Computer Networks 113 (2017) 29–45

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

The stellar transformation: From interconnection networks to datacenter networks

Alejandro Erickson[a,*], Iain A. Stewart[a], Javier Navaridas[b], Abbas E. Kiasari[b]

[a] School of Engineering and Computing Sciences, Durham University, Science Labs, South Road, Durham DH1 3LE, UK
[b] School of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, UK

ARTICLE INFO

Article history:
Received 26 June 2016
Revised 5 November 2016
Accepted 2 December 2016
Available online 5 December 2016

Keywords:
Dual-port server-centric datacenter networks
Fault tolerance
Generalized hypercubes
Interconnection networks
Performance evaluation
Routing
Topological properties

ABSTRACT

The first dual-port server-centric datacenter network, FiConn, was introduced in 2009 and there are several others now in existence; however, the pool of topologies to choose from remains small. We propose a new generic construction, the stellar transformation, that dramatically increases the size of this pool by facilitating the transformation of well-studied topologies from interconnection networks, along with their networking properties and routing algorithms, into viable dual-port server-centric datacenter network topologies. We demonstrate that under our transformation, numerous interconnection networks yield datacenter network topologies with potentially good, and easily computable, baseline properties. We instantiate our construction so as to apply it to generalized hypercubes and obtain the datacenter networks GQ⋆. Our construction automatically yields routing algorithms for GQ⋆ and we empirically compare GQ⋆ (and its routing algorithms) with the established datacenter networks FiConn and DPillar (and their routing algorithms); this comparison is with respect to network throughput, latency, load balancing, fault-tolerance, and cost to build, and is with regard to all-to-all, many all-to-all, butterfly, random, hot-region, and hot-spot traffic patterns. We find that GQ⋆ outperforms both FiConn and DPillar (sometimes significantly so) and that there is substantial scope for our stellar transformation to yield new dual-port server-centric datacenter networks that are a considerable improvement on existing ones.

1. Introduction

The digital economy has taken the world by storm and completely changed the way we interact, communicate, collaborate, and search for information. The main driver of this change has been the rapid penetration of cloud computing which has enabled a wide variety of digital services, such as web search and online gaming, by offering elastic, on-demand computing resources to digital service providers. Indeed, the value of the global cloud computing market is estimated to be in excess of $100 billion [46]. Vital to this ecosystem of digital services is an underlying computing infrastructure based primarily in datacenters [5]. With this sudden move to the cloud, the demand for increasingly large datacenters is growing rapidly [20].

This demand has prompted a move away from traditional datacenter designs, based on expensive high-density enterprise-level switches, towards using commodity-off-the-shelf (COTS) hardware. In their production datacenters, major operators have primarily adopted (and invented) ideas similar to Fat-Tree [3], Portland [36], and VL2 [18]; on the other hand, the research community (several major operators included) maintains a diverse economy of datacenter architectures and designs in order to meet future demand [16,20,22,33,39,42]. Indeed, the "switch-centric" datacenters currently used in production datacenters have inherent scalability limitations and are by no means a low-cost solution (see, *e.g.*, [8,20,21,32]).

One approach intended to help overcome these limitations is the "server-centric" architecture, the first examples of which are DCell [20] and BCube [19]. Whereas in a switch-centric datacenter network (DCN) there are no links joining pairs of servers, in a server-centric DCN there are no links joining pairs of switches. This server-centric restriction arises from the circumstance that the switches in a server-centric DCN act only as non-blocking "dumb" crossbars. By offloading the task of routing packets to the servers, the server-centric architecture leverages the typically low utilisation of CPUs in datacenters to manage network communication. This can reduce the number of switches used in a DCN, the capabilities required of them, and their cost. In particular, the switches route only locally, to their neighbouring servers, and therefore have

* Corresponding author at: 3-864 Swan St., Saanich, BC, V8X 2Z3, Canada, Tel.: +1 778 350 4976.
E-mail address: alejandro.erickson@gmail.com (A. Erickson).

http://dx.doi.org/10.1016/j.comnet.2016.12.001

no need for large or fast routing tables. Thus, a server-centric DCN can potentially incorporate more servers and be both cheaper to operate and to build (see [37] for a more detailed discussion). Furthermore, using servers (which are highly programmable) rather than switches (which have proprietary software and limited programmability) to route packets will potentially accelerate research innovation [30]. Of course, the server-centric approach is not a panacea as packet latency can increase, with the need to handle routing imposing a computational overhead on the server.

