented. In ordinates the number of compute nodes in logarithmic scale is shown. There are three curves, one for each CFT with 2, 3 and 4 levels, with diameters 2, 4 and 6 respectively; there are other four curves for each RFC and Hydra with 2 and 3 levels (diameters 2, 4) and two more curves for the corresponding RRGs. A first look at the figure shows that for the same number of servers, in general, random topologies can be built with routers of much lower radices than the ones employed in CFTs. Moreover, in many practical cases, indirect random topologies can service a much larger number of terminals with less levels. This is exacerbated in the case of the OFT that shows the highest scalability but, as it will be shown in the next section, its strict definition presents an important drawback that compromises its expandability. Notwithstanding, it is noticeable that the 2-level OFT scales as the 3-level CFT, halving its diameter; OFTs with more levels exhibit an impressive scalability difference over CFTs.

Concerning RFC and Hydra, they clearly scale much better than CFTs. It should be highlighted that the scalability of indirect random networks is similar to the direct RRN with the same diameter, being the Hydra the one providing the highest scalability among them. In order to extract other practical consequences from Figure 8, let us to concentrate on 4-level (diameter 6) CFTs, which are quite common on large data center deployments. For the whole range of router radices, there is always a 3-level (diameter 4) Hydra able to service much more computing nodes. This means lower average and maximum latencies (up to 33% reduction in diameter), which also translates in lower energy consumption. In addition, very simple counts reveal an important ($2/7 = 28\%$) reduction in the number of switches and ($1/3 = 33\%$) reduction in the number of wires. Similar reasoning can be applied to RFCs although they are a little bit less scalable.

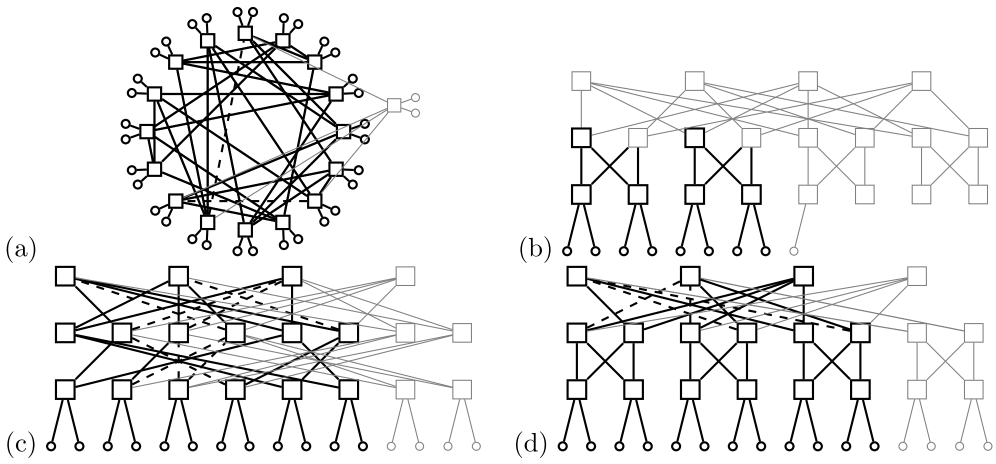


Figure 9: Expansion of (a) RRN, (b) CFT, (c) RFC and (d) Hydra networks.

5 Expandability

Different terms have been used to define the possibility of incrementally adding more compute nodes to a network. In this paper, we use the term expandability as with the Jellyfish in [34]. Therefore, given a network with N switches of radix R, a *N'-expansion* is an upgrade of the current configuration to a network with $N + N'$ switches and the same radix R and diameter. Then, a network will be said to be expandable if there is a $N' > 0$ such that it is possible to make a N'-expansion.

The CFT is not expandable since its definition imposes to increase the number of levels to add new switches, which implies increasing the diameter. It is also needed to rewire half of the wires on the top level. In the same way, the OFT is not expandable. Its strict definition involves a huge amount of new switches although the rewiring is the same, respect to its total number of links, than in the CFT.

However, RRN, RFC and Hydra are all of them expandable networks. Figure 9 tries to illustrate how these networks are expanded; for every network a small example is done. In each example, a network is represented with thick lines, its minimal expansion with thin lines, and with dashed lines the wires removed for it.

A Δ-RRN with N switches can be easily upgraded to a Δ-RRN with $N + N'$ switches, where approximately $\Delta N'/2$ links must be rewired. For conserving the diameter D, it must maintain $\Delta \approx (2(N' + N)\ln(N' + N))^{1/D}$. Hence, a problem in the RRN expansion is the potential incorrect balance in the switches between ports for terminals and ports for connecting to other switches. As mentioned in Subsection 4.4, the proper number of servers per switch is around Δ/D. Nevertheless, the number of computing nodes per switch must be kept constant along expansion, so there is a single point along the expansion in which the network is well balanced and dimensioned.

RFCs, similarly to RRNs, are also expandable as new switches can be added without increasing the network levels. In this case, minimal upgrades add two new switches in every level and only one on the top level. Thus, at each incremental expansion it is possible to add R new compute nodes. This can be done till achieving the threshold in Theorem 4.4. In other case, the up/down-connectedness is not guaranteed but this can be avoided by increasing the number of levels as other tree-based networks do.

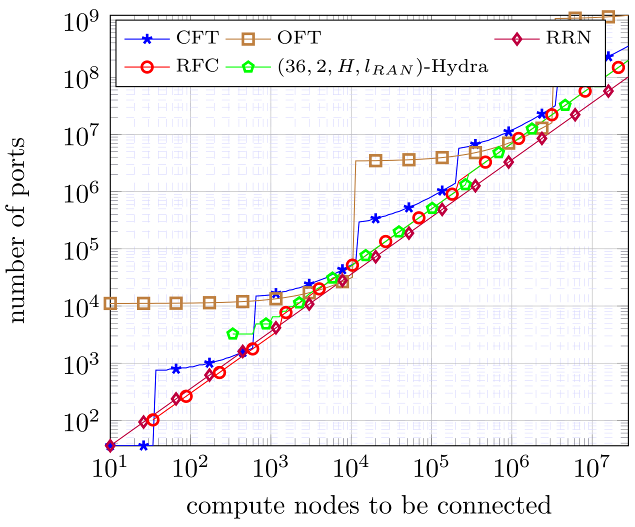
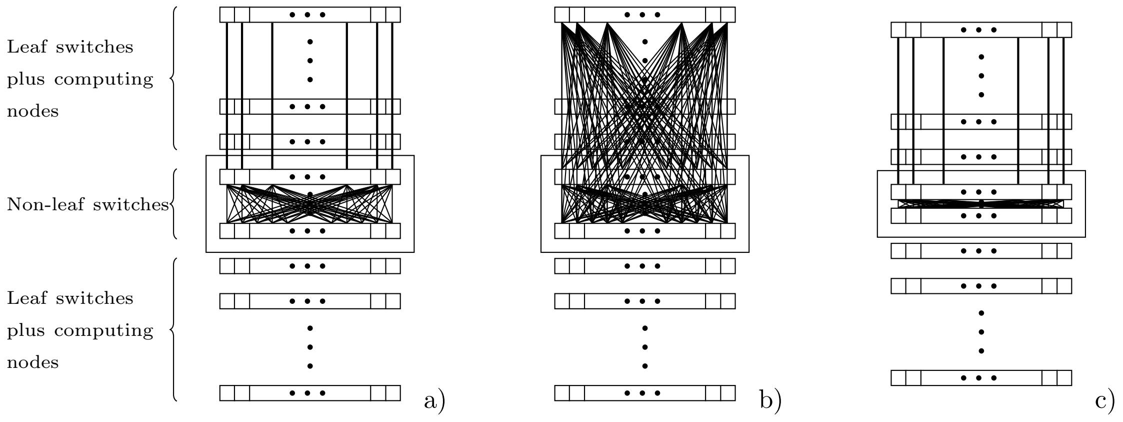


Figure 10: Expandability of RFC and Hydra with radix 36.



Leaf switches plus computing nodes
Non-leaf switches
Leaf switches plus computing nodes
a)
b)
c)

Figure 11: Centralized layout of networks with radix $R = 18$ and $T = 13.122$ computing nodes: (a) 4-level CFT (b) 4-level RFC and (c) (18, 2, 162, 2)-Hydra.

Hydra networks are also expandable. In this case, every incremental expansion is done adding new heads and the necessary switches in the random levels. The minimum number of heads to be added is determined by the number of switches in each level a head has. If this number is even, then it is possible make an expansion in just one head. If it is odd, it is necessary to add an even number of heads (at least two).

In Figure 10, network expandability against raw cost is shown. The number of computing nodes is represented in abscissas. The ordinates represent the total number of ports, a coarse-grain measure of the network cost; note that the number of network wires is half the number of network ports. As it can be observed, non-random topologies (CFT and OFT) result in functions characterized by their great discontinuity. Each step means an upgrade by adding a new level. Then, the continuous part of the function gives the number of compute nodes which is possible to add without requiring a new level. Note that for the calculation only the necessary switches to connect the new computing nodes are considered, since in these cases the network is built depopulated. All random topologies lead to almost linear functions, which reinforces that they are really suitable for expansion, as it was already proved in previous papers for RRNs. The small steps in the function for RFC correspond to the addition of a level to guarantee the up/down-connectedness, as argued before. Note that the random topologies have more or less the same cost. Observe that, due to the logarithmic scale in Figure 10, differences in raw cost can be really huge, always in favour to the random topologies. As it will be shown next, even for specific configurations in which a CFT is fully populated, our networks clearly beat them in terms of cost.

It is also worthwhile to note that, when upgrading random topologies, the rewiring needed is similar. For example, if a RRN and a RFC with $R = 36$ and $T \approx 10,000$ are both increased by 180 compute nodes, then the required rewiring in both topologies is approximately 1.8% of the total number of links. However, Hydra needs less rewiring, proportionally to the number of random levels.

Table 5: Maximum and total cable length for $13,122$ processing nodes and radix 18.

	Switches	**Wires**	**Tot. length**	**Max. length**
4-level CFT	5,103	39,366	879,000	44
4-level RFC	5,103	39,366	1,161,000	68
$(18, 2, 162, 2)$-Hydra	**3,645**	**26,244**	**479,000**	**42**

6 Deployment

In this section, the physical deployment of the topologies considered in the paper is analysed. For this study, let us assume a typical centralized deployment, that is, all the non-leaf switches are contained in the racks of central rows, similarly to [33]. Other options to deploy CFTs, such as [2], are not considered in this paper. In the remainder of the section an example is used to illustrate the problem.

Let $R = 18$ be the switch radix and $T = 13,122$ the number of processing nodes, which is the maximum in a 4-level CFT. In Figure 11, the centralized deployment of three different networks is considered. Note that, for the sake of the clarity, not all rows of racks and wires are represented. There are two types of rows, the ones devoted to leaf switches and servers, and the ones in the central rectangle containing the remaining switches. In Figure 11(a), the deployment of the 4-level CFT is shown. Assuming that racks have capacity for allocating 10 swithes/computing nodes, there are 40 racks in each row. In the 8 rows of the central rectangle, there are 40×8 racks containing only non-leaf switches. The remaining racks, 16 rows above and 16 rows down, contain servers and Top of the Rack (ToR) switches. As it can be seen, due to the structure of the CFT, all the wires from ToR switches to the central racks can be handled in bundles. However, if we consider RFCs, things are really different as shown in deployment in Figure 11(b) in which the layout of the 4-level RFC is presented. In the figure it can be seen that the wires in the central racks, conveniently managed in hoses, are the same than in the CFT. However, wires from/to ToR switches cannot be packaged into bundles, which is clearly both the main drawback of RFCs and the main motivation behind Hydra networks. In layout in Figure 11(c), a (18, 2, 162, 2)-Hydra is deployed, which is capable to connect also $13,122$ computing nodes but with one less level, which translates into less rows of racks in the central rectangle (5 instead 8). It can be observed that, the wires from ToRs in the Hydra can be bundled as in the CFT, concluding that the wiring complexity of both topologies is similar, but with less and shorter wires in the case of the Hydra, as shown next.

Considering the previous physical organization, let us to compare CFT, RFC and Hydra in terms of total and maximum cable length. Table 5 summarizes the differences among these specific configurations. Only wires between switches are counted. It is assumed a square grid of racks using Manhattan distance. For the calculation of cable length, it has been assumed that every cable uses 2m to connect endpoints, plus 1m per rack crossed, similarly to [5]. As can be seen, since the Hydra network has less levels, it has less wires, switches and total and maximum wire length. It should be noted that this comparison has been done for a number of servers which favours CFT, as it is fully populated. Even in this case, the corresponding Hydra can be deployed with savings of 29% and 44% in switches and wires, respectively. Any other number of servers leads to higher savings. For this and other radices, as R=36 in Figure 10, huge savings can be achieved when dealing with a number of servers leading to partially populated CFTs. Moreover, as it will be seen in Section 7, it comes with a noticeable latency improvement.

7 Performance Evaluation

In this section several experiments are shown to evaluate the different networks and to study the implications of adding more/less random levels to a Hydra. These experiments are performance simulations under different synthetic traffic patterns. Simulations have been done using INSEE (Interconnection Network Simulation and Evaluation Environment) [30] with the parameters shown in Table 6. Each run consists of $10,000$ cycles for statistics, preceded by a network warmup. At least 5 simulations are averaged for each point. The employed up/down routing is deadlock-free but virtual lanes are used to reduce Head of Line (HoL) blocking.

All the experiments have been done using three synthetic traffic patterns, adapted from [11], which have been selected to represent typical communication patterns. The traffics are:

- *uniform*: Each generated packet has as destination a random computing node selected uniformly.
- *permutation*: The set of computing nodes is initially divided into pairs in a random uniform way. Each compute node generates packets with destination its paired compute node. This traffics pattern represents the case of a random data permutation of the computing nodes.
- *fixed random*: At the beginning, each computing node selects a different computing node in a random uniform way. During the simulation each node generates packets towards the selected computing node. Note that, since several computing nodes could have selected the same destination, these preferred endpoints become light hotspots.

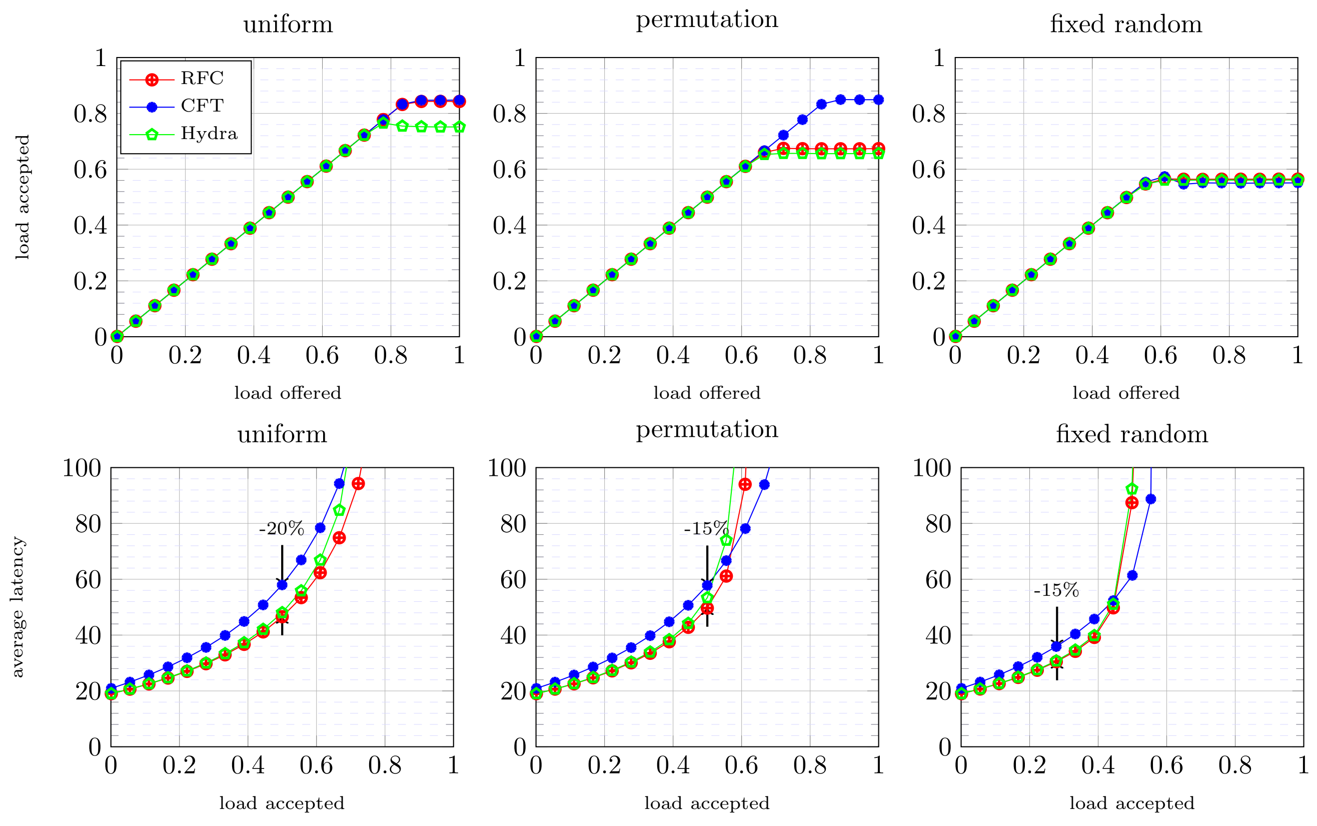


Figure 12: Simulated throughput and latency of 3-level RFC, (36,2,309,2)-Hydra and 4-level CFT with radix 36 and 100,008 compute nodes.

