Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Doctoral School of Computer Science

Efficient Cabling in Data Center Networks

Márton Csernai
MSc in Electrical Engineering

Summary of PhD Dissertation

Advisor:

Dr. András Gulyás, PhD

High Speed Networks Lab
Dept. of Telecommunications and Media Informatics and
MTA-BME Information System Research Group,
Budapest University of Technology and Economics

Budapest, Hungary
2015.

1 Introduction

Enormous amount of data is accumulating these days. As the primary goal of constant accumulation of data sets, large-scale information retrieval poses new challenges for traditional data processing systems. The ever-growing demand for computing and storage capacity led to the recent birth of cloud computing architectures, where users are provided with the illusion of infinite resources. In reality, the cloud is a complex ecosystem of various hardware resources and plethora of software that is accessible over the internet for anyone without upfront costs. The "cloud" generally incorporates a complex computer architecture that is commonly abstracted away for everyday users. Such computer systems are called data centers (DCs), which have to scale up to previously unseen proportions. Hundreds of thousands of servers in one DC is already the norm for large providers, and this number is predicted to grow significantly over the next few years. In parallel to migrating to huge public clouds, more and more organizations decide to consolidate their computing resources into private clouds, as they provide a high level of security and privacy for organizations. As a consequence, the increasing proliferation of both small and large data centers are highly likely.

Amongst the many challenges, the management of cabling in data centers is a complex task [1]. Appropriate cabling management can minimize downtime, maximize space use, and reduce operational costs [4]. State of the art DC architectures adapt sophisticated supercomputer interconnection networks (e.g., fat trees, hypercubes) that have been optimized for high performance since their first appearance. However, the expansion of such architectures can only be done in large steps, which involves large investment costs. Furthermore, the *expansion of the rigid structure requires a large amount of cables to be rewired*, which is a laborious and error-prone task, which in turn raises the costs of the expansion process. Recent works have already addressed the issue of *incremental expansion* in data centers to a limited extent [6, 5, 10, 19]. In general, the proposed architectures devise asymmetrical topologies that complicate the cable management of such systems (e.g., link reparation following a failure). Moreover, *the control plane of existing proposals rely on global topology information* and constantly maintaining such topology data leads to an unfavorable and possibly intolerable overhead in case of large DCs. On the other hand, large-scale data centers can have hundreds of kilometers of cables deployed, among which long optical fibers can span more than hundred meters [11]. Reducing *cabling complexity*, defined as *the number of long inter-rack cables* [9], is an important, yet unsolved and under illuminated issue regarding current DC networks.

My dissertation contains the most important results of my research work presenting novel methods for incremental expansion and cabling complexity reduction in DCs. The dissertation is structured around the following three Thesis Groups:

- **Thesis Group 1**: Definition and analysis of an incrementally expandable interconnection network based on hyperbolic tessellations.
- **Thesis Group 2**: Design and analysis of a hyperbolic DC architecture that supports incremental expansion and low complexity control plane implementations.
- **Thesis Group 3**: Cabling complexity reduction in state of the art data center architectures.

2 Research Goals

In the first part of my dissertation, I aim at designing an interconnection architecture that is both easily expandable and efficiently navigable. The main research objective in Sec. 4.1 is to find an appropriate structure that is both suitable for incremental expansion, and also supports a local navigational mechanism, i.e., greedy geographic routing. Inspired by recent theoretical findings in hyperbolic geometry and complex networks [16], I propose a novel interconnection network based on hyperbolic tessellations. These hyperbolic structures generalize regular lattice graphs, which thus acquire the property of exponential expansion inherently present in hyperbolic spaces. Importantly, the proposed structure also possesses favorable distributed and low-overhead routing capability thanks to greedy forwarding. Moreover, my novel interconnection network is designed to adhere to low diameter and high path diversity, because these two topological properties are respective primary indicators of latency and throughput in a distributed system.

The second research goal is to augment my proposed hyperbolic interconnection structure such that it is applicable for an incrementally expandable, high throughput data center network. In Sec. 4.2, my aim is to meet specific operational requirements (e.g., multipath routing, load balancing, fault tolerance, cost effectiveness) of today's DCs, while keeping the complexity of routing and cabling management as low as possible. The proposed system must be designed to support incremental plug & play upgradability, in case of both servers and switches, in a cost and cabling efficient manner.

The main research objective in Sec. 4.3 is to find an appropriate method, which reduces the cabling complexity (i.e., the number of long inter-rack cables) in high performance data center networks. The goal is to achieve cabling reduction results beyond topology optimization techniques [18] without increasing the control plane overhead [15]. The proposed solution builds on current optical telecommunication techniques, e.g., dense wavelength division multiplexing (DWDM) and the application of passive optical switching devices, i.e., arrayed waveguide grating routers (AWGR). The goal is to determine the technical and economical feasibility of the application of these technologies in the data center, depending on price variability of required optical components.

3 Methodology

I extensively use *graph theoretical analysis* in the process of design and evaluation of the proposed systems and methods. The design and analysis of my proposed incrementally expandable interconnection architecture in Sec. 4.1 are obtained through analytical methods from *hyperbolic geometry*. I use both analytical and numerical methods to confirm my results regarding the topological properties (i.e., diameter, path diversity) of the proposed structure. I validate the throughput performance of the proposed hyperbolic DC architecture in Sec. 4.2 through a flow level traffic *simulation* tool that I developed in C++ and R Statistical Language. In Sec. 4.3, I construct a multi-input, highly *integrated techno-economical model*, in which I evaluate the economical aspects of my proposed cabling reduction methods. The cost considerations throughout my dissertation are based on the economics of real-world data centers (e.g., costs of equipment, labor, etc.) and the *technological specifications of the equipments* involved in the analysis.

4 New Results

4.1 Expandable Hyperbolic Interconnection Network

The processing elements in a distributed and interconnected system, e.g., thousands of servers in a data center, exchange data with each other to execute a given task. The incurred control and data communication is facilitated by a high performance interconnection network. A tree is a very cost effective interconnection structure when routing has to be solved on a population of network nodes. On the downside, trees cannot ensure path diversity and high throughput, which are two key requirements for interconnection systems. Recently, several augmentations of trees have been proposed to overcome these limitations. A very popular method increases the number of root nodes (i.e., fat tree) to provide multiple paths, large bisection bandwidth, and no single point of failure [3]. As a common drawback of such approaches, when their topology is gradually expanded, these interconnection structures have to be completely rewired on a recurring basis (Fig. 1).

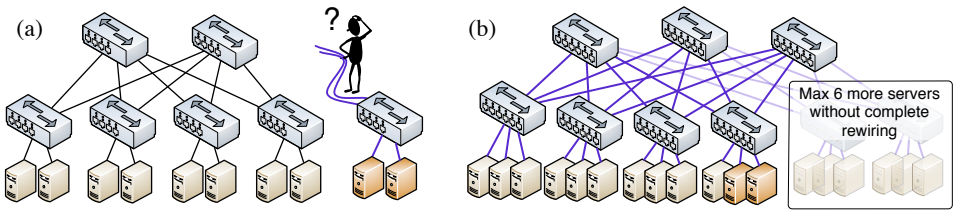


Figure 1: (a) The biggest 2-level fat tree structure that can be built with 4-port switches. (b) When we want to add additional servers, then the arity of the tree has to be increased, so all switches have to be replaced to maintain the defined structure.

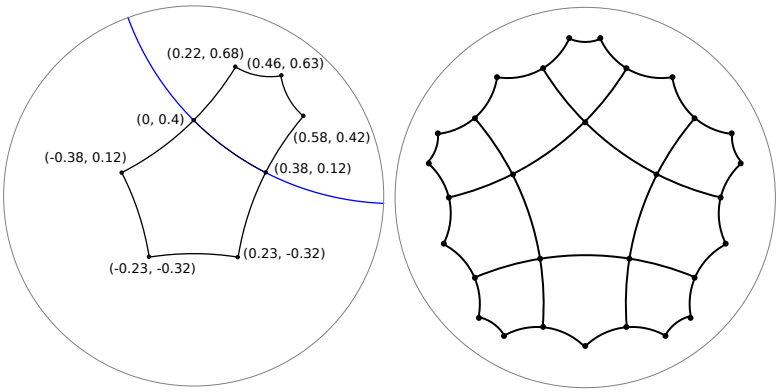
To address this problem, I looked for a regular topology that can be incrementally expanded. The main goal is to construct a structure where such states in the growth process are avoided, when the whole structure has to be completely rewired. Moreover, the proposed structure has to provide high performance with low control plane overhead.

Thesis Group 1. *I have proposed a hyperbolic interconnection network that is incrementally expandable without the recurrent need for complete rewiring. I have showed that it has logarithmically growing diameter, it supports optimal greedy geographic routing, and by link densification, its latency is reduced and its throughput can be fine tuned.*

Recent findings show that complex networks have hidden metric structure, that can be well represented in hyperbolic (non-Euclidean) geometry, and this metric structure is highly efficient for greedy routing [16]. Inspired by these theoretical results, my proposed interconnection network is based on uniform tessellations of the hyperbolic plane (*Sec. 3.2.1 and 3.2.2*).

Thesis 1.1. *[C3, J1] I have proposed a method to create an efficiently expandable interconnection topology, called* ***hyperbolic tessellation topology*** *(**HTT**). I have shown (by giving a formula) that the diameter of HTT grows logarithmically with the number of nodes, and greedy routing always finds paths between its vertices.*

In general, a regular tessellation is a tiling (or covering) of the plane by congruent regular polygons such that the same number of polygons meet at each vertex. An (n, k) regular tessellation contains regular n-gons, from which k meet at every vertex to fill the plane without overlaps or gaps. On the Euclidean plane, there exist only three kinds of tiling, $(3, 6)$, $(4, 4)$, and $(6, 3)$ because the sum of angles meeting at one vertex must be exactly 360°. However, on the hyperbolic plane, there exist infinitely many (n, k) tilings[1] because there the angles of a regular n-gon could be arbitrary small. Regular hyperbolic tessellations provide important advantageous properties. On one hand, they are inherently embedded in hyperbolic space that enables efficient greedy routing. On the other hand, they allow virtually infinite incremental expansion without impacting existing parts in the core of the structure.