The server-centric paradigm is currently an area of intensive study with numerous new server-centric DCNs having been proposed and scrutinized, although there is still much to be done before server-centric DCNs make it through to production. Since the advent of DCell and BCube, various server-centric DCNs have been proposed, some of which further restrict themselves to requiring at most two ports per server, with FiConn [28] and DPillar [31] being the most established of this genre. This dual-port restriction is motivated by the fact that many COTS servers presently available for purchase, as well as servers in existing datacenters, have two NIC ports (a primary and a backup port). Dual-port server-centric DCNs are able to utilise such servers without modification, thus making it possible to use some of the more basic equipment (available for purchase or from existing datacenters) in a server-centric DCN and thereby reduce the building costs.

The server-centric DCN architecture provides a versatile design space, as regards the network topology, evidenced perhaps by the sheer number of fairly natural constructions proposed from 2008 to the present. On the other hand, this pool is small relative to the number of interconnection networks found in the literature, *i.e.*, highly structured graphs with good networking properties. One of the challenges of identifying an interconnection network suitable for conversion to a DCN topology, however, lies in the fact that the literature on interconnection networks is focused primarily on graphs whose nodes are homogeneous[1], whereas in both a switch-centric and a server-centric DCN we have server-nodes and switch-nodes which have entirely different operational roles. Some server-centric DCN topologies arise largely from outside the interconnection network literature, *e.g.*, DCell and FiConn, whilst others arise from transformations of well-known interconnection networks, *e.g.*, BCube and DPillar.

The transformations used to obtain BCube and DPillar take advantage of certain sub-structures in the underlying base graphs of the interconnection networks in question (generalized hypercubes and wrapped butterfly networks, respectively) in order to create a server-centric DCN that inherits beneficial networking properties such as having a low diameter and fault-tolerant routing algorithms. The limitation, of course, is that not every prospective base graph has the required sub-structures (cliques and bicliques, respectively, in the cases of BCube and DPillar). New methods of transforming interconnection networks into server-centric DCNs may therefore greatly enlarge the server-centric DCN design space by lowering the structural requirements on potential base graphs.

It is with the construction of new dual-port server-centric DCNs that we are concerned in this paper. In particular, we provide a generic methodology to systematically transform interconnection networks, as base graphs, into dual-port server-centric DCNs, which we refer to as *stellar* DCNs. The stellar transformation is very simple and widely applicable: the edges of the base graph are replaced with paths of length 3 involving two server-nodes each, and the nodes of the base graph become the switch-nodes of the stellar DCN (see Fig. 3). By requiring very little of the base graph in the way of structure, the stellar construction greatly increases the pool of interconnection networks that can potentially serve as blueprints to design dual-port server-centric DCN topologies.

We validate our generic construction in three ways: first, we prove that various networking properties of the base graph are preserved under the stellar transformation; second, we build a library of interconnection networks that suit the stellar transformation; and third, we empirically evaluate GQ*, an instantiation of a stellar DCN whose base graph is a generalized hypercube, against both FiConn and DPillar, and we also compare GQ* and its routing algorithm (inherited from generalized hypercubes) against what might be optimally possible in GQ*. This latter validation demonstrates that not only does our methodology allow us to transport properties from interconnection networks to dual-port DCNs in general, but also that a specific application of it yields a very competitive dual-port DCN in comparison with other well-established dual-port DCNs.

Our empirical results are extremely encouraging. We employ a comprehensive set of performance metrics so as to evaluate network throughput, latency, load balancing capability, fault-tolerance, and cost to build, within the context of all-to-all, many all-to-all, butterfly, random, hot-region, and hot-spot traffic patterns, and we show that GQ* broadly outperforms both FiConn and DPillar as regards these metrics, sometimes significantly so. Highlights of these improvements are as follows. In terms of aggregate bottleneck throughput (a primary metric as regards the evaluation of throughput in an all-to-all context), our DCN GQ* improves upon both FiConn and DPillar (upon the former markedly so). As regards fault-tolerance, our DCN GQ*, with its fault-tolerant routing algorithm *GQ*-routing* (inherited from generalized hypercubes), outperforms DPillar (and its fault-tolerant routing algorithm *DPillarMP* from [31]) and competes with FiConn even when we simulate optimal fault-tolerant routing in FiConn (even though such a fault-tolerant routing algorithm has yet to be exhibited). Not only does *GQ*-routing* (in GQ*) tolerate faