Random Folded Clos Topologies for Datacenter Networks

Cristóbal Camarero Carmen Martínez Ramón Beivide
Computer Science and Electronics Department
University of Cantabria
{cristobal.camarero, carmen.martinez, ramon.beivide}@unican.es

ABSTRACT

In datacenter networks, big scale, high performance and fault-tolerance, low-cost, and graceful expandability are pursued features. Recently, random regular networks, as the Jellyfish, have been proposed for satisfying these stringent requirements. However, their completely unstructured design entails several drawbacks. As a related alternative, in this paper we propose Random Folded Clos (RFC) networks. They constitute a compromise between total randomness and maintaining some topological structure. As it will be shown, RFCs preserve important properties of Clos networks that provide a straightforward deadlock-free equal-cost multi-path routing and enough randomness to gracefully expanding. These networks are minutely compared, in topological and cost terms, against fat-trees, orthogonal fat-trees and random regular graphs. Also, experiments are carried out to simulate their performance under synthetic traffics that emulate common loads in datacenters. It is shown that RFCs constitute an interesting alternative to currently deployed networks since they appropriately balance all the important design requirements. Moreover, they do that at much lower cost than the fat-tree, their natural competitor. Being able up to connect the same number of compute nodes, saving up to 95% of the cost, and giving similar performance.

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 current warehouse-scale computers heavily rely on exploiting a high degree of parallelism. Some recent installations contain around 10,000 racks comprising tens of thousands servers [1]. Having to manage such amount of communicating components, the datacenter network (DCN) plays a critical and pivotal role. In addition, according to [2], bandwidth demands in the datacenter are doubling every 12–15 months.

Although there have been many recent proposals for DCN topologies such as Dcell [3], BCube [4], HyperX [5], Jellyfish [6], and Space Shuffle [7], the reality is that most modern datacenters are equipped with a folded Clos network. Clos topologies [8] were proposed more than sixty years ago and have been extensively used in telephony and high-performance computing. In particular, after the Leiserson's work [9] and the Connection Machine commercialization [10], a specific family of folded Clos networks has been known as *fat-tree networks*. More or less half of current supercomputers on the Top 500 list use fat-tree networks, or some of its variants [11]. Such non-blocking topologies can be built to an arbitrary scale using regular switches with affordable radices (number of ports). With the advent of merchant silicon [12], big companies have built their own fat-tree network fabrics, using general purpose merchant silicon instead of commercial switches. This allows to have custom packaging and networks protocols.

Topologies based on random regular graphs have been considered for interconnection networks. The Jellyfish topology was recently proposed in [6] for datacenter network design. This network is based on building a random regular graph in the top of 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 has been conceived as a direct network while indirect ones such as fat-trees are predominately used in the datacenter industry. Although extremely interesting, the Jellyfish could be considered as a quite long and disruptive step forward in the network design arena.

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 equal cost multi-path mechanisms.

This work explores an intermediate evolving step based on the natural idea of randomizing the interconnection pattern between switch layers of indirect folded Clos topologies; we have denoted them as *random folded Clos networks* (RFC). In Graph Theory, similar and related structures have been known long ago, mainly associated with the introduction of expander graphs [13, 14, 15]. In the same way that typical fat-trees, RFCs conserve a structure based on different stages of switches which, under certain conditions studied in this paper, allows for the existence of common ancestor switches for every pair of communicating servers (network terminals). RFCs leverage this property for providing a simple deadlock-free equal-cost multipath routing mechanism, identical to the

2378-203X/17 $31.00 © 2017 IEEE
DOI 10.1109/HPCA.2017.26

one employed in standard fat-trees. Moreover, as it will be shown, RFCs constitute a compromise among cost, performance, scalability, expandability and fault-tolerance among other indirect topologies.

When looking for efficient DCN solutions, it is critically important to know in advance the expected traffic to achieve judicious designs. There are multiple evidences concluding that datacenter traffic is mostly uniform [2, 16]. Thus, 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 networks proposed in this research follow this design principle. To show that, in this paper, such topologies are deeply studied to bring to light their outstanding properties and to prove that they deserve to be considered for forthcoming datacenter interconnection networks. Summarizing, the main achievements of this research are:

- To provide a unified definition and fast generation algorithms for RFCs.
- To characterize the diameter, the bisection bandwidth and the scalability of such topologies.
- To compare, in terms of cost, expandability and fault-tolerance, RFCs against other known topologies (including random and non-random ones).
- To provide an empirical performance evaluation of RFCs and compare them against to its natural competitor, *i.e.*, fat-trees, under different representative synthetic datacenter traffic patterns and in the presence of faults.

The remainder 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, provides an algorithm for their construction and a diameter, bandwidth and scalability characterization. Section 5 compares RFCs with other topologies such as fat-trees, orthogonal fat-trees and random regular graphs in terms of expansion. Section 6 presents the results of the experiments for performance evaluation. Section 7 addresses a resiliency study. Section 8 summarizes the main obtained achievements and future lines of research. The paper finishes with an appendix containing the algorithms for random graphs generation.

2. RELATED WORK

Although random graphs have been previously studied in the field of interconnection networks, the recently proposed Jellyfish topology [6] has motivated the present work. Ten years before the Jellyfish was introduced, Lakamraju et al. already proposed in [17] to randomly generate interconnection networks for parallel computer systems. 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. [18] 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. [19] complete this research by adding random swaps between links; although they mention the indirect network case, their methods and results are all focused on direct topologies.

Space Shuffle is another topology proposed for datacenter networks [7]. It connects routers in several cycles randomly, which allows for an easy non-minimal routing.

Scafida, an architecture for DCNs based on scale-free networks, reduces the average path lengths compared to other topologies with the same server numbers [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 [21]. These two architectures both employ random links. However, they require to manage the correlation among links, which is unknown when the networks expand. Another interesting and related unstructured topology is REWIRE [22].

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 patterns 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 [13] where random topologies for multi-stage 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 [14], 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 multiprocessor parallel computer systems.

Related to expander graphs, an interesting novel result is the one appeared in [23]. Their authors acknowledge that datacenter designs with 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 show that any expander topology (random graphs are just an example of the expander family) comes with these benefits. Notwithstanding, the suggested practical approaches to building such expander-based DCNs as well as the explicit construction employed, seem to be not clear enough and far for being applied in current datacenters.

3. FOLDED CLOS NETWORKS

Since the original article by Clos [8] many authors have dealt with such networks.

The present 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 [24]. Table 1 contains the notation used in this article when dealing with folded Clos networks.

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).

Table 1: Folded Clos Parameters.

DEFINITION 3.1. *A l-level folded Clos network is a network 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 $i-1$ 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 $1 \leq i < l$.*

In this paper, we are going to compare random folded Clos topologies against the popular fat-tree used in many high-performance clusters and modern datacenters. The name fat-tree [9] originally made reference to a tree in which the edges closer to the root are 'fatter'—using 2^i parallel cables 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 [25]. A remarkable introduction to fat-trees can be found in [26].

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*.*

The fat-trees considered in [27] are another interesting case. A *R-commodity fat-tree* (CFT) is a radix-regular fat-tree whose arities are all $R/2$ except $k_l = R$. These fat-trees have been called R-port l-trees in [28]. The present paper avoids the latter terminology since there are fat-trees with R ports other than the R-CFT. Figure 1 represents a CFT of four levels for switches of radix 4. It is important to see that, for the same number of levels, a CFT doubles the number of nodes of the k-ary l-tree.

Finally, let us introduce *Orthogonal fat-trees* (OFT) [29], 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 [30]. For their definition, let q be a power of a prime number. 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 attains a upper bound in the number of compute nodes for a given radix. Figure 2 shows a 2-level OFT. Minimal routes in the 2-level OFT are unique, which reduces worst-case performance.

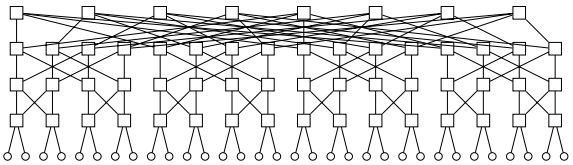

Figure 1: The 4-commodity fat-tree or 4-port 4-tree.

One of the advantages of folded Clos networks is the simplicity of their deadlock-free routing. If, for every pair of leaves, there is a path beginning with some up-links followed by the same number of down links, then using these paths provides a deadlock-free routing that does not require virtual channels. In such a case, the network is said to have an *up/down routing*. If a folded Clos network has an up/down routing, then its diameter fulfills $D \leq 2(l-1)$. However, the reciprocal sometimes is false.

Two important qualities that a network can have are *full bisection bandwidth* and *rearrangeably non-blocking* [24]. The *bisection bandwidth* of a network is the minimum bandwidth along all possible cuts of the network in two halves. 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. It is only quantitative and it depends on the routing to make an efficient use of it. For example, OFTs are essentially full bisection bandwidth, but this bandwidth is underused for many traffic patterns if minimal routing is used. In contrast, CFTs are rearrangeably non-blocking, which means they are able to manage at full speed communication patterns based on any node permutation; this, in turn, implies full bisection bandwidth. Thus, switching from a CFT to an OFT implies trading worst case performance and other properties for more scalability. Notwithstanding, there exist many networks that even not being full bisection bandwidth can route uniform traffic at full rate. This is the case, for example, of dragonflies [31] which are gaining momentum in the HPC arena. Such networks manage adverse traffic patterns by using Valiant random routing [32], which allows to achieve half of their maximum performance. The RFCs introduced in the next section behave, in this sense, better than dragonflies. They course at full rate uniform traffic while some adversarial traffic can be routed with much more than 50% performance, even without using any randomization mechanism, which reduces cost and complexity.

4. RANDOM FOLDED CLOS NETWORKS

Random graphs are widely known in the mathematical literature [33]. The direct interconnection networks based on

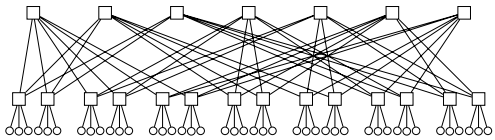

Figure 2: The 2-level orthogonal fat-tree.

random graphs proposed in [17] and [6] 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 [34]. They propose the following algorithm to generate random regular graphs of N vertices and degree Δ:

Algorithm

1. For each vertex, create Δ points. Initialize $U = \{1, \ldots, N\Delta\}$, the set of available points.
2. Repeat the following until no suitable pair can be found: Choose two random points i and j in U, and if they are suitable, pair i with j and delete them from U.
3. Create a graph G with edge from vertex r to vertex s if and only if there is a pair of points i, j with i belonging to r and j to s. If G is regular it is the output, otherwise return to step 1.

In the algorithm, a pair of points is suitable if it does not generate loops or multiedges in the graph. The resulting graphs will be generated with almost uniform probability. The implementation of step 2 given in [34] has expected complexity time $O(N\Delta^2)$. Listing 1 in the Appendix provides a new implementation written in Python programming language that performs better, in time $O(N\Delta \ln \Delta)$. An example of a random regular network (RRN) obtained with this algorithm can be seen in Figure 3.

It is known that a Δ-regular random graph with N vertices of diameter D can be easily obtained if $\Delta^D \approx 2N \ln N$, [33]; thus, $\Delta \approx (2N \ln N)^{1/D}$. Nevertheless, as our proposal is to force the random interconnection network to be a folded Clos, we continue formally defining them.

DEFINITION 4.1. *A random folded Clos (RFC) network with parameters $\mathscr{P}$ is a topology chosen randomly with uniform probability from all possible folded Clos networks with parameters $\mathscr{P}$.*

Remember that the set of parameters $\mathscr{P}$ in the previous definition are those listed in Table 1. As an example, the Hashnet interconnection network in [15] can be obtained by unfolding the RFC of previous definition 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 [13] and [14], although with more restrictions.

However, for practical reasons, in this paper our interest is focused on radix-regular RFCs. An example of such a interconnection network is represented in Figure 4. As it can be

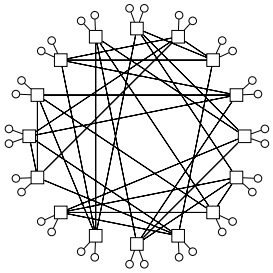

Figure 3: A random network with 16 routers of degree 4 and 2 compute nodes per router.

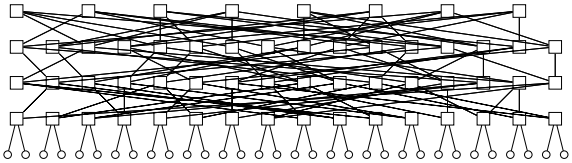

Figure 4: RFC of radix 4, $N_1 = 16$ and 4 levels.

seen, this topology is identical to a CFT, conserving its structure of switch levels, except for the interconnection pattern among levels which is random. Algorithm 2 in Appendix 9, derived from Algorithm 1, generates a random l-level folded Clos network by obtaining $l-1$ random bipartite graphs.