(a) (b)

Figure 2: Construction of a (5,4) hyperbolic tessellation. (a) Initial polygon is reflected about one of its own side to get a polygon of the new layer. (c) Remaining area around all initial vertices tiled with polygons to finish the new layer.

During my HTT topology construction, I create uniform hyperbolic tessellations in the Poincaré disk model of the hyperbolic plane. In this model, an (n, k) **regular hyperbolic tessellation** is constructed recursively from regular n-gons through reflections on the sides of the polygons (Fig. 2). During this process, the coordinates of nodes are computed and stored in a list L, which is sorted in increasing order of the distance of the nodes' location (x_i, y_i) from the center of the disk $(0, 0)$. Importantly, if we consider the vertices of polygons as nodes and the sides of polygons as links, then we may consider the hyperbolic tessellation as a topology embedded into the hyperbolic plane. Although we can continue the construction process to infinity, we simply generate the coordinates of a *sufficiently* large tessellation.

The HTT structure employs the simple **greedy geographic routing** paradigm (GR), in which no routing states have to be kept in switching devices. Here, routing decision is solely based on metric distances between neighbor nodes $v_i(v_{i,1}, v_{i,2})$ of a current node u and the destination node $t(t_1, t_2)$

$$d(v, t) = \operatorname{arccosh}\left(1 + 2\frac{(v_1 - t_1)^2 + (v_2 - t_2)^2}{(1 - v_1^2 - v_2^2)(1 - t_1^2 - t_2^2)}\right), \tag{1}$$

[1]There exist an (n, k) tiling on the hyperbolic plane, if $\frac{1}{n} + \frac{1}{k} < \frac{1}{2}$ [8].

which is defined as the distances between nodes in the hyperbolic Poincaré disk model. Following the simple greedy rule, an intermediate node on the forwarding path then always forwards a packet to its neighbor that is closest to the destination. Through analytical reasoning, I have shown that inside an infinite HTT greedy routing can always find a next hop towards any destination. In short, I indirectly assume that two arbitrary nodes of the tessellation cannot reach each other through greedy forwarding, and conclude that, in this case, the destination node is not a vertex of the tessellation.

One can note that the simplicity of the control plane is traded off for the introduced computational requirement in the data plane. I argue that this is a justified choice in the design. Since arccosh() is monotone, this operation can be left out of the calculation for boosting forwarding performance. Hence, the required computation at each forwarding decision is reduced to about a dozen of simple arithmetic operations which consumes reasonably few CPU cycles. By default, nodes are required to calculate the next-hop distances for each packet to be forwarded. To further improve forwarding performance, routes can be cached by intermediate switches, and flow labels can be used for per-flow forwarding decisions. This trade-off was demonstrated in a working greedy routing environment. I emphasize that HTT retains all advantages of greedy routing, thus, there is no link state propagation protocol prevalently used in DC architectures. This routing mechanism does not require carefully adjusted routing tables and implements routing with in essence zero messaging overhead.

A key parameter of interconnection networks is the **diameter** of the underlying topology, i.e., the longest of shortest paths measured between all node pairs, because it is the primary metric of latency in the network. I have shown that the diameter of HTT grows logarithmically with the number of nodes. The proof contains the following steps:

- v_m is the num. of vertices that belong to layer m, and do not belong to layer $m-1$.
- v_m can be recursively calculated for any n (n vertices in each polygon).
- $v_m \approx n\left((n-2)(k-2)-1\right)^m, n > 3$ (there is a similar formula for $n = 3$).
- Since the diameter grows linearly with the number of layers, but the number of nodes grows exponentially with the number of layers, this means that the diameter grows logarithmically with the number of nodes.

A great advantage in HTT is that greedy routing always finds the **shortest path** between arbitrary node pairs, when no link or node failure occurs (*Sec. 3.2.3*).

Thesis 1.2. *[C3, J1] I have shown that greedy routing always finds the shortest paths between arbitrary node pairs in HTT, assuming there are no failures in the topology.*

The optimal performance of greedy routing on HTT can be proven mainly by following the simple rule of greedy routing and that the HTT structure is reflection symmetrical. The proof contains a mathematical induction, where, assuming that greedy routing can find the shortest path for any path that is of length $\leq l$, I show it to be true for $l + 1$ as well. By following the induction, we can conclude that greedy routing always finds a shortest path between source and destination nodes.

Finite size effects. In case of finite topologies, there may be source-destination node pairs on the perimeter of the tessellation, for which GR fails. This is simply due to our

incremental expansion requirement. Since we may add one node at a time, there can be polygons in the outermost layer of the tessellation, which are not yet complete.

According to our construction method, we fill up the topology with nodes in an ascending order based on their distance from the center. Fig. 3a shows a snapshot of the (5,4) HTT evolution with 7 nodes, where the 6th and 7th nodes are s and t respectively. Now, let us assume that GR tries to find a path from s to t. Since node s has only one neighbor node c, and since $d(c,t) > d(s,t)$, GR will fail at node s. It may be inferred from the figure, that if node v were present in the topology, GR would find it as the next hop towards node t. In case of such situations, I propose to connect nodes, which cannot reach each other through greedy forwarding (i.e., we add link $\{s,t\}$ to the topology as seen on Fig. 3b). On the other hand, after we have added node v to the topology, we may remove the additional link $\{s,t\}$ to free up network resources (e.g., ports, cables) as depicted on Fig. 3c. The complete incremental expansion method is formalized in Algorithm 1.

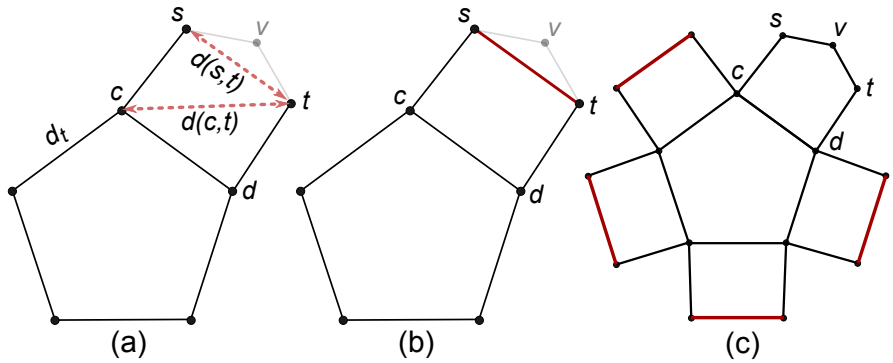


Figure 3: (a) Greedy routing may fail on the perimeter of HTT. (b) Unreachable nodes are thus connected through an additional link. (c) Additional link is removed when no longer needed for greedy reachability. Note: d_t is the basic tessellation lattice distance.

Algorithm 1 Incremental HTT generation ensuring reachability for greedy routing.

AddNode (coordinate list L, nodes V, tessellation links E_t, additional links E_r):
Coords: $(s_x, s_y) = Pop(L)$; $V = V \cup s$
% Add tessellation links
for all $i \in V \setminus s$ **do**
  $dist$ = CalculatePoincareDistance(s, i)
  if $dist = d_t$ **then**
    $E_t = E_t \cup \{s, i\}$
  end if
end for
% Add greedy reachability ensuring links
for all $i \in V \setminus s$ **do**
  if GreedyRoutingFails(s, i) **then**
    $E_r = E_r \cup \{s, i\}$
  end if
end for
% Remove unneeded reachability links
for all $i \in$ set of neighbors of s **do**
  $E_{r,i} = \{\{e_1, e_2\} \in E_r \mid e_1 = i \text{ or } e_2 = i\}$
  for all $e = \{e_1, e_2\} \in E_{r,i}$ **do**
    $E_r = E_r \setminus e$
    if GreedyRoutingFails(e_1, e_2) **then**
      $E_r = E_r \cup e$
    end if
  end for
end for

(n,k)	d_t	d_r	min	avg	max	$\lvert E_t \rvert$	$\lvert E_r \rvert$
(3,16)	3.2		-	-	-	9568	0
(5,6)	2.1		2.93	2.93	2.93	5870	280
(10,4)	1.6		2.61	3.24	3.49	4700	420

Table 1: Number $|E_r|$ and length d_r of additional greedy reachability ensuring links in different (n,k) tessellations based HTTs, each with 4640 nodes.

The number and length of required additional links are shown in Table 1 for simulated HTT topologies with 4640 nodes, generated by Algorithm 1. My simulations show that the above problem of unreachability through greedy routing arises in case of less than 1% of all source-destination pairs, therefore, the number of required additional links will be relatively small $(5-10\%)$ compared to the total number of links. Moreover, d_r, the length of these additional links, i.e., the distance between end nodes of such links, is in the same range as the tessellation distance d_t, thus *keeping the localized nature of links* that enable simplified cabling management. To put this in perspective, it means that we only need to manage a small fraction of all links during the evolution of an HTT network, adding and removing them strictly in the vicinity of a newly connected node. This is in strong contrast, for example, with the Folded Clos structure, where recurring complete rewirings of the structure is inevitable during its growth (Fig. 1).

Although the analytical proof in Th. 1.2 considers infinite hyperbolic tessellations, my simulations readily showed that greedy routing can effectively find the shortest paths on finite HTT topologies, thus eventuating quasi the same values for average shortest path length and average greedy path length shown in Table 2.

The formal process of the incremental expansion is further discussed in Th. 2.1, where a more detailed practical algorithm adhering to realistic data center specific constraints (e.g., port counts, performance upgrades) is carefully described.

To improve the **throughput performance** of HTT, we can add more links between the nodes in the topology. For analysis and evaluation purposes, I present a simple heuristic algorithm, which produces more dense topologies by systematically adding more links on top of the basic topology (*Sec. 3.3.1*).