In Figure 12, throughput and latencies for different networks are shown. In this case, the networks evaluated have 100K computing nodes. Both RFC and Hydra have 3 levels, but the CFT needs 4 levels to accommodate such number of computing nodes. Hence, the CFT for these experiments is not a fully populated one. Note that results are shown with normalized load, so 1 in the plots means that every computing node is injecting 1 phit per cycle. As it can be observed, under uniform traffic both RFC and CFT exhibit the same performance, and the Hydra provides 11% less maximum throughput than the CFT. However, both RFC and Hydra networks provide 20% better average latency since they have less levels.

When considering permutation traffic, RFC provides 79% and Hydra 77% of the performance given by CFT. This is not surprising since CFT is, by definition, a full bisection bandwidth network. Again, as a consequence of having less number of levels, the average latency improvements of RFC and Hydra are around 15%. Finally, if we consider the performance under fixed random traffic, all topologies have almost identical throughput, which translates on that all of them tolerate equally well traffic generating hot spots. The average latency improvements of RFC and Hydra are again in the surroundings of 15%. Moreover, as it was shown in Section 6, having less levels implies lower cost, power and

Table 6: Simulation parameters

Parameter	**Value**
Simulated cycles	10,000
Virtual channels	4
Buffer size	4 packets
Flow control	Virtual cut-throught
Injection mode	shortest
Request mode	up/down random
Arbiter	random
Packet length	16 phits
Link latency	1 cycle
Arbitration iterations	1

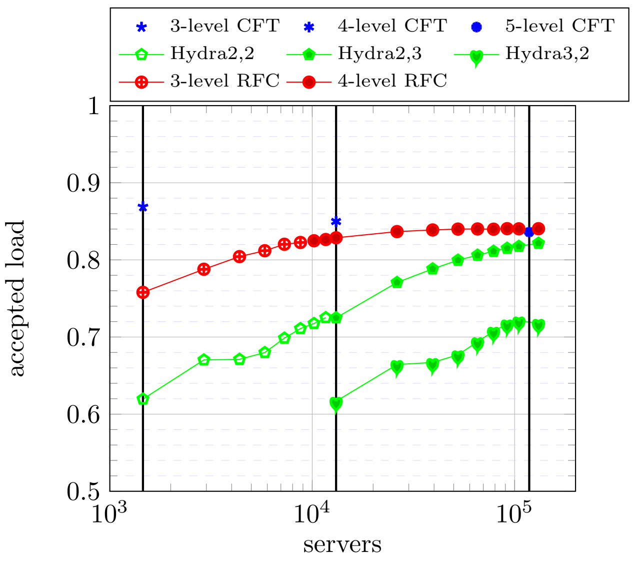


Figure 13: Maximum throughput in uniform traffic and radix 18.

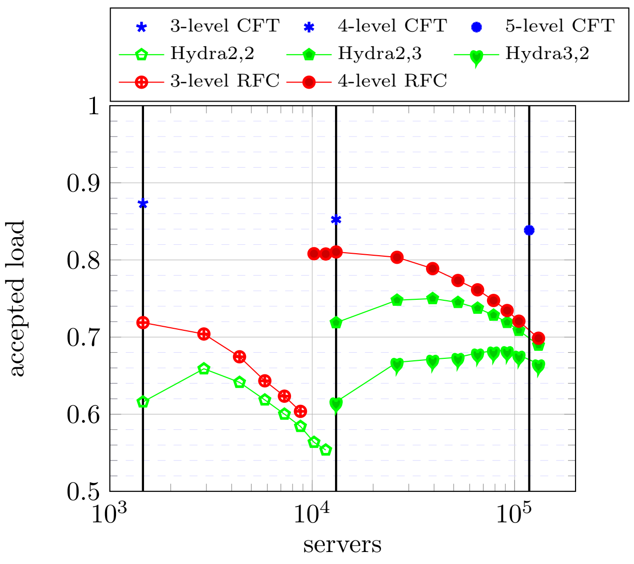


Figure 14: Maximum throughput in permutation traffic and radix 18.

energy.

Figures 13, 14 and 15 show the maximum throughput achieved for different networks for each traffic pattern. In these experiments, the effect on performance of two different network upgrades is going to be studied. It will be considered that networks can be expanded, up to a point, by widening them or, after that, by adding more levels. In order to compare Hydra networks varying the amount of random levels, networks with more than three levels are also considered. For all of these reasons, and for correctly illustrating a wide range of examples, the radix of the topologies have been fixed to 18.