4.1 Up/Down Routing

As stated before, one important feature in a multi-stage network is the existence of an up/down routing, which is deadlock-free and can be easily implemented without virtual channels for breaking cycles. The following theorem, which represents one of the main findings of this work, states the probabilistic condition for the existence of such a routing on a RFC network.

THEOREM 4.2. *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 a common ancestor in a l-level R-radix regular random folded Clos network tends to $e^{-e^{-x}}$.*

PROOF. Let $\Delta = R/2$. The number of ancestors of a given leaf is $\Delta^{l-1}(1+o(1))$. Given two leaves a, b, the probability that they have disjoint set of ancestors is

$$Prob(anc(a) \cap anc(b) = \emptyset) = \frac{\binom{N_l - \Delta^{l-1}(1+o(1))}{\Delta^{l-1}(1+o(1))}}{\binom{N_l}{\Delta^{l-1}(1+o(1))}}$$
$$= (1+o(1)) \left(1 - \frac{\Delta^{l-1}(1+o(1))}{N_l}\right)^{\Delta^{l-1}(1+o(1))}.$$

Let λ be the expected number of pairs with disjoint ancestor set. Intuitively, the threshold must be around $\lambda \approx 1$. So

$$\lambda = (1+o(1)) \binom{N_1}{2} \left(1 - \frac{\Delta^{l-1}(1+o(1))}{N_l}\right)^{\Delta^{l-1}(1+o(1))},$$

and

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

Then,

$$\Delta = \left((1+o(1))N_l(\ln\binom{N_1}{2} + o(1) - \ln\lambda)\right)^{1/(2(l-1))}.$$

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 ancestor, so it equals $(1-\frac{\lambda}{\binom{N_1}{2}})^{\binom{N_1}{2}}$, which tends to $e^{-\lambda}$. □

Theorem 4.2 implies that $2(N_l \ln\binom{N_1}{2})^{1/(2(l-1))}$ is a sharp threshold for finding a RFC network with up/down routing. 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 a RFC with up/down routing. When $x = 0$, the equality $R = 2(N_1 \ln N_1)^{1/(2(l-1))}$ implies that the probability converges to a constant; so it is easy—after a few random graph generations—to build a RFC with up/down routing. In this case, $e^{-e^{-x}}$ becomes $1/e$ which means that, in average, a RFC with up/down routing is obtained every three times the algorithm in Listing 2 of Appendix 9 is executed. Small positive or negative values of x quickly impact on the probability of finding RFCs with up/down routing. For example, if it holds that $R = 2(N_1 \ln N_1 + \ln\ln N_1)^{1/(2(l-1))}$ then the probability of having common ancestors tends to 1 and if $R = 2(N_1 \ln N_1 - \ln\ln N_1)^{1/(2(l-1))}$, then the probability tends to 0. In most of the cases, RFCs close to this sharp threshold will be considered later on the paper. However, in other situations such as expanding a network or improving resiliency, RFCs having a switch radix contemplating a small positive value of x could be beneficial, as it will be shown in Sections 5 and 7.

4.2 Diameter and Bisection Bandwidth

As stated before, the diameter of a RFC with up/down routing is trivially upper bounded by $2(l-1)$. As a consequence of Theorem 4.2, 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. Any other construction, not fulfilling this equality, can be obtained expanding the original one without adding a new level, as it will be shown in Section 5. After further expansions, once the equality occurs again, it is necessary to add a new level. The expanded network will again connect the maximum number of compute nodes, but for the next possible diameter.

Figure 5 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 used in current commodity switches. The solid lines correspond to RRNs and RFCs, which can be implemented for any number of switches. They are rather similar with the major difference that the RFC can only have even diameters. The figure also indicates that random topologies admit an amount of compute nodes between the CFT and the OFT.

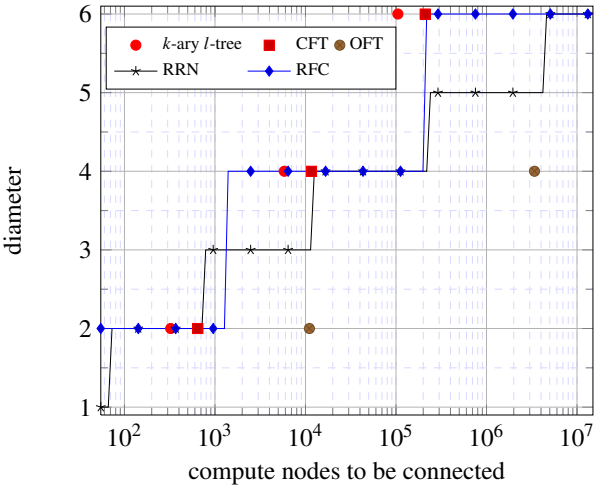

Figure 5: Diameter of RFC.

As an example to illustrate the resemblance between direct and indirect random networks, consider diameter 4 and radix 36. Theorem 4.2 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 sharing between network ports, Δ, and compute node 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 corresponds with 227,730 compute nodes, 12% above the RFC. The corresponding CFT of diameter 4, has 3 levels (as the RFC) but only 11,664 compute nodes.

In respect to the bisection width, random regular graphs have been known to be good expanders for a long time [13], [35]. This implies a good isoperimetric constant, or *Cheeger constant*, which in turn implies large bisection. Bollobás [36] proved that the isoperimetric number of a Δ-regular random graph is at least $\frac{\Delta}{2} - \sqrt{\Delta\ln 2}$. This implies that a RRN has a bisection width with 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 into two parts of the same size.

For a RFC it is possible to calculate such a value using the previous solution for the random regular case. Let S and $\bar{S}$ be the two halves of the worst possible—with least bandwidth—cut of the nodes into two parts of the same size. Now, identify the vertices into groups of 2 vertices of each level except 1 of level l. The resulting graph 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.

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 the CFT, each path traverses the bisection at most once regardless the traffic pattern. However, it is easy to see that in the RFC the average number of traversals is $l-1$. The CFT is always full-bisection bandwidth and thus has normalized bisection 1. Bollobas' bound for a RRN gives 0.88, for a 2-level RFC gives 0.80 and for a 3-level RFC gives 0.86. These three topologies perform at full rate under uniform traffic. However, there are adverse traffic patterns in which they perform slightly worst but, as it will be examined in Section 6, managing such more rare traffic at a very good rate.

4.3 Scalability

Current datacenter networks aim to reach large sizes, however if a bound is imposed in the diameter, they are limited by the number of ports in their switching chips. In this paper the term scalability is used as a measure of how many computing nodes (or servers) can be arranged in the network for a given router radix.