Thesis 1.3. *[C3, J1] I have proposed a heuristic algorithm that systematically adds links to the topology, while it preserves structural symmetry and link locality. The algorithm linearly improves the bisection bandwidth of HTT with each additional link. When increasing the number of links by* 100%*, it reduces the average path length by* $\simeq 30\%$

We can add links heuristically as follows. Take an (n,k) HTT as a basic topology and a radius r, and connect the nodes whose distance in the hyperbolic space is less than r. The Poincaré distance d_t between neighbor nodes is different for different (n,k) tessellations. If r is smaller than d_t, then there will be no extra links in the graph other than links in basic topology. If we increase r, more and more links are added. To meet the realistic constraint that servers can have a very limited number of ports, we use different connection radii, r_{sw} and r_{se}, in case of switches and servers, respectively (see Fig. 4 for an example). We connect two switches if their distance is less than r_{sw}. In case of server-server and server-switch links, r_{se} is used as a connection radius.

The quantitative performance improvement of the densification process is presented in Table 2. The shortest path property holds for the improved DHTT topologies as well, which is readily supported by simulations. Note that the generalization of this performance improvement method is described later in Rule 3 of Thesis 2.1.

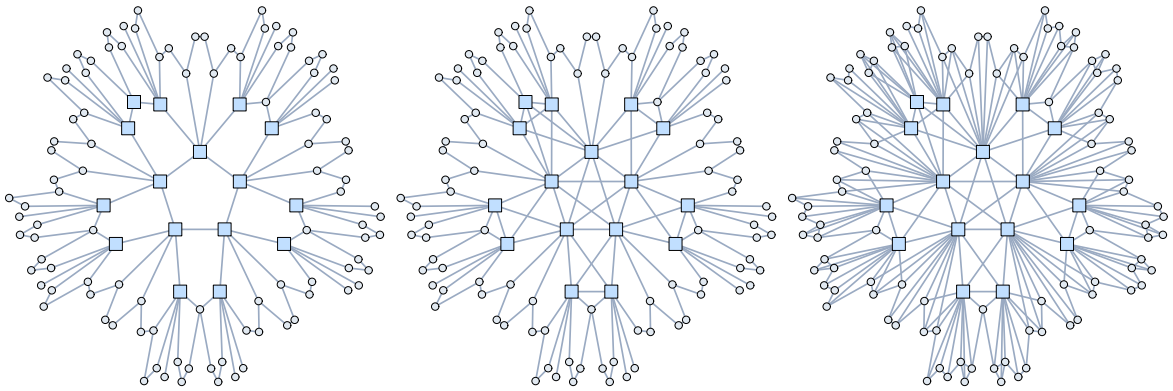


Figure 4: A basic HTT (left) and performance improved DHTT topologies achieved by increasing the connection radius r_{sw} (middle) and both radii r_{sw} and r_{se} (right).

Notation	Description	Notation	Description
$\|p_{sw}\|$	total number of switch ports	$\|p_{se}\|$	total number of server ports
$\hat{k}_{sw}$	switch max. degree	$\hat{k}_{se}$	server max. degree
$\bar{l}$	avg. shortest path length	$\bar{l}_g$	avg. greedy path length
D	diameter	B	bisection bandwidth

(n,k)	Type	r_{sw}	r_{se}	$\|p_{sw}\|$	$\|p_{se}\|$	$\hat{k}_{sw}$	$\hat{k}_{se}$	$\bar{l}$	$\bar{l}_g$	D	B
(3,16)	HTT	3.2	3.2	6700	12436	16	9	6.4339	6.4339	7	4709
	DHTT	5.5	3.2	9988	12436	64	9	5.4857	5.4857	7	5563
	DHTT	5.5	4.8	11252	18940	64	9	5.3190	5.3190	7	7468
(5,6)	HTT	2.1	2.1	3785	8515	6	5	10.0113	10.0113	12	3043
	DHTT	4.4	2.1	9045	8515	48	5	6.5071	6.5073	9	4260
	DHTT	4.4	3.3	11595	11885	48	5	5.6165	5.6166	9	6259
	DHTT	4.6	3.3	13605	11885	72	5	5.3321	5.3323	9	6701
(10,4)	HTT	1.6	1.6	2520	6880	4	4	13.3033	13.3047	17	2501
	DHTT	3.7	1.6	7810	6880	36	4	8.0920	8.0930	11	3752
	DHTT	3.7	3.3	11910	13460	36	4	6.2270	6.3171	11	6471

Table 2: The effect of densification process on different (n,k) tessellations in case of 4000-server topologies. B is the sum of the bandwidth of links in a cut that partitions the entire network nearly in half, thus upper bounds its achievable throughput.

In this Thesis Group, I proposed a theoretical interconnection structure that is both efficiently expandable and navigable. In the next Thesis Group, I augment the proposed structure to meet realistic constraints and requirements in today's data center networks.

4.2 Incrementally Expandable Data Center Architecture

The results so far demonstrated the basic properties of the previously defined, theoretical HTT interconnection network in terms of incremental expandability and routing performance. Data center architectures, however, must adhere to more specific practical considerations. For this end, I define a practical implementation of the HTT structure in this Thesis group to be applicable for real data center networks. Concretely, I present a fully fledged data center architecture, called *Poincaré DC* that includes a more realistic growth algorithm to accommodate the configuration (e.g., port number, interface speed, etc.) of commercially available switches and servers. Moreover, I augment default greedy routing to support multipath capabilities in HTT, and evaluate the cost efficiency of the system through real-world metrics.

Thesis Group 2. *I have proposed a data center architecture called Poincaré DC that implements the HTT structure, while it also considers real world DC requirements. I have proposed low complexity extensions to the original greedy routing algorithm to support load balancing and fault tolerance. I have showed that Poincaré DC requires low upfront cost and it is more cost efficient compared to a state of the art fat tree DC architecture.*

One important, albeit under-researched aspect of data center networks is incremental upgradability, i.e., adding servers and network capacity to the data center on-demand [19] [6]. Incremental roll-out is a logical strategy supported by industry experts [17]. In reality, there are fixed port switches, and the servers have limited port capabilities. In the following Thesis, first I show how we can build a *Poincaré DC* structure taking these real world aspects of DC components into consideration. Later, it is also discussed how the proposed system satisfies the requirement of incremental upgradability, and describes various methods for capacity provisioning in a cost optimized manner (*Sec. 4.1*).

Thesis 2.1. *[J1] I have designed an algorithm to expand the topology of Poincaré DC with an arbitrary number of servers without the need for complete rewiring during the expansion process. I have shown that the expansion process is strictly local, i.e., links must be rewired only in the immediate vicinity of the affected servers and switches, and the capacity of the structure can be seamlessly increased by adding extra links.*

From a very high level point of view, the expansion process will be the following. To build a *Poincaré DC* topology, first, we have to select an allowed (n, k) HTT. We may choose the parameters n and k such that properties (e.g., diameter, max degree, etc.) of the resulting topology best approaches our preferences. Now, we may simply start building the structure solely with servers. However, when the topology is gradually increased, servers may run out of free ports. If we do not want to (or cannot) increase a server's port count, then this server needs to be shifted towards the perimeter, to be replaced by a switch with higher port count. In general, we have to take care of two main things during the physical construction of our network. First, we need to ensure the correct allocation of coordinates of new servers in the system:

Rule 1. *[J1] The next position, where a new node can be installed, is an unoccupied place having the minimal distance from the center (0,0). (Ties are broken randomly.)*

We generate the coordinate list L with the method described in Section 4.1, which tells us the possible logical places of the devices. Recall that the coordinate list is sorted according to the distance of the location (x_i, y_i) from the center of the disk $(0, 0)$ in increasing order. This complies with Rule 1, and therefore, the expanding topology will be balanced.

Secondly, to exploit advantageous properties of the hyperbolic tessellation shown in Thesis Group 1, our main task is to carefully maintain the base HTT topology when growing a *Poincaré DC* structure. This is formalized in Rule 2:

Rule 2. *[J1] After adding a new node to the topology, corresponding links of the base HTT topology must be present in Poincaré DC.*

Rule 2 basically means that if there is a link between nodes v_1 and v_2 in HTT, then devices that are on the coordinates of v_1 and v_2 must be connected. The procedure of growing a *Poincaré DC* structure is described in Algorithm 2, and Figure 5 shows the expansion process for the (5,4) tessellation based HTT structure.

Algorithm 2 Growth of a *Poincaré DC* network.

AddServer(G, s):
$E_H = \varnothing$, $Q = [s]$, p_v = number of available ports in server v
repeat
  Let $r = Pop(Q)$, insert server r to coordinates $(x, y) = Pop(L)$ to meet Rule 1
  Try installing HTT links to meet Rule 2, and add these links to set E_H
  for all server w, s.t. new HTT links would increase its port requirement above p_w **do**
    Substitute server w with a switch, and $Push$ server w to Q
  end for
until $Length(Q) > 0$
Add links E_H to G to meet Rule 2

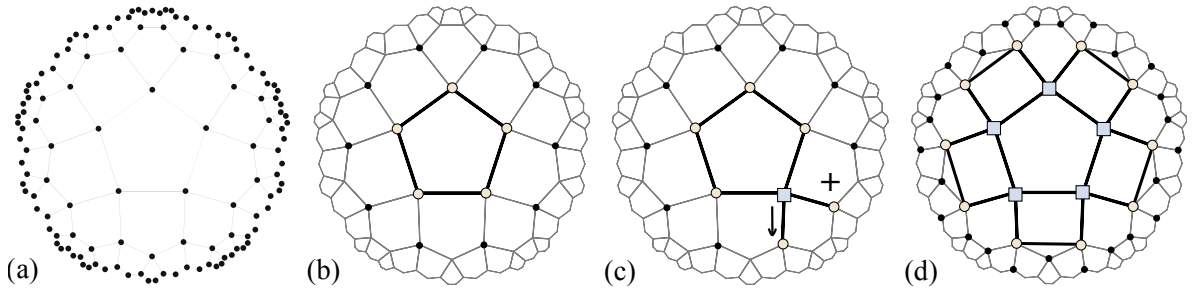


Figure 5: (a) Coordinates of empty locations (dots) in $(5, 4)$ HTT. (b) Initial topology with 5 severs (circle), $p_v = 2$. (c) Server is moved to an outer location and gets replaced by a switch (square) with higher port count, thus, the structure can accommodate a new server (+). (d) Snapshot of growth process with 5 switches and 10 servers.