In each figure, in abscissas the number of compute nodes is represented in logarithmic scale. In ordinates, the maximum throughput is shown. The number of computing nodes is divided into three different regions, which are delimited by solid vertical lines. These lines correspond to the number of computing nodes of a CFT with 3, 4 and 5 levels, respectively. Results are normalized with respect to an ideal full bisection bandwidth network. In adition to CFTs, five networks are evaluated: 3-RFC, 4-RFC, $(18, 2, H, 2)$-Hydra, $(18, 3, H, 2)$-Hydra and $(18, 2, H, 3)$-Hydra. Note that the random networks that appear in each region have always one level less than the corresponding CFT. This will be translated in latency reductions, as previously shown in Figure 12.

Inside every region, the expansion of the networks is done by adding computing nodes and switches, preserving the number of levels. However, from the first region to the second one, the upgrade is made by adding a new level. Therefore, 3-RFC is upgraded to 4-RFC. Also, the $(18, 2, H, 2)$-Hydra is upgraded to $(18, 2, H, 3)$-Hydra. Observe that

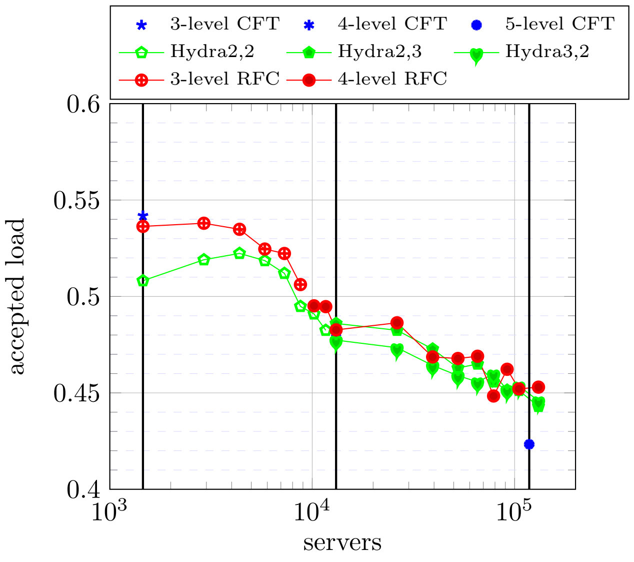


Figure 15: Maximum throughput in fixed random traffic and radix 18.

the last region is not delimited since random topologies would be able to attach more than 500K computing nodes. These points are not included since the simulations come with a very high computational cost.

As it can be observed in Figure 13, RFC always provides better performance under uniform traffic than any Hydra. Moreover, more random levels implies better performance, as observed for the two 4-level Hydra networks. For any of the topologies, fine grain expansions provide for small performance improvements. Most of the random topologies tend to achieve the good behaviour of the CFT under uniform traffic. In permutation traffic, as it can be seen in Figure 14, again more random levels implies a higher maximum throughput. In general, and opposite to uniform traffic, fine grain expansions preserving the number of levels produces a small degradation of the network performance. However, the upgrade by adding a new level involves an improvement of 30%. All the networks under fixed random traffic have similar behaviour, as shown in Figure 15. Finally, as it can be observed, for very large datacenters (corresponding to the last region results), Hydra comes with many benefits losing no significant throughput. While for moderate sized datacenters, there is a more noticeable throughput loss. As it has been demonstrated in previous sections, Hydra networks are more scalable, enough expandable and have a manageable deployment with less cost. The datacenter's designers should decide if the trade-off is advantageous enough for their purposes.

8 Conclusions

Random multistage topologies were proposed decades ago. In this paper, we have analyzed and characterized a specific family of such networks, denoted as Random Folded Clos networks. Moreover, we have proposed Hydra, which differs from Random Folded Clos networks on the possibility of combining random and non-random connectivity. This provides a more practical solution when considering current technologies.

We have algebraically proved the topological conditions for implementing a deadlock-free multi-path routing on such networks. They have been compared against three important counterparts: fat-tree (which is the standard indirect topology), OFT (which is a cost-optimal highly scalable indirect topology) and random regular graphs. The comparison has been made in terms of cost, scalability and expandability. Physical deployments for fat-trees, Random Folded Clos and Hydra networks have been also compared.