In respect to direct networks, as the average distance of a RRG is always a little below than its diameter, Δ/D compute nodes (a.k.a. servers, terminals) per switch are appropriate for a well balanced network . 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

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

Equivalently, 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 $R = 2(q+1)$, $N_1 = 2(q^2+q+1)^{l-1}$ leafs and $T = 2(q+1)(q^2+q+1)^{l-1}$ compute nodes, so $T \approx R\left(\frac{R}{2}\right)^{2(l-1)}$. Although its radix is $2(q+1)$, its arities, as a tree, are $k_1 = \cdots = k_{l-1} = q^2+q+1$ and $k_l = 2(q^2+q+1)$.

Figure 6 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 is a curve for each topology, for levels 2, 3 or 4, which implies different diameters. A first look at the graph confirms that, as expected, OFT shows the best scalability. Moreover, the l-level OFT scales at least as the CFT of level $l+1$. Concerning RFC, it clearly scales much better than the standard CFTs. Also, its scalability is really close to the RRN with the same diameter. Although OFTs are clearly more scalable, as it will be shown in the next section, the strict definition of this topology presents an important drawback that compromises its expandability.

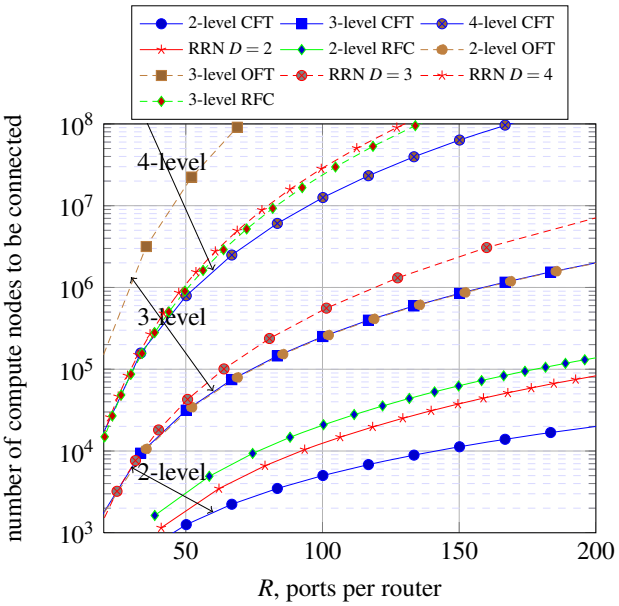

Figure 6: Scalability of RFC.

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 [6]. Therefore, given a network with N switches of radix R, its N'*-expandability* means the possibility of upgrading the current configuration to a network with $N+N'$ switches while preserving radix R. It is distinguished between *strong* or *weak* expandability, that is, if the expanded network maintains the same diameter or not, respectively.

The CFT is weak 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 also weak 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.

Conversely, a Δ-RRN on N switches can be easily upgraded into a Δ-RRN on N' switches, where approximately $\Delta(N'-N)/2$ links must be rewired. For maintaining the diameter D, it is necessary to maintain $\Delta \approx (2N'\ln N')^{1/D}$. Hence, a specific problem in the expansion of RRNs is the potential incorrect balance in the switches between links for terminals and links for connecting to other switches. As mentioned in Subsection 4.3, the proper number of compute nodes per switch is around Δ/D. Nevertheless, the number of compute 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 strong expandable since new switches can be added without increasing the levels of

the network. In this case, minimal upgrades add two new switches in every level and only one on the top level. This means that at each incremental expansion it is possible to add R new compute nodes. This can be done till achieving the threshold stated in Theorem 4.2. In other case, the up/down routing is not guaranteed. Nevertheless, if needed, this can be avoided by weakly expanding the network, that is, increasing the number of levels as other tree-based networks do.

In Figure 7, the network expandability against raw cost is represented. The number of compute nodes is represented in abscissas. The ordinates represent the total number of ports of the system, 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 result in different step functions, where each step means a weak expansion by adding a new level. Then, the constant part of the function gives the number of compute nodes which is possible to add without adding a new level. Both random topologies lead to almost linear functions, which reinforces that these topologies are really suitable for expansion, as it was already demonstrated in previous papers for RRNs. The small steps in the function for RFC corresponds to the addition of a level, that is, to the weak expansion to guarantee the up/down routing, as argued before. Note that the two random topologies have more or less the same cost. It is also worthwhile to note that, when updating 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 of the links.

Let us consider some examples to illustrate the differences between the expandability of CFTs and RFCs that will be also used for simulation in Section 6. With 3-levels and radix 36, the maximum size for a CFT is $T = 2(\frac{36}{2})^3 = 11,664$ (11K) compute nodes, distributed in 648 leaf switches. A RFC with equal resources (servers, ports, switches, levels and wires) can be easily built. However, a RFC with almost the same number of compute nodes can be implemented with 20-radix routers and the same number of levels and wires. In this RFC, the number of first level routers is 1,166 for a total of 11,660 compute nodes. Although this is a reduction in an important technological switch constraint, the radix, the resulting network costs are similar in terms of ports (or wires). The reverse also holds: with radix 36 we can implement a network of greater size for which a CFT would require greater radix or more levels.

The maximum expansion of a 3-level RFC with 36-radix switches has $2 \cdot 5627 \cdot 18 = 202,572$ (200K) terminals. A comparable 4-level CFT connects $2 \cdot 18^4 = 209,952$ compute nodes which is a little bit more than the RFC. The 4-level CFT uses 40,824 switches and 629,856 wires. The RFC uses 28,135 switches and 405,144 wires, leading to savings of 31% and 36% in switches and wires, respectively. Note that, beyond these terminal numbers, the RFC needs to be upgraded to 4 levels (to preserve the up/down routing) and the CFT to 5 levels.

From 11K to 200K compute nodes there are many intermediate cases; let us consider $2 \cdot 2778 \cdot 18 = 100,008$ (100K) compute nodes. Remember that, departing from the initial 3-level (11K) CFT configuration, the single addition of a compute node implies to upgrade the network levels. Hence, a 100K CFT needs 4-levels, leaving free ports for future expansion. Obviously, the RFC can be implemented with 3-levels. Remember that the 3-level RFC can be incrementally upgraded in steps of just 5 switches until arrive to the 200K case. Note that, although both networks connect the same number of compute nodes, the RFC uses far fewer resources. The 3-level RFC uses 13,890 switches and 200,016 wires to connect them. Although a fully-equipped 4-level CFT employs 40,824 switches and 629,856 wires, a convenient pruning could reduce such numbers at the expense of losing its full bisection capability. For this reason, the cost comparison is more difficult but, in any case, the savings for RFCs are superior than those of the maximum (200K) expansion case.