As resource demand is getting higher over time, the performance of *Poincaré DC* can be structurally upgraded by adding more switching equipment as well. Switches with a higher port count can replace smaller switches in a plug-and-play manner always resulting in better structural benchmarks. Clearly, adding links to the inner part of the topology can have larger impact. With a simple performance upgrade rule we can assure that a new bigger switch is deployed in an easy manner:

Rule 3. *If a switch is replaced by a new switch with higher port count k_{nsw}, the new switch has to be connected to those k_{nsw} closest nodes, which have available free ports.*

Note that base HTT links can be immediately reinstated, which makes such upgrades conformant to Rule 2. The links that are shown with orange color on Figure 6 are not tessellation links. These links can be removed whenever a shorter link could be used as the result of employing Rule 3. When there are ties between distances between upgraded devices, these non-tessellation links can be arbitrarily added/removed following Rule 3 to optimize the overall performance.

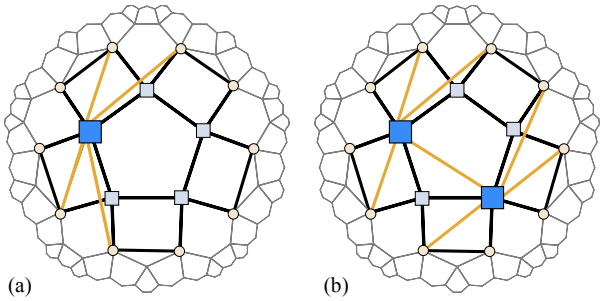


Figure 6: (a) A 4-port switch is replaced by an 8-port switch to enhance server connectivity, and thus, the performance of the structure. (b) To enhance connectivity among switches, we have to install at least another bigger switch. Orange links are non-HTT links, these can be easily managed by following Rule 3.

It is worth to note that, by following Rule 3, we can implement the DHTT algorithm described in Thesis 1.3, although Rules 1–3 permit custom tailored *Poincaré DC* topologies as well. During the growth process, according to Rules 1 & 2, the outer perimeter of the topology can be temporarily asymmetric. However, the rules also ensure that the topology expands symmetrically with each additional new layer of the tessellation, which results in a transparent cabling system. Importantly, the HTT structure requires such devices to be connected only that are also close to each other in physical space, which further simplifies cabling in the structure. It is also emphasized that the upgrade process in *Poincaré DC* affects only the immediate vicinity of the newly connected servers or switches, i.e., without impacting the data center core and main operation. Although, Rule 3 allows operators to construct non-tessellation links between arbitrary switch pairs to flexibly customize the capacity of the topology to real traffic patterns, they can also upgrade the capacity of the structure in a symmetrical manner to increase cabling transparency. Furthermore, the scope of an unintended individual cabling error is limited, as it would affect only a small part of the global network.

Besides an efficient network topology, high performance **routing with multipath features** (e.g., for multipath TCP, VM migration, and error tolerance) is crucial in data centers. In the following, I present low complexity multipath algorithms that implement load balancing and a failure tolerating mechanism in *Poincaré DC*. The proposed algorithms enhance the default greedy forwarding paradigm through simple local rules without requiring up-to-date global topology information (*Sec. 4.2*).

Thesis 2.2. *[C3, J1] I have showed numerically that Poincaré DC provides multiple greedy paths between node pairs. I have proposed low complexity multipath routing protocols for Poincaré DC to provide multiple paths for load balancing and fault tolerance. I have shown the protocols' efficiency through simulations.*

Disjoint greedy paths. Simple multipath algorithms, which forward packets based on local decisions, rely strongly on the number of edge disjoint paths that can be used by greedy routing. To measure how many such paths exist between pairs of nodes in an HTT topology G, I generate a special directed subgraph G_d from G for every d destination. In G_d, a link pointing from u to v exists only if u and v are connected in G and $d(u,d) > d(v,d)$. All link capacities are set to 1, and the maximal flow on G_d is calculated. Applying the Max-flow min-cut theorem on G_d, I can calculate the exact number of link-disjoint greedy routable multiple paths between s and d. Table 3 shows the number of link-disjoint greedy routable multiple paths averaged for all source-destination pairs in 1000-node *Poincaré* topologies. After only moderate topology upgrades, on average 2, maximally 4 greedy routable link disjoint paths are present in our simulated topologies.

Topology (1000 nodes)	Server ports		Greedy disjoint paths	
	avg.	max.	avg.	max.
$(4,10)$ DHTT, r_{sw}=3.3, r_{se}=3.3	3.371	4	2.138	4
$(4,10)$ DHTT, r_{sw}=3.7, r_{se}=3.3	3.663	4	2.569	4

Table 3: Link disjoint paths accessible by greedy routing.

Load-balancing. For load-balancing purposes, we can use the following simple distributed multipath algorithm: for a new incoming flow, choose the least-loaded outgoing link through which the packets can reach the destination on a greedy path. Algorithm 3 details the load balancing process proposed for *Poincaré DC*, while Table 4 shows the throughput simulation results of the algorithm. Here, we use the *Poincaré DC* topology from the second row of Table 5. We choose random pairs of servers in 4000-server topologies, and send 100 different flows of 1MB data from the same source to the same destination. Results are averaged over 1000 different source-destination pairs. In contrast to fat tree, where the bottleneck is the access link speed of the server, *Poincaré DC* leverages the multiple disjoint paths between source and destination servers.

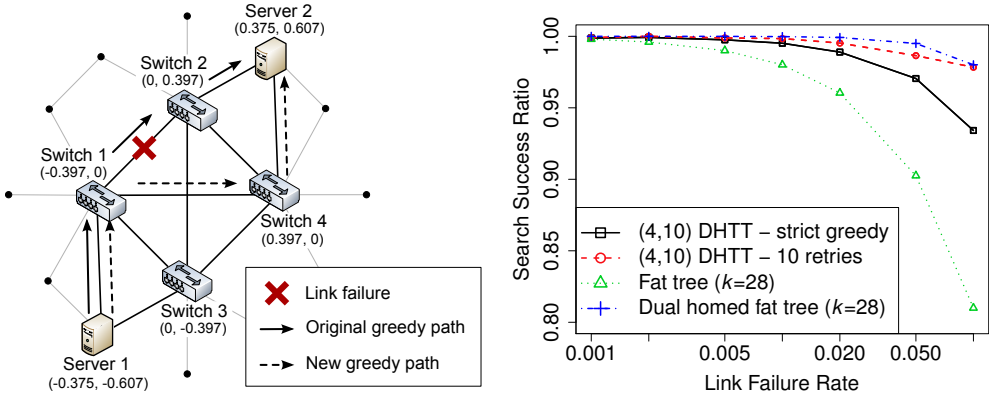
Algorithm 3 Greedy Load Balancing.

GreedyLoadBalancing(current node u, destination node t):
C = link neighbors i of node u, where $d(i,t) < d(u,t)$.
for all $j \in C$ **do**
  $w[j]$ = Traffic on link (u,j).
end for
$next\_hop = j$ such that $w[j]$ is minimal among $j \in C$.
ForwardPacket($next\_hop$.)

Topology	Avg. Multipath Throughput	Var	Min.	Max.
Poincaré DC fat tree equiv.	1527.41	406.43	1000	2898.55
fat tree ($n = 32$))	1000	0	1000	1000

Table 4: Comparison of multipath throughput capability of 4000 server fat tree and *Poincaré DC* systems (Mbps).

Fault tolerance. Greedy routing provides a fairly natural way of tolerating failures. Consider the scenario on Fig. 7a, where Server 1 sends traffic to Server 2. From the coordinates, we can compute the greedy path as Server 1 – Switch 1 – Switch 2 – Server 2. If, for example, the link between Switch 1 and 2 goes down, as indicated in the figure, Switch 1 notices the failure, and for the next packet received from Server 1 the greedy calculation gives Switch 4 as the next hop. Therefore, routing in *Poincaré DC* avoids the failed link without any global failure propagation and route recomputation, and it requires time only for the detection of the failure.



(a) (b)

Figure 7: (a) Fault tolerance in a (4,5) tessellation based *Poincaré DC* topology. (b) Greedy search and fat tree search success ratio versus random link failure rates.

In case of link failures, it can occur that greedy routing fails (stuck in a local minimum), however, there would be available non-greedy paths. In the dissertation, it is shown that the probability of such events in a *Poincaré DC* topology is very low compared to, e.g., the disconnection probability of servers from the fat tree topology in case of link failures. Moreover, we can exploit the path diversity of *Poincaré DC* to reduce the probability of greedy faults by considering the following algorithm. When a source node fails to find its destination with default greedy routing, it can assign a random trajectory "bias" (α) to the next packet, and retry the transmission. Intermediate nodes use this bias to assign weights d_i^α to their neighbors based on their distance d_i to the destination, and pick a neighbor with greater probability that has larger weight. We can easily set up such a probability distribution, and pick a neighbor with probability $p(j) = d_j^\alpha / \sum_i^{\#neighbors} d_i^\alpha$. By setting α to a very low value the algorithm always favors the shortest greedy path, while if set to a higher positive value, the traversed routes will be distributed among the many greedy routable paths hereby avoiding the local minimum. The pseudocode of the algorithm is shown in Algorithm 4.

Figure 7b shows that the resilience of the architecture is remarkable in case of realistic

Algorithm 4 Greedy Failure Handling.