As it has been shown, Random Folded Clos and Hydra networks are as scalable and expandable as random regular graphs, but with the natural advantages of being a folded Clos topology. The presented theoretical analysis has been confirmed by an exhaustive simulation-driven experimentation. Compared towards the standard fat-tree, Random Folded Clos and Hydra networks are more scalable, allow for small incremental expansions and have comparable performance under typical traffic patterns. Besides, these networks can be built with a quite big save of cost, as they typically need a lower number of levels than fat-trees. This implies that our networks manage traffic with significant latency and energy savings. Importantly, Hydra networks admit a standard layout similar to the one used by fat-trees. In the light of all the results, it can be concluded that Hydra networks accomplish the requirements of nowadays datacenters and thus, are a realistic and practicable alternative to commodity fat-trees.

Acknowledgments

This work has been supported by the Spanish Science and Technology Commission (CICYT) under contract TIN2016-76635-C2-2-R and by the European Union's Horizon 2020 research and innovation program under grant Agreements No. 671697 (Mont Blanc 3) and No. 671610 (EuroLab-4-HPC). In addition, the costly experimental work in this paper has been carried out on Altamira and MareNostrum, supercomputing facilities hosted at the University of Cantabria and the Barcelona Supercomputing Center.

References

[1] Jung Ho Ahn, Nathan Binkert, Al Davis, Moray McLaren, and Robert S. Schreiber. HyperX: Topology, routing, and packaging of efficient large-scale networks. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, pages 1–11, New York, NY, USA, 2009. ACM.

[2] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, pages 63–74, New York, NY, USA, 2008. ACM, ACM.

[3] Noga Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, 1986.

[4] Leonid Alexandrovich Bassalygo and Mark Semenovich Pinsker. Complexity of an optimum nonblocking switching network without reconnections. *Problems of information transmission*, 9(1):64–66, November 1974. Transalated from Problemy Peredachi Informatsii.

[5] Maciej Besta and Torsten Hoefler. Slim Fly: A cost effective low-diameter network topology. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 348–359, Piscataway, NJ, USA, 2014. IEEE Press.

[6] Bartosz Bogdański. *Optimized Routing for Fat-Tree Topologies.* PhD thesis, University of Oslo, 2014.

[7] Béla Bollobás. The isoperimetric number of random regular graphs. *Eur. J. Comb.*, 9(3):241–244, May 1988.

[8] Béla Bollobás. *Random Graphs.* Cambridge studies in advanced mathematics, 2nd edition, 2001.

[9] C. Camarero, C. Martínez, E. Vallejo, and R. Beivide. Projective networks: Topologies for large parallel computer systems. *IEEE Transactions on Parallel and Distributed Systems*, 28(7):2003–2016, July 2017.

[10] Cristóbal Camarero, Carmen Martínez, and Ramó Beivide. Random folded Clos topologies for datacenter networks. In *Proceedings of the 23rd IEEE Symposium on High Performance Computer Architecture*, HPCA '17, pages 193–204, 2017.

[11] Dong Chen, Noel Eisley, Philip Heidelberger, Sameer Kumar, Amith Mamidala, Fabrizio Petrini, Robert Senger, Yutaka Sugawara, Robert Walkup, Burkhard Steinmacher-Burow, Anamitra Choudhury, Yogish Sabharwal, Swati Singhal, and Jeffrey J. Parker. Looking under the hood of the IBM Blue Gene/Q network. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 69:1–69:12, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[12] Tao Chen, Xiaofeng Gao, and Guihai Chen. The features, hardware, and architectures of data center networks. *J. Parallel Distrib. Comput.*, 96(C):45–74, October 2016.

[13] Charles Clos. A study of non-blocking switching networks. *Bell System Technical Journal, The*, 32(2):406–424, March 1953.

[14] William Dally and Brian Towles. *Principles and Practices of Interconnection Networks.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[15] Scott E Fahlman. The hashnet interconnection scheme. June 1980.

[16] Nathan Farrington, Erik Rubow, and Amin Vahdat. Data center switch architecture in the age of merchant silicon. In *Proceedings of the 2009 17th IEEE Symposium on High Performance Interconnects*, HOTI '09, pages 93–102, Washington, DC, USA, 2009. IEEE, IEEE Computer Society.

[17] Ikki Fujiwara, Michihiro Koibuchi, Hiroki Matsutani, and Henri Casanova. Swap-and-randomize: A method for building low-latency HPC interconnects. *Parallel and Distributed Systems, IEEE Transactions on*, 26(7):2051–2060, July 2015.

[18] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. Bcube: A high performance, server-centric network architecture for modular data centers. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pages 63–74, New York, NY, USA, 2009. ACM.

[19] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. Dcell: A scalable and fault-tolerant network structure for data centers. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, pages 75–86, New York, NY, USA, 2008. ACM.