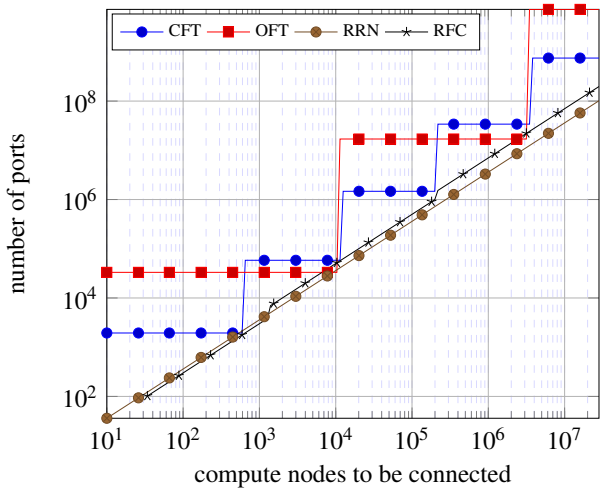

Figure 7: Expandability of RFC.

6. EMPIRICAL EVALUATION

As it has been seen in the previous sections, RRNs (*i.e.*, Jellyfish) and RFCs are similar in terms of scalability and expandability. However, the Jellyfish has some difficulties when its practical implementation is considered. On the one hand, as already stated in [6], the Jellyfish has a poor performance using the same minimal paths for communicating two nodes; thus, it is necessary to use k-shortest paths routing. This algorithm has a high computational cost, and should be executed when the network is modified by an expansion or a new fault. On the other hand, deadlock quickly arises in flow controlled or lossless direct networks if no mechanism for its avoidance is used. The RFC, as a multi-tree network, does not need any deadlock avoidance mechanism since it has up/-down routing. For these reasons, in our humble opinion, the Jellyfish (or RRN), is out of the natural competition between RFCs and CFTs.

The experiments for both RFC and CFT have been done using three synthetic traffic patterns, adapted from [37], that have been selected to represent typical application behaviour in a datacenter, which are:

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

Table 2: Simulation parameters

- *uniform*: Each generated packet has as destination a random compute node selected uniformly.
- *random-pairing*: The set of compute 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 is a case of a random permutation of the compute nodes.
- *fixed-random*: At the beginning, each compute node selects a different compute node in a random uniform way. During the simulation each node generate packets towards the selected compute node. It is not a permutation since several compute nodes can have selected the same destination.

Simulations have been done using INSEE (Interconnection Network Simulation and Evaluation Environment) [38] with the parameters shown in Table 2. Simulation consists on $10,000$ cycles for statistics, preceded by a network warmup. At least, 5 simulations are averaged for each point. Using up/down routing the networks are deadlock-free. Nevertheless, virtual lanes are used, to reduce Head of Line (HoL) blocking.

For the comparison, the three scenarios considered in the previous section, all of them for networks using routers of radix 36, will be simulated; remember they are:

1. Equal resources (11K): CFT and RFC have 3-levels, the first one with $N_1 = 648$ routers and the number of compute nodes is $11,664$.
2. Intermediate expansion (100K): 3-level RFC and 4-level CFT (with free ports for future expansion) with $100,008$ compute nodes.
3. Maximum expansion (200K): a 3-level RFC with $202,572$ compute nodes and a 4-level CFT with $209,952$ compute nodes.

The results from simulations for each scenario are shown, respectively, in Figures 8, 9 and 10. Note that such results are shown with normalized load, so 1 in the plots means that every compute node is injecting 1 phit per cycle. As it can be observed in the figures, under uniform traffic both topologies have almost the same performance, except for about 20% better average latency in those RFCs having less levels (second and third scenario). There is only an almost negligible degradation of the accepted load for the first scenario (11K), but remember that, as it was shown in Section 5, for this case the RFC can be made from 20-radix switches. In Figure 8 we have also shown the performance and latency for this network. As it can be seen in the figure, the 20-radix (11K) RFC has better performance than the 36-radix RFC, even having reduced the router size.

When considering random-pairing traffic, the effect of expanding the RFC is clearly reflected. In the first (11K) scenario, the RFC provides 88% of the performance given by CFT. This is not surprising since CFT is, by definition, a rearrangeably non-blocking network. In Figure 8 it can be seen that the small CFT accepts 0.86 of the theoretical maximum load while the RFC accepts 0.76. RFC, as we showed in Section 4, for radix 36, has a theoretical normalized bisection bandwidth of ≈ 0.86. This implies that HoL blocking has a slightly smaller effect on RFC's performance than on CFTs. It can also be observed that, as the random topology becomes wider, the performance is smoothly degraded when managing this supposedly less common traffic. While CFT has a penalty of 2.3% due to the increase of levels, RFC has a 11.84% worst performance in the 100K compute nodes scenario and 22.36% in the 200K compute nodes network. As a consequence of having less number of levels, the average latency improvements of RFC in second and third scenarios are around 15%.

Finally, if we consider the performance under fixed-random traffic, both topologies have practically identical throughput, which translates on that both tolerate equally well traffic generating hot spots. The average latency improvements of RFC in second and third scenarios are again in the surroundings of 15%.

$\approx T$	CFT	RRN	RFC	OFT
512	45.6%	45.6%	35.5%	–
1024	51.3%	49.0%	38.2%	21.6%
2048	56.3%	48.9%	40.7%	–
4096	61.7%	55.5%	43.5%	–
8192	65.3%	56.6%	44.0%	27.1%

Table 3: Faults to disconnect a diameter 4 network.

7. RESILIENCY

As a first measure of the fault-tolerance level exhibited by RFC networks, this section calculates the number of links that have to be removed in order to disconnect them. In Table 3 the percentage of links that must be removed to disconnect networks with diameter 4 (3 levels) is shown. This has been carried out in a similar way to [39]. These percentages have been calculated as an average of 100 different values, where the links are one by one randomly removed.

The first thing that attracts attention is the small percentages shown by the OFT, which appears to be the less fault-tolerant network. It is the low path diversity of OFT which makes it very sensitive to faults. Moreover, the radix of an OFT is, more or less, half of the other networks while deploying the same number of compute nodes. For example, when $T \approx 1024$, the OFT is built with $R = 8$, while the radix of both RRN and RFC is $R = 12$, and for the CFT is $R = 16$.