GreedyFailureHandling(current node u, destination node t, α):
C = link neighbors i of node u, where $d(i,t) < d(u,t)$
for all $j \in C$ **do**
  $w[j] = d(j,t)^{\alpha}$
end for
$next\_hop = Random(p(j) = \frac{w[j]}{\sum_{i \in C} w[i]})$
ForwardPacket($next\_hop$)

link failure rates. I simulated random link failure events in the topology (4000 servers, r_{sw}=4.3, r_{se}=3.3), and measured the overall success ratio of greedy routing for 50000 source-destination pairs. The plot also shows the proposed failure handling feature of greedy search, retrying to find a path for a maximum of 10 times, which improves routing success. To compare the results to fat tree based topologies, I also plot the probability of host pairs remaining connected in these structures at the presence of link failures, which is an upper bound on the success rate.

After demonstrating both incremental expansion and high performance of my proposed DC architecture, as a primary real world aspect of data centers, I have also analyzed the structure's cost efficiency. I have compared the **throughput vs. cost performance** of *Poincaré DC* to the fat tree system via flow-level traffic simulations and by estimating costs from real world DC components' prices (*Sec. 4.4 and 4.5*).

Thesis 2.3. *[C3, J1] I have shown that the capital cost of Poincaré DC is price competitive compared to an equal size, equal performance fat tree system. Poincaré DC requires low upfront capital costs, and it eliminates the need for complete rewiring during the expansion, which further reduces operational costs.*

To evaluate the systems' throughput, I have implemented a simple flow-level traffic simulator in C++ adapted from [7], that simulates the original fat tree [3] and greedy routing on the generated topologies. All topologies contain 4000 servers with varying number of switches (S), and the results are averaged over 10 simulation runs. The results of the simulations is shown in Table 5.

Notation	Description	Notation	Description
$\lvert p_{sw,10G}\rvert$	num. of 10 Gbps switch ports	$\sum T$	total aggregate throughput (Gbps)
$\lvert p_{sw,1G}\rvert$	num. of 1 Gbps switch ports	$\bar{T}_{se}$	avg. per server throughput (Gbps)
$\lvert p_{se,1G}\rvert$	num. of 1 Gbps server ports	t	runtime of total data transfer (ms)

Topology	r_{sw} / r_{se}	S	$\lvert p_{sw,10G}\rvert$	$\lvert p_{sw,1G}\rvert$	$\lvert p_{se,1G}\rvert$	K$	$\sum T$	$\bar{T}_{se}$	t
$(4,10)$ DHTT	3.3 / 3.3	640	2400	8048	13510	2755.8	239.91	0.143	1334.67
$(4,10)$ DHTT	4.3 / 3.3	640	4700	9898	13510	3515.8	494.47	0.280	649.33
$(4,10)$ DHTT	4.6 / 3.3	640	7000	8878	13510	3988.8	558.77	0.364	573.17
fat tree ($k = 28$)		980	0	25952	4000	2995.2	471.91	0.265	680.5
fat tree ($k = 32$)		1280	0	36768	4000	4076.8	473.97	0.265	680.17

Table 5: Throughput comparison of 4000 server *Poincaré DC* and fat tree DCs.

Based on these traffic simulations, I compared the throughput vs. cost performance of the two systems such that I generated dual homed fat tree (DHFT) topologies with 500, 1000, 2000, ..., 8000, 8200 servers. Then, densified hyperbolic tessellation topologies (DHTT) with the same number of servers were generated, and the parameters of the system were manually adjusted in every network size instance, so that the throughput of *Poincaré DC* would be the same as in case of the corresponding sized fat tree topologies. The cost of the two systems were calculated and interpolated (Fig. 8). When starting with a low port DHFT (small), a complete rewiring becomes necessary earlier in the evolution of the structure compared to the case of high port count DHFT (big). Note the first big jump in the cost of DHFT (small) around the size of 1000 servers. Moreover, DHFT (big) implies higher upfront costs, and the inevitable complete rewiring comes around 8200 servers. *Poincaré DC*, however, requires low upfront costs and eliminates the need for complete rewiring during the expansion. The two lines corresponding to *Poincaré DC* show the minimum amount of money needed to build a working tessellation system (marked with plus signs), and more importantly, a system that is equivalent to a fat tree system in terms of throughput (marked with triangles). In the first case, the initial DC has a very low entry cost, and it can be smoothly upgraded in small increments. In the second case, a tessellation based DC, that achieves the same throughput as fat tree, can be built with lower entry cost, and stays cheaper than or at worst comparable to its fat tree counterpart.

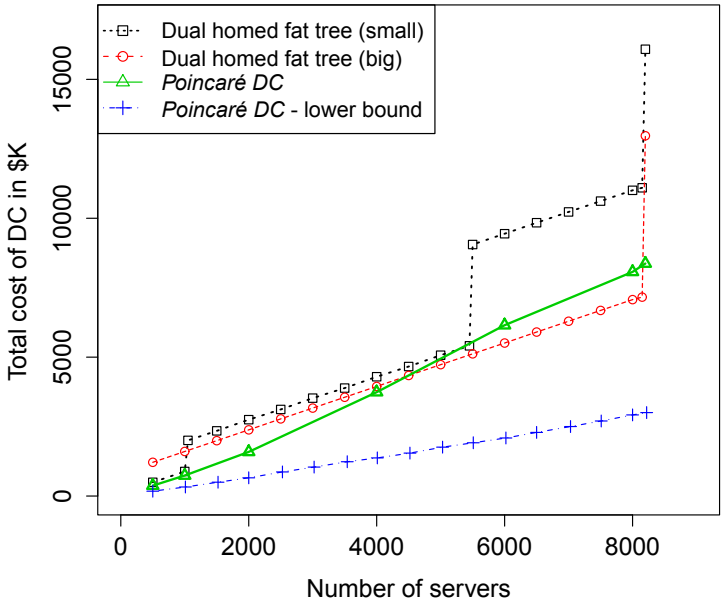


Figure 8: Comparison of total cost for dual homed fat tree and densified hyperbolic tessellation topology based *Poincaré DCs* while adding servers incrementally.

4.3 Cabling Complexity Reduction in Data Center Networks

A practical alternative to the currently prevalent fat-tree DC architecture [3] is the flattened butterfly (FBFly) structure made possible by current high radix switches [14]. It maintains similar performance at lower cost by using fewer networking equipment and more complex (i.e., adaptive) routing [14, 2]. Since the dominant factor in the FBFly cost is determined by the cost of long inter-rack cables, there are proposals for architectural modifications of its structure to reduce the cabling complexity. This is achieved by trading off the high number of long inter-rack cables for further increased control plane complexity [15].

The butterfly and flattened butterfly topology is simply defined as the following. A k-ary n-fly butterfly topology contains k^n input and output terminals (Fig. 9a). It consists of n levels (rows) of butterfly interconnects: at level l, where $l = 0, 1, ..., (n-1)$ nodes are connected to others at distances that are multiples of k^l. Now, a k-ary n-flat FBFly topology is constructed from a k-ary n-fly butterfly by combining the switches in each of the k^{n-1} columns of the butterfly topology into one single switch (Fig. 9b). The unidirectional input and output ports of the terminals are combined into bidirectional ports of $N = k^n$ servers, such that each switch hosts k servers. In general, when a butterfly is flattened, the transformation results in a $n-1$ dimensional array of $S = k^{n-1}$ switches. Each switch is connected to each of the $k-1$ other switches that align with it in each of the dimensions. Thus, switches are connected to each other by a full mesh along *every* dimension. Note that there are $k^{n-2}(n-1)$ full meshes in a k-ary n-flat FBFly topology.

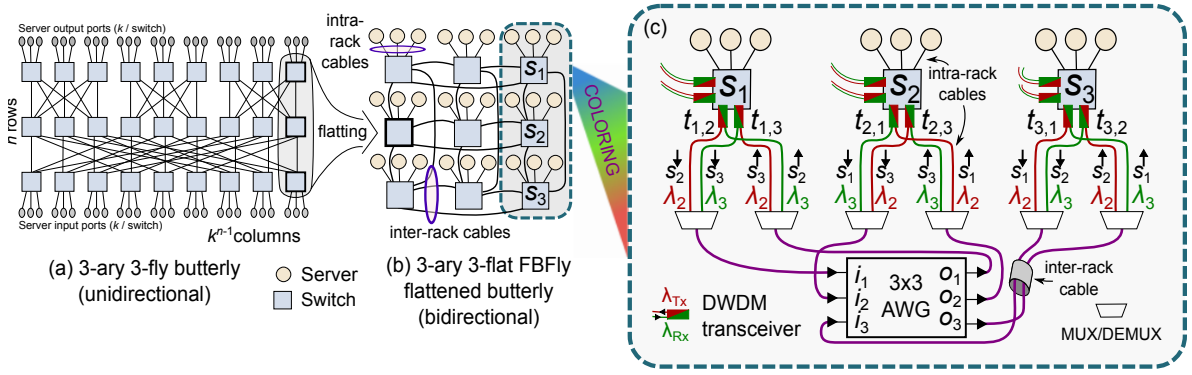


Figure 9: (a) 3-ary 3-fly butterfly topology. (b) 3-ary 3-flat flattened butterfly obtained by combining the switches in a column. (c) Each of the $k^{n-2}(n-1) = 6$ original FBFly full meshes is transformed into a DWDM optical "pseudo"-full mesh (colored FBFly).

In this Thesis group, I show that cabling complexity (defined as the number of long inter-rack cables) in FBFly structures can be reduced by an order of magnitude *without* trading off an increase in the control plane complexity.