[20] László Gyarmati and Tuan Anh Trinh. Scafida: A scale-free network inspired data center architecture. *SIGCOMM Comput. Commun. Rev.*, 40(5):4–12, October 2010.

[21] L. Y. Ho, J. J. Wu, and P. Liu. Optimal algorithms for cross-rack communication optimization in mapreduce framework. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 420–427, July 2011.

[22] Georgios Kathareios, Cyriel Minkenberg, Bogdan Prisacari, German Rodriguez, and Torsten Hoefler. Cost-effective diameter-two topologies: Analysis and evaluation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, pages 36:1–36:11, New York, NY, USA, November 2015. ACM.

[23] John Kim, William J. Dally, Steve Scott, and Dennis Abts. Technology-driven, highly-scalable dragonfly topology. In *Proceedings of the 35th Annual International Symposium on Computer Architecture*, pages 77–88. IEEE Computer Society, 2008.

[24] M. Koibuchi, I. Fujiwara, H. Matsutani, and H. Casanova. Layout-conscious random topologies for HPC off-chip interconnects. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 484–495, February 2013.

[25] Michihiro Koibuchi, Hiroki Matsutani, Hideharu Amano, D. Frank Hsu, and Henri Casanova. A case for random shortcut topologies for HPC interconnects. In *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA '12, pages 177–188, Washington, DC, USA, 2012. IEEE Computer Society.

[26] Vijay Lakamraju, Israel Koren, and C. Mani Krishna. Filtering random graphs to synthesize interconnection networks with multiple objectives. *Parallel and Distributed Systems, IEEE Transactions on*, 13(11):1139–1149, November 2002.

[27] Charles E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *Computers, IEEE Transactions on*, C-34(10):892–901, October 1985.

[28] Charles E. Leiserson, Zahi S. Abuhamdeh, David C. Douglas, Carl R. Feynman, Mahesh N. Ganmukhi, Jeffrey V. Hill, W.Daniel Hillis, Bradley C. Kuszmaul, Margaret A. St. Pierre, David S. Wells, Monica C. Wong-Chan, Shaw-Wen Yang, and Robert Zak. The network architecture of the connection machine cm-5. *Journal of Parallel and Distributed Computing*, 33(2):145–158, 1996.

[29] Hans Meuer, Erich Strohmaier, Jack Dongarra, Horst Simon, and Martin Meuer. Top500 supercomputer sites. `http://www.top500.org/lists/2017/06/`, June 2017.

[30] Javier Navaridas, José Miguel-Alonso, Jose Antonio Pascual, and Francisco Javier Ridruejo. Simulating and evaluating interconnection networks with INSEE. *Simulation Modelling Practice and Theory*, 19(1):494–515, 2011.

[31] Fabrizio Petrini and Marco Vanneschi. k-ary n-trees: high performance networks for massively parallel architectures. In *Parallel Processing Symposium, 1997. Proceedings., 11th International*, pages 87–93, April 1997.

[32] Ji-Yong Shin, Bernard Wong, and Emin Gün Sirer. Small-world datacenters. In *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, SOCC '11, pages 2:1–2:13, New York, NY, USA, 2011. ACM.

[33] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. Jupiter rising: A decade of Clos topologies and centralized control in Google's datacenter network. In *Sigcomm '15*, 2015.

[34] Ankit Singla, Chi-Yao Hong, Lucian Popa, and P. Brighten Godfrey. Jellyfish: Networking data centers randomly. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 17–17, Berkeley, CA, USA, 2012. USENIX Association.

[35] Angelika Steger and Nicholas C Wormald. Generating random regular graphs quickly. *Combinatorics, Probability and Computing*, 8(04):377–396, 1999.

[36] Eli Upfal. An $O(\log N)$ deterministic packet-routing scheme. *J. ACM*, 39(1):55–70, January 1992.

[37] Asaf Valadarsky, Gal Shahaf, Michael Dinitz, and Michael Schapira. Xpander: Towards optimal-performance datacenters. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies, CoNEXT 2016, Irvine, California, USA, December 12-15, 2016*, pages 205–219, 2016.

[38] Marcos Valerio, Louise E. Moser, and P. Michael Melliar-Smith. Recursively scalable fat-trees as interconnection networks. In *Computers and Communications, 1994., IEEE 13th Annual International Phoenix Conference on*, pages 40–46, April 1994.

[39] Leslie G. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11(2):350–361, 1982.

[40] Ye Yu and Chen Qian. Space shuffle: A scalable, flexible, and high-performance data center network. *IEEE Transactions on Parallel and Distributed Systems*, 27(11):3351–3365, November 2016.