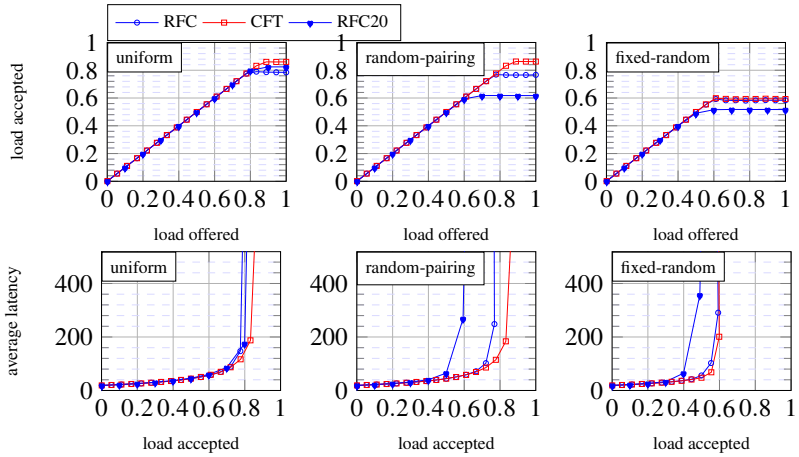

Figure 8: Simulated latency and throughput of 3-level CFT and RFC with radix 36 and 11664 compute nodes.

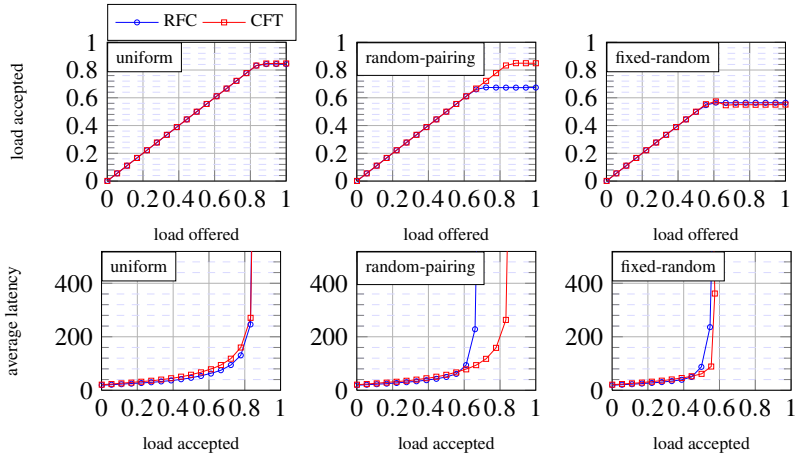

Figure 9: Simulated latency and throughput of 3-level RFC and 4-level CFT with radix 36 and 100,008 compute nodes.

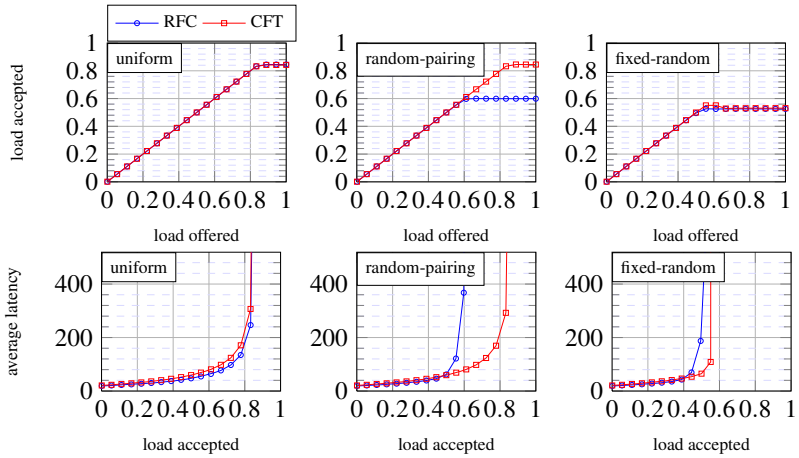

Figure 10: Simulated latency and throughput of 3-level RFC and 4-level CFT with radix 36. RFC has 202,572 and CFT has 209,952 compute nodes.

Therefore, there is a correspondence between the radix and the fault-tolerance: OFT gets disconnected by removing half of the links to disconnect a CFT, but it also has half of its radix.

RRNs and CFTs present similar percentages. Also, the RFC is not very far from these ones (the percentages differ 10% in average). However, again there is an important difference between the radix needed to build the network. For example, when $T \approx 2048$ the CFT needs $R = 20$ switches, while the RRN and RFCs use $R = 13$ and $R = 14$ switches, respectively, which is 30% less connectivity due to the smaller radix.

As it has been highlighted before, an important feature of folded Clos networks is the possibility of up/down routing. Thus, a second measure of fault-tolerance is the capability of performing such routing under faults. This ability is, in some cases, quickly lost. Take, for example, the 2-level OFT, where up/down paths are unique. The removal of any edge results in many pairs of leaves without any up/down route. For OFTs with more levels, in spite of having multiple paths for many pairs, the situation is similar.

Theorem 4.2 indicates that taking $R \approx 2(N_1 \ln N_1)^{1/(2(l-1))}$ is enough to easily obtain RFCs equipped with up/down routing. However, this could imply a lower tolerance to link removals. If a RFC network with up/down routing able to tolerate a proportion p of faulty links is wanted, then a greater radix value must be used. This can be seen as a way of trading scalability for fault-tolerance. In contrast, both the CFT and OFT have fixed tolerance. Figure 11 aims to illustrate this situation. For radix $R = 12$, the graph shows the fraction of broken links in which up/down routing is possible for different topologies. In abscissas the number of compute nodes is represented and, in ordinates, the percentage of tolerated broken links. The three lines in the figure correspond to the empirically calculated fault-tolerance of a RFC for 2, 3, and 4 levels. The three isolated points with 0 tolerance (cutting the X axis) correspond to OFTs and the other three isolated points to the CFTs. It can be seen that, for 3 and 4 level networks, the CFT points are below the RFC's ones. This means that it is possible to build a RFC with the same radix and size as the CFT, but with more fault-tolerance. Or conversely, building a RFC with the same fault tolerance and more compute nodes.