Thesis Group 3. *I have proposed the colored flattened butterfly (C-FBFly) structure to reduce cabling complexity in flattened butterfly (FBFly) networks by an order of magnitude. I have shown that C-FBFly also reduces the total cost of cabling in large topologies, while its capital cabling cost strongly depends on fiber and transceiver prices, but only weakly depends on the costs of extra optical devices required by the modification.*

I adopt the idea widely known in networking to replace mesh cabling by switched star topologies by employing arrayed waveguide grating routers (AWGR) and dense wavelength division multiplexing (DWDM or colored) optics [13, 20]. Contrarily to the already proposed optical switching concepts for DCs [12], my colored flattened butterfly (C-FBFly) structure allocates sufficient optical circuits for arbitrary traffic patterns, without introducing a complex optical control plane; switching in C-FBFly is solely done by the electrical switches. First, I introduce my proposed C-FBFly structure, and then I present my cabling complexity reduction results (*Sec. 5.4.1 and 5.4.2*).

Thesis 3.1. *[C1, C2] I have defined the colored flattened butterfly structure, which I evaluated in a techno-economical model. I have shown that C-FBFly reduces the total length of fiber cables by an order of magnitude compared to the original FBFly. I have shown that if colored vs grey transceiver price difference is* $\leq 110\%$*, then C-FBFly reduces capital expenditure cabling costs by* $\geq 5\%$ *in large (*$N > 70000$*) networks. I have shown that CapEx cabling costs of C-FBFly strongly depends on optical fiber and transceiver prices and weakly depends on the price and installation costs of extra optical devices.*

The key idea in the C-FBFly structure is the transformation of the $k\,(k-1)/2$ long inter-rack cables, *for all full meshes*, within each dimension of the k-ary n-flat FBFly topology, into a "pseudo"-full mesh consisting of just k shorter inter-rack cables (Fig. 9c).

Definition 3.1. *[C1, C2] The* ***colored flattened butterfly*** *is obtained by replacing the regular grey optical transceivers in the switches of a regular flattened butterfly structure with DWDM transceivers. Furthermore, to substitute full meshes, optical arrayed waveguide grating routers (AWGR) are connected to all colored transceivers through a layer of multiplexers and demultiplexers on each end.*

The AWGR implements a logical full mesh on a star topology by resolving the contention of signals in the wavelength domain. It routes each wavelength λ of input i to output $[(i+\lambda-2)\ mod\ M)]+1, 1 \leq i \leq M, 1 \leq \lambda \leq \Lambda$, where M is the number of AWGR ports and Λ is the total number of wavelengths [12]. Each transceiver of a switch in the pseudo-full mesh is set to transmit on a particular wavelength such that the AWGR routes its signal to the appropriate destination switch, i.e., the corresponding switch in the original FBFly. The resulting structure is an optical star network with the AWGR in the center. The transformation is repeated for every full mesh of FBFly, and for every corresponding switch. In this way, every physical full mesh of the FBFly topology is substituted for a pseudo-full mesh, and the resulting structure is called as a colored flattened butterfly (C-FBFly).

Importantly, the proposed modification does not increase the control plane complexity of the original FBFly. From the switches point of view, this star topology is logically a fully meshed network. Since the proposed modification only affects the physical layer (L1), it does not incur any control plane modification. Furthermore, I point out that a pseudo-full mesh contains more, but mainly short fibers than a regular full mesh, since there are twice as much cables between the switch ports and the Mux/Demux's as in a regular full mesh. However, as I show in my following Theses, the number of long inter-rack cables (i.e., the cabling complexity) is reduced by a significant amount, which

in turn reduces the total amount of fibers in the entire network, resulting in capital cost savings in case of large structures.

I have evaluated the proposed modification in a **techno-economical model**, in which I have considered all relevant technological and financial constraints.

Definition 3.2. *[C1] The techno-economical model of C-FBFly (Def. 3.1) integrates total optical cable length estimation based on physical constraints (switch capacity, power consumption profiles, transceiver and optical fiber parameters) and optical cabling cost calculations (equipment prices and installation costs of grey vs. colored optics).*

Cable length calculation. In the first part of the model, I calculated the total optical cable length in FBFly by adopting its original floor space model, and I also incorporated realistic rack power constraints. Concretely, I considered the limitations of the maximum power that can be provided for a rack to calculate the allowable server density per a square meter. Similarly to the original flattened butterfly cabinet layout [14], I assume that the racks are aligned into rows of working cells in a raised floor area. The edge length of this area is estimated by $E_l = \sqrt{N/\rho}$, where ρ is the node density in $nodes/m^2$. Then the average cable length between the switches of a FBFly network can be simply estimated as $L_{avg} = E_l/3$.

Multi-gigabit links longer than 5 meters cannot be efficiently built with electrical cables, so the common practice is that these longer links are created with optical cables [18]. In my model I consider optical cables of length $L_{avg} \geq$ 2 m, accounting thus for the fact that the inter-rack cables are running in cable trays few feet over the racks, which adds about $L_{const} = 3$ meters to every inter-rack cable. I analyzed real world power density scenarios and found that, according to the presented cable length estimation method, FBFly DCs containing a thousand or more servers can only be built efficiently with optical inter-rack cables. Fig. 10 shows the arrangement of both short and long optical cables in C-FBFly.

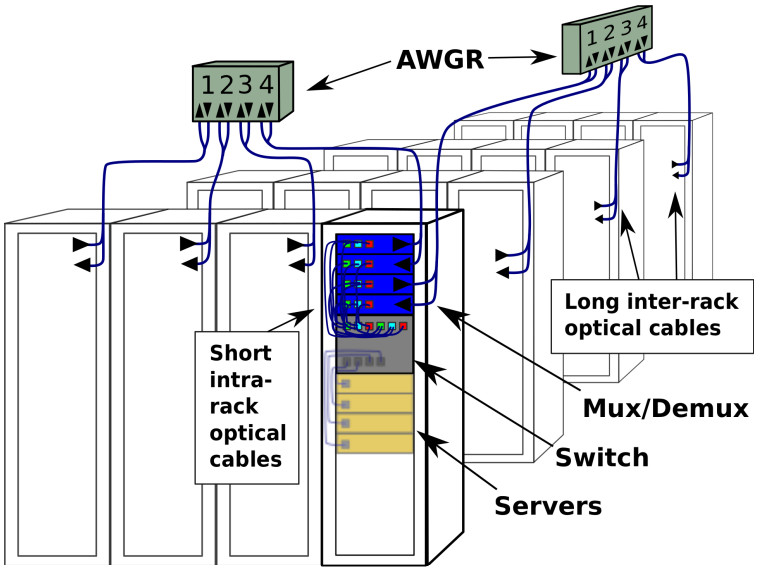


Figure 10: Optical cabling in C-FBFly, one pseudo-full mesh is shown in each dimension.

Cabling cost calculation. I calculated the total costs of **grey** and **colored** optical cabling in FBFly and C-FBFly topologies, respectively. In particular, I calculated the fiber cable costs and the transceiver costs, and I also account for the extra optical devices (Mux/Demux and AWGR) in case of C-FBFly. I strived for an accurate cost comparison by accounting for the installation costs of both the inter-rack cables and extra optical devices. I have omitted the cost components, however, that are equivalent in case of both structures, so switch, server, etc. equipment and installation costs are not included in my model. The cost of cabling between servers and switches is omitted as well.

The cost reduction results presented in these Theses are quite conservative, relative to C-FBFly, because they are exclusively based on capital expenditure (CapEx) costs. I must point out that the operational cost component (i.e., OpEx) is arguably significantly reduced given the very large cabling complexity reduction in C-FBFly.

First, using my techno-economical model of C-FBFly (Def. 3.2), I have numerically calculated the total optical cable lengths in case of both structures, then I compared the CapEx cabling costs of the colored and the grey FBFly structure. Fig. 11a clearly shows that the C-FBFly's total cable length is about an order of magnitude shorter than the grey FBFly's.

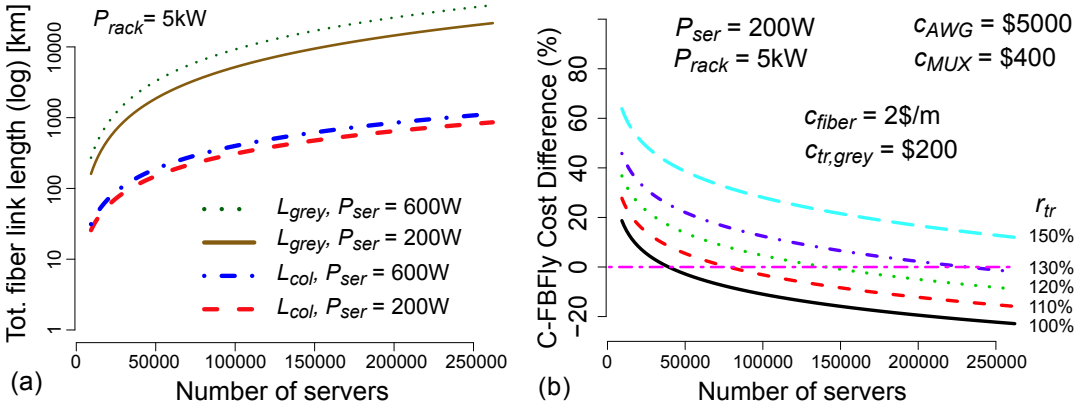


Figure 11: (a) Total length of grey and colored optical cables for different server power consumption and given rack power availability. (b) Cost save (-) or increase (+) in case of implementing the C-FBFly structure in a reference scenario for different network size.