This can be illustrated using the first scenario in Section 6. For same resources (routers, cables and compute nodes) the CFT and the RFC give similar performance and Figure 11 shows that the RFC tolerates more faulty links before losing up/down routing. Furthermore, Figure 12 shows the progression in performance when links become faulty. For that, links have been randomly removed in multiples of 300 of a total of 23,328 wires. Then, the maximum throughput is shown for that amount of random failures. The curves show that the small difference in throughput between the topologies quickly vanishes. Indeed, at some point around 3,000 faulty links, which constitute the 12% of the wires, the situation reverses and the RFC gives slightly better performance. A curious observation is that under fixedrandom and random-pairing traffics the tolerated amount of failures is greater than under uniform traffic. This is caused because in uniform traffic any pair of leaves without common ancestor is enough

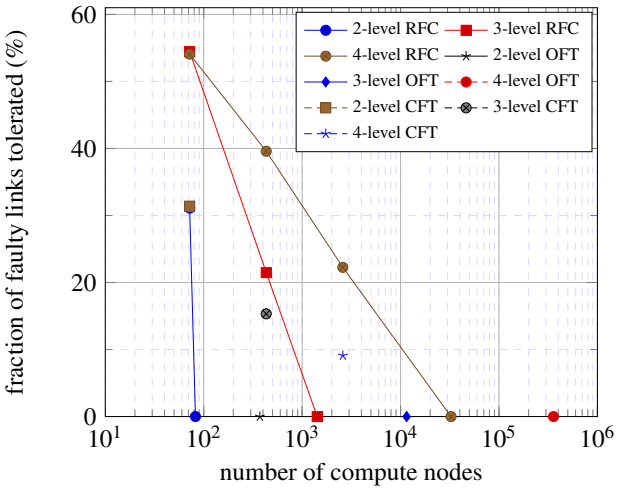

Figure 11: Fault-tolerance preserving up/down routing.

to block the network, while in the other traffics more specific pairs without common ancestor are needed.

Finally, it is important to stand out that well structured networks lose important properties in faulty scenarios. For example, a CFT with faults stops being rearrangeably non-blocking (it is no longer full bisection). In the case of the OFT, it happens something worst: a few links removed cause deadlock with the standard routing algorithm. However, random topologies (both RRNs and RFCs) are robust in the presence of faults in the sense that they remain being random after faults.

8. CONCLUSIONS

Random multi-stage topologies were proposed in the late 70's. In this paper we have analyzed and characterize a specific case, denoted as random folded Clos networks, which is valuable for current datacenter network design. We have algebraically proved the conditions for implementing a deadlock-free equal-cost multi-path routing on such networks. They have been compared against three important figures: 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, expandability and fault-tolerance. As it has been shown, random folded Clos networks are as scalable and expandable as random regular graphs, but with the natural advantages of being a folded Clos topology. Moreover, random folded Clos networks are highly fault-tolerant, preserving the up/down routing in extreme conditions. Compared towards the standard fat-trees, random folded Clos networks are more scalable, allow for small incremental expansions and have similar fault-tolerance. Besides, these can be done with a quite big save of cost. In addition, the lost of rigidity in its random definition enables to tune up all these parameters according to a design goal. The theoretical analysis presented has been confirmed by an exhaustive experimentation considering both fault-free and faulty scenarios.

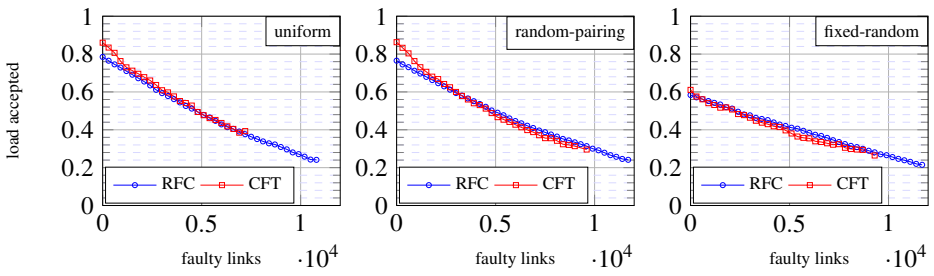

Figure 12: Simulated throughput of 3-level CFT and RFC with radix 36 and 11664 compute nodes under faults.

Acknowledgments

This work has been supported by the Spanish Science and Technology Commission (CICYT) under contract TIN2016-76635-C2-2-R, the European Union's Horizon 2020 research and innovation program under grant Agreement No 671697 (Mont Blanc 3) and the European HiPEAC Network of Excellence.

9. APPENDIX

The two algorithms for generating regular random and bipartite random graphs referred in Section 4 are presented below.

Listing 1: Generation of a random graph

```
def random_graph(n,d):
  #Generate a random regular graph with n vertices and
       degree d.
  go=True
  while go:
    go=False
    U=range(n*d)
    adj=[set() for k in range(n)]
    A=set(range(n))
    while len(U)>0:
      if len(A)<d:
        #Check if there is at least one available edge.
        good=False
        for a in A:
          for b in A:
            if a!=b and a not in adj[b]:
              good=True
        if not good:
          go=True
          break
      while True:
        r=random.randint(0,len(U)-1)
        U[r],U[-1] = U[-1],U[r]
        r=random.randint(0,len(U)-2)
        U[r],U[-2] = U[-2],U[r]
        #U ends with two random points.
        u=U[-1]/d
        v=U[-2]/d
        if u!=v and v not in adj[u]: break
      del U[-1]
      del U[-1]
      adj[u].add(v)
      adj[v].add(u)
      for w in [u,v]:
        if len(adj[w])==d:
          A.remove(w)
  return adj
```

This algorithm can be adapted to generate RFCs. To generate a random l-level RFC we propose to make $l-1$ random bipartite graphs using the implementation in Listing 2.

Listing 2: Generation of a random bipartite graph

```
def random_bipartite(n1,d1,n2,d2):
  #Generate a random bipartite graph with n1+n2 vertices
       with the n1 vertices having degree d1 and the n2
       vertices having degree d2.
  go=True
  while go:
    go=False
    U1=range(n1*d1)
    U2=range(n2*d2)
    adj1=[set() for k in range(n1)]
    adj2=[set() for k in range(n2)]
    A1=set(range(n1))
    A2=set(range(n2))
    while len(U1)>0:
      if len(A1)<d1 and len(A2)<d2:
        good=False
        for a in A1:
          for b in A2:
            if b not in adj1[a]:
              good=True
        if not good:
          go=True
          break
      while True:
        r=random.randint(0,len(U1)-1)
        U1[r],U1[-1] = U1[-1],U1[r]
        r=random.randint(0,len(U2)-1)
        U2[r],U2[-1] = U2[-1],U2[r]
        u=U1[-1]/d1
        v=U2[-1]/d2
        if v not in adj1[u]: break
      del U1[-1]
      del U2[-1]
      adj1[u].add(v)
      adj2[v].add(u)
      if len(adj1[u])==d1:
        A1.remove(u)
      if len(adj2[v])==d2:
        A2.remove(v)
  return adj1,adj2
```

THEOREM 9.1. *The expected time for each iteration in the implementations in Listings 1 and 2 is $O(N\Delta \ln \Delta)$, where $N = \max(N_1, N_2)$ and $\Delta = \max(\Delta_1, \Delta_2)$ in the latter.*

10. REFERENCES

[1] T. Chen, X. Gao, and G. Chen, "The features, hardware, and architectures of data center networks: A survey," *Journal of Parallel and Distributed Computing*, 2016. Accepted for publication.

[2] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat, "Jupiter rising: A decade of Clos topologies and centralized control in Google's datacenter network," in *Sigcomm '15*, 2015.

[3] 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 *Proceedings*

of the ACM SIGCOMM 2008 Conference on Data Communication, SIGCOMM '08, (New York, NY, USA), pp. 75–86, ACM, 2008.

[4] 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 *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, (New York, NY, USA), pp. 63–74, ACM, 2009.

[5] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. 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, (New York, NY, USA), pp. 1–11, ACM, 2009.

[6] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, (Berkeley, CA, USA), pp. 17–17, USENIX Association, 2012.

[7] Y. Yu and C. Qian, "Space shuffle: A scalable, flexible, and high-performance data center network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, pp. 3351–3365, Nov. 2016.

[8] C. Clos, "A study of non-blocking switching networks," *Bell System Technical Journal, The*, vol. 32, pp. 406–424, Mar. 1953.

[9] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *Computers, IEEE Transactions on*, vol. C-34, pp. 892–901, Oct. 1985.

[10] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, W. Hillis, B. C. Kuszmaul, M. A. S. Pierre, D. S. Wells, M. C. Wong-Chan, S.-W. Yang, and R. Zak, "The network architecture of the connection machine cm-5," *Journal of Parallel and Distributed Computing*, vol. 33, no. 2, pp. 145–158, 1996.

[11] H. Meuer, E. Strohmaier, J. Dongarra, H. Simon, and M. Meuer, "Top500 supercomputer sites." `http://www.top500.org/lists/2014/11/`, Nov. 2014.

[12] N. Farrington, E. Rubow, and A. 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, (Washington, DC, USA), pp. 93–102, IEEE, IEEE Computer Society, 2009.

[13] L. A. Bassalygo and M. S. Pinsker, "Complexity of an optimum nonblocking switching network without reconnections," *Problems of information transmission*, vol. 9, pp. 64–66, Nov. 1974. Transalated from Problemy Peredachi Informatsii.

[14] E. Upfal, "An $O(\log N)$ deterministic packet-routing scheme," *J. ACM*, vol. 39, pp. 55–70, Jan. 1992.

[15] S. E. Fahlman, "The hashnet interconnection scheme," June 1980.

[16] 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*, pp. 420–427, July 2011.

[17] V. Lakamraju, I. Koren, and C. M. Krishna, "Filtering random graphs to synthesize interconnection networks with multiple objectives," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 13, pp. 1139–1149, Nov. 2002.

[18] M. Koibuchi, H. Matsutani, H. Amano, D. F. Hsu, and H. Casanova, "A case for random shortcut topologies for HPC interconnects," in *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA '12, (Washington, DC, USA), pp. 177–188, IEEE Computer Society, 2012.

[19] I. Fujiwara, M. Koibuchi, H. Matsutani, and H. Casanova, "Swap-and-randomize: A method for building low-latency HPC interconnects," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 26, pp. 2051–2060, July 2015.

[20] L. Gyarmati and T. A. Trinh, "Scafida: A scale-free network inspired data center architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 4–12, Oct. 2010.

[21] J.-Y. Shin, B. Wong, and E. G. Sirer, "Small-world datacenters," in *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, SOCC '11, (New York, NY, USA), pp. 2:1–2:13, ACM, 2011.

[22] A. R. Curtis, T. Carpenter, M. Elsheikh, A. López-Ortiz, and S. Keshav, "REWIRE: An optimization-based framework for unstructured data center network design," in *INFOCOM, 2012 Proceedings IEEE*, pp. 1116–1124, Mar. 2012.

[23] A. Valadarsky, M. Dinitz, and M. Schapira, "Xpander: Unveiling the secrets of high-performance datacenters," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, HotNets-XIV, (New York, NY, USA), pp. 16:1–16:7, ACM, 2015.

[24] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[25] F. Petrini and M. Vanneschi, "*k*-ary *n*-trees: high performance networks for massively parallel architectures," in *Parallel Processing Symposium, 1997. Proceedings., 11th International*, pp. 87–93, Apr. 1997.

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

[27] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, (New York, NY, USA), pp. 63–74, ACM, ACM, 2008.

[28] X.-Y. Lin, Y.-C. Chung, and T.-Y. Huang, "A multiple LID routing scheme for fat-tree-based InfiniBand networks," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, pp. 11–20, Apr. 2004.

[29] M. Valerio, L. E. Moser, and P. M. Melliar-Smith, "Recursively scalable fat-trees as interconnection networks," in *Computers and Communications, 1994., IEEE 13th Annual International Phoenix Conference on*, pp. 40–46, Apr. 1994.

[30] G. Kathareios, C. Minkenberg, B. Prisacari, G. Rodriguez, and T. 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, (New York, NY, USA), pp. 36:1–36:11, ACM, Nov. 2015.

[31] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*, pp. 77–88, IEEE Computer Society, 2008.

[32] L. G. Valiant, "A scheme for fast parallel communication," *SIAM Journal on Computing*, vol. 11, no. 2, pp. 350–361, 1982.

[33] B. Bollobás, *Random Graphs*. Cambridge studies in advanced mathematics, 2nd ed., 2001.

[34] A. Steger and N. C. Wormald, "Generating random regular graphs quickly," *Combinatorics, Probability and Computing*, vol. 8, no. 04, pp. 377–396, 1999.

[35] N. Alon, "Eigenvalues and expanders," *Combinatorica*, vol. 6, no. 2, pp. 83–96, 1986.

[36] B. Bollobás, "The isoperimetric number of random regular graphs," *Eur. J. Comb.*, vol. 9, pp. 241–244, May 1988.

[37] D. Chen, N. Eisley, P. Heidelberger, S. Kumar, A. Mamidala, F. Petrini, R. Senger, Y. Sugawara, R. Walkup, B. Steinmacher-Burow, A. Choudhury, Y. Sabharwal, S. Singhal, and J. 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, (Los Alamitos, CA, USA), pp. 69:1–69:12, IEEE Computer Society Press, 2012.

[38] J. Navaridas, J. Miguel-Alonso, J. A. Pascual, and F. J. Ridruejo, "Simulating and evaluating interconnection networks with INSEE," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, pp. 494–515, 2011.

[39] M. Besta and T. 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, (Piscataway, NJ, USA), pp. 348–359, IEEE Press, 2014.