I have also compared the total optical cabling costs of C-FBFly and FBFly. I have defined the cabling cost of C-FBFly as the proportional increase (positive) or decrease (negative) of capital optical costs compared to the original FBFly optical cabling costs

$$C_{C-FBFly} = (C_{colored} - C_{grey})\,100\,/C_{grey}. \tag{2}$$

$C_{C-FBFly}$ is calculated for FBFly data centers in the range of 9K to 260K servers. The cost of the grey and the colored FBFly's optical interconnection network is calculated based on web prices. I assumed cable installation cost of \$6.25 per inter-rack cable, and \$2.5 for short cables similarly to [18]. The installation cost of an AWGR or Mux/Demux device is assumed \$50. For the transceiver cost component, I have defined a variable r_{tr} to denote the price of a colored transceiver compared to the price of a grey transceiver

in terms of percentages:

$$r_{tr} = \frac{c_{tr,col}}{c_{tr,grey}} 100. \tag{3}$$

Gridlines of different r_{tr} values help to identify the ratio of colored vs. grey transceiver prices, at a given size of the structure, when the cabling complexity reduction results in cost reduction at the same time. My purpose in this Thesis is to give an overview of prices which result in CapEx cost saving for reasonable sized large scale data center networks. Fig. 11b shows the cost savings for a C-FBFly reference scenario with realistic power parameters and optical equipment costs. Note that $C_{C-FBFly} < 0\%$ corresponds to capital cost savings, when using colored optics. In turn, when $C_{C-FBFly} > 0\%$, then the reduction in cabling complexity is achieved at a higher capital cost. For example, in case of a 200K server structure, implementing C-FBFly with 10% more expensive colored transceivers and reference optical prices, results in 12% cabling cost reduction compared to FBFly, which amounts to about \$14M in total cost saving.

I was interested in equipment prices when the colored and the grey cabling costs equal out each other for a given size of structure. This tells us the desired equipment prices when deciding to chose the colored structure over the grey one. For this reason, I indicated the sensitivity of the cost balance to the change in different equipments' prices, such that I have assessed which type of equipment dominates the cost balance. The

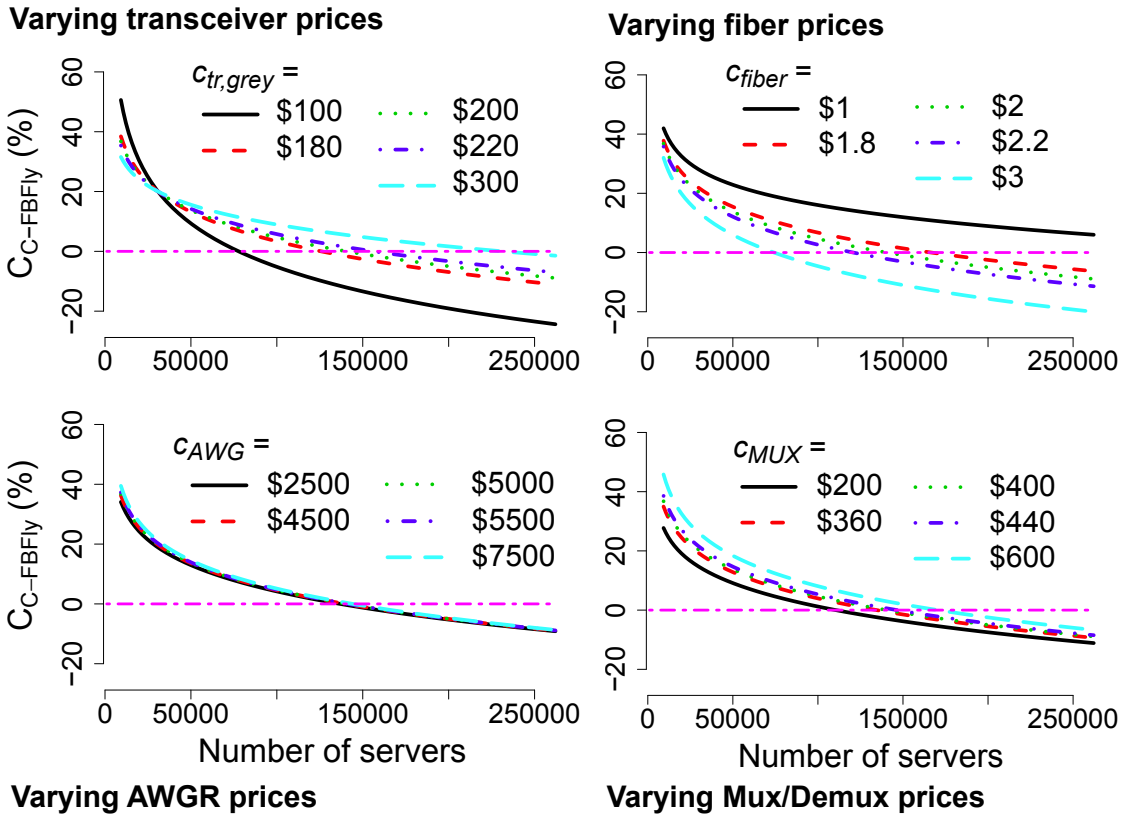


Figure 12: Sensitivity analysis for the change in FBFly cabling costs $C_{C-FBFly}$ when specifying $r_{tr} = 120\%$ and varying optical equipment prices. Note the difference of cost sensitivity to fiber cable and grey transceiver prices vs. AWGR and Mux/Demux prices.

price difference of colored and grey transceiver prices strongly determines the financial feasibility of C-FBFly, since we must use the same amount of transceivers in both cases. I analyzed the change in FBFly cabling costs $C_{C-FBFly}$ for distinct equipment price values, while setting $r_{tr} = 120\%$. This sensitivity analysis is done by increasing and decreasing each optical equipment price in turns by 10% and 50% relative to the reference scenario (Fig. 11b). The first and second plot of the first row on Fig. 12 show that the cost saving achieved by C-FBFly is highly sensitive to grey transceiver and fiber cable prices. The second row indicates that the capital cost is less sensitive to AWGR and Mux/Demux price variance. I mention that the sensitivity of $C_{C-FBFly}$ to all installation prices are similar to the sensitivity of the AWGR equipment price component. The main message here is that the purchase and installation of extra optical devices do not contribute much to the CapEx cost of cabling in C-FBFly. On the other hand, high optical fiber prices and low transceiver prices greatly favor C-FBFly.

To elaborate on the OpEx cost components, I have quantified the cabling complexity reduction as the reduction in the number of inter-rack cables (*Sec. 5.4.3*).

Thesis 3.2. *[C1, C2] I have shown that C-FBFly achieves up to* 48*-fold reduction in the number of inter-rack cables compared to regular flattened butterfly (FBFly) networks.*

The C-FBFly cabling complexity reduction depends on four factors: the parameter k of the FBFly structure, the width of the optical C-band B_{C-band}, the channel spacing of the signals $Ch_{spacing}$, and the number of sub-links in a link l_{sub}. The cabling complexity reduction can be quantified by taking the ratio of the number of inter-rack cables in the grey structure to the number of inter-rack cables in the colored structure:

$$R = \frac{k-1}{2} \left\lceil \frac{\left\lfloor \frac{B_{C-band}}{Ch_{spacing}\, k} \right\rfloor}{l_{sub}} \right\rceil . \tag{4}$$

An immediate upper bound with a 4800 GHz wide C-Band and 50 GHz channel spacing is $\hat{R} = (48(k-1))/(k\, l_{sub}) \approx 48$. The approximation is achieved when $k \to 96$, assuming one sub-link per link ($l_{sub} = 1$). Thus, 48 is a rough estimation of the theoretical upper limit of the cabling complexity reduction. Fig. 13a shows the scaling of the inter-

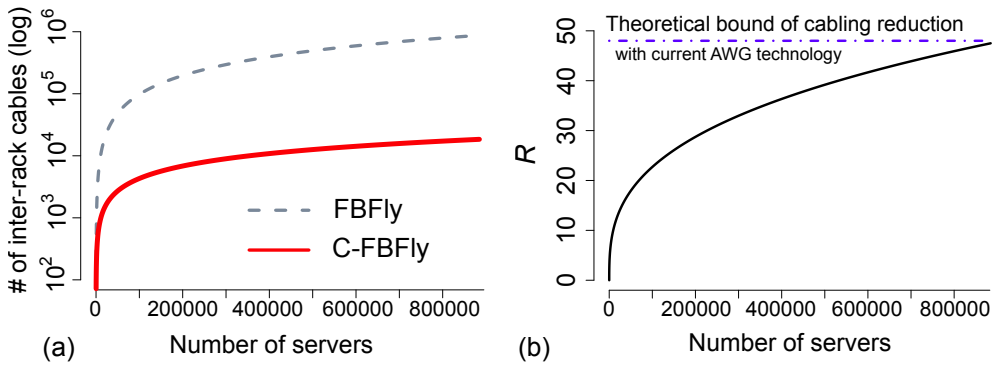


Figure 13: (a) Total number of inter-rack cables in FBFly and C-FBFly. (b) Ratio R of the number of inter-rack cables in FBFly to inter-rack cables in C-FBFly.

rack cables for FBFly and C-FBFly. Fig. 13b illustrates R. Note that the logarithmic behavior of R in the number of servers N is due to the exponential scaling of servers.

I argue that, by localizing the fully meshed cabling into the racks, the overall cabling management is significantly simplified. The simplified cable management further reduces the risk of miswiring and unplanned downtime of the data center. The detailed quantification of these operational (OpEx) costs is regarded as future work.

5 Validation

I have used computer simulations to validate the proposed hyperbolic tessellation topology generation methods and greedy routing algorithms. The structural properties and navigational efficiency in the hyperbolic interconnection structure was confirmed by analytical proofs. I have implemented a flow level traffic simulator and analyzed the throughput and load-balancing performance. During the evaluation of the cabling cost model, I have used current web based prices as an estimate of the optical cabling equipment, and discussions with experts from the optical community (at OFC 2014) confirmed my assumptions regarding my techno-economical model.

6 Application of Results

As one of my main contributions, I propose *Poincaré DC*, a fully fledged incrementally expandable data center architecture. The presented feasibility analysis indicates that the proposed structure provides throughput performance comparable to the fat tree structure, which is the foundation for current state of the art DC topologies. However, contrary to the fat tree system, *Poincaré DC*'s flexibility enables owners to start from a very cheap yet working DC with significantly low entering costs, and incrementally upgrade their system to any desired performance level. Concretely, servers can be added in *Poincaré DC* one by one, and every single link improves performance without the need for additional configuration. I have shown that the seamless evolution of the network is free of cost and labor intensive full structural rewiring, and this feature can be a huge incentive for companies and organizations to adopt my system. Importantly, the proposed architecture is deployable today, as the implementation of the structure is readily available for the OpenFlow platform, and with minimal effort it can be implemented on the NetFPGA platform as well. Moreover, my work may be regarded as an interesting contribution among the widely known DC architectures, and as such it may be useful for academic curriculum in networking and distributed system courses.

On the other hand, the proposed method for cabling complexity reduction builds upon commercially available optical devices, so it can be employed in data centers today. My integrated techno-economical model can be applied to evaluate individual conditions under which the proposed modification also achieves capital expenditure reduction in a new DC deployment.

7 Conclusion

In my dissertation, I have made both theoretical and practical contributions towards efficient cabling in data centers. I have proposed an incrementally expandable interconnection network based on hyperbolic tessellations (HTT) that provides low diameter and low control plane overhead. My initial assumption on the possible utilization of a hyperbolic geometrical structure in data centers has been confirmed in terms of the optimality of paths. To achieve high capacity in the center of the structure, the links are densified ($DHTT$), yet this procedure can be seamlessly implemented in the structure yielding both flexibility and cost effectiveness. Moreover, the theoretical structure is augmented with practical extensions to devise a high performance, cost efficient, and fully fledged incrementally expandable data center fabric (*Poincar DC*). Although greedy geometric routing effectively finds the shortest path in the proposed structure, high performance traffic engineering in state of the art DCs require more sophisticated routing fabrics. I have showed that the default routing algorithm can be easily supplemented with simple routines, which modify the routing decision process to enable load balancing and failure tolerance. I argue that localized greedy routing can provide a simple but efficient substrate for more complex routing fabrics in real word HTT based systems.

In my work, I have also evaluated a general method for cabling complexity reduction in state of the art DC structures by employing state of the art optical wavelength division multiplexing techniques. This method significantly lowers the number of long inter-rack cables without imposing additional control plane complexity on the underlying system; the routing of optical paths is solely performed in the physical layer. Moreover, I have showed a methodology to assess when it is also cost efficient to employ the proposed cabling reduction method. One can argue, that in the near future the diminishing difference between colored vs. grey optical transceiver prices will boost the adoption of the presented cabling complexity reduction method in real word DCs.

Publications

Publications marked with an asterisk (*) are related to the Theses.

Journal Papers

[J1] ***M. Csernai**, A. Gulyás, A. Kőrösi, B. Sonkoly, G. Biczók. Incrementally Upgradable Data Center Architecture Using Hyperbolic Tessellations. *Computer Networks*, 57(6):1373-1393. Elsevier, Apr. 2013. (6/4 = 1.5 points)

[J2] *A. Telcs, **M. Csernai**, A. Gulyás. Load Balanced Diffusive Capture Process On Homophilic Scale-Free Networks. *Physica A: Statistical Mechanics and its Applications*, 392(3):510-519. Elsevier, Feb. 2013. (6/2 = 3 points)

[J3] G. Rétvári, A. Gulyás, Z. Heszberger, **M. Csernai**, J. Bíró. Compact Policy Routing. *Distributed Computing*, 26(5-6):309-320. Springer, Oct. 2013. (6/4 = 1.5 points)

[J4] *D. Szabó, A. Gulyás, **M. Csernai**, Z. Heszberger. Struktúrafüggetlen címzésen alapuló önszervezõdõ útvonalválasztási architektúra. *Híradástechnika*, 66:2-10. 2011. (2/3 = 0.66 points)

[J5] ***M. Csernai**, A. Gulyás, Z. Heszberger, S. Molnár, B. Sonkoly. Congestion control and Network Management in Future Internet. *Infocommunications Journal*, 4:14-22. 2009. (4/4 = 1 point)

[J6] *T. Henk, R. Szabó, S. Molnár, B. Sonkoly. **M. Csernai**, A. Gulyás, Z. Heszberger, L. Gyarmati, T. A. Trinh. A jövõ Internetének kutatásai. *Híradástechnika*, 64:23-33. 2009. (2/8 = 0.25 points)

Conference Papers

[C1] ***M. Csernai**, F. Ciucu, R.-P. Braun, A. Gulyás. Towards 48-Fold Cabling Complexity Reduction in Large Flattened Butterfly Networks. To appear in *IEEE INFOCOM*, 2015. (3/3 = 1 point)

[C2] ***M. Csernai**, F. Ciucu, R.-P. Braun, A. Gulyás. Reducing Cabling Complexity in Large Flattened Butterfly Networks by an Order of Magnitude. In *Optical Fiber Communication Conference (OFC)*, paper Th2A.56., Optical Society of America (OSA) Technical Digest, 2014. (3/3 = 1 point)

[C3] ***M. Csernai**, A. Gulyás, A. Kőrösi, B. Sonkoly, G. Biczók. Poincaré: a Hyperbolic Data Center Architecture. In *32nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 8-16. IEEE, 2012. (3/4 = 0.75 points)

[C4] **M. Csernai**, A. Gulyás. Wireless Adapter Sleep Scheduling Based on Video QoE: How to Improve Battery Life When Watching Streaming Video? In *Computer Communication and Networks (ICCCN)*, pages 22-27. IEEE, 2011.
(3/1 = 3 points)

[C5] *D. Szabó, **M. Csernai**. Self-Organizing Structureless Routing Architecture. In *International Student Conference on Electrical Engineering (POSTER)*, pages 1-8. 2011. (3/2 = 1.5 points)

[C6] G. Rétvári, A. Gulyás, Z. Heszberger, **M. Csernai**, J. Bíró. Compact Policy Routing. In *Principles of Distributed Computing (PODC)*, pages 149-158. ACM, 2011. (3/4 = 0.75 points)

[C7] **M. Csernai**, A. Gulyás, G. Rétvári, Z. Heszberger, A. Császár. The Skeleton of the Internet. In *Global Telecommunications Conference (GLOBECOM)*, pages 1-5, IEEE, 2010. (3/4 = 0.75 points)

[C8] ***M. Csernai** Közösségi hálózaton alapuló útvonalválasztó eljárás tervezése és modellezése. *TDK Paper.* 1st prize, with special award of the Dean of BME. 2009.

References

[1] D. Abts and B. Felderman. A Guided Tour Through Data-Center Networking. *Communincations of the ACM*, 55(6):44–51, June 2012.

[2] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu. Energy Proportional Datacenter Networks. In *ACM SIGARCH Computer Architecture News*, volume 38, pages 338–347. ACM, 2010.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat. A Scalable, Commodity Data Center Network Architecture. In *Proc. of ACM SIGCOMM*, pages 63–74. 2008.

[4] Corning Incorporated. Sustaining the Cloud with a Faster, Greener and Uptime-Optimized Data Center. `http://goo.gl/fAloQ`, 2012.

[5] A. Curtis, T. Carpenter, M. Elsheikh, A. Lopez-Ortiz, and S. Keshav. REWIRE: An Optimization-based Framework for Data Center Network Design. In *Proc. of IEEE INFOCOM*. 2012.

[6] A. Curtis, S. Keshav, and A. Lopez-Ortiz. LEGUP: Using Heterogeneity to Reduce the Cost of Data Center Network Upgrades. In *Proc. of the 6th International COnference on emerging Networking EXperiments and Technologies (CoNEXT)*. 2010.

[7] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee. DevoFlow: Scaling Flow Management for High-Performance Networks. In *Proc. of ACM SIGCOMM*, pages 254–265. 2011.

[8] David E. Joyce. Hyperbolic Tessellations. http://goo.gl/hbpgrz.

[9] N. Farrington, E. Rubow, and A. Vahdat. Data Center Switch Architecture in the Age of Merchant Silicon. In *Proc. of 17th IEEE Symposium on High Performance Interconnects*, pages 93–102. 2009.

[10] L. Gyarmati, A. Gulyás, B. Sonkoly, T. A. Trinh, and G. Biczók. Free-Scaling Your Data Center. *Computer Networks*, 57(8):1758–1773, June 2013.

[11] T. Issenhuth. Representative Cloud Scale Data Center Design. *IEEE 802.3 400Gb/s Ethernet Study Group Plenary, Geneva, Switzerland*, July 2013.

[12] C. Kachris and I. Tomkos. A Survey on Optical Interconnects for Data Centers. *IEEE Communications Surveys Tutorials*, 14(4):1021–1036, 2012.

[13] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown. Scaling Internet Routers Using Optics. In *Proc. of ACM SIGCOMM*, pages 189–200. ACM, 2003.

[14] J. Kim, W. J. Dally, and D. Abts. Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks. In *Proc. of International Symposium on Computer Architecture*, 13, pages 126–137. 2007.

[15] J. Kim, W. J. Dally, S. Scott, and D. Abts. Technology-Driven, Highly-Scalable Dragonfly Topology. *ACM SIGARCH Computer Architecture News*, 36(3):77–88, 2008.

[16] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguna. Hyperbolic Geometry of Complex Networks. *Physical Review E*, 82(3):036106, 2010.

[17] A. Licis. Data Center Planning, Design and Optimization: A Global Perspective. http://goo.gl/Sfydq, June 2008.

[18] J. Mudigonda, P. Yalagandula, and J. C. Mogul. Taming the Flying Cable Monster: A Topology Design and Optimization Framework for Data-Center Networks. In *Proc. of USENIX ATC*. 2011.

[19] A. Singla, C. Y. Hong, L. Popa, and P. B. Godfrey. Jellyfish: Networking Data Centers, Randomly. In *Proc. of USENIX NSDI*. 2012.

[20] X. Ye, S. J. B. Yoo, and V. Akella. AWGR-Based Optical Topologies for Scalable and Efficient Global Communications in Large-Scale Multi-Processor Systems. *Journal of Optical Communications and Networking*, 4(9):651–662, 2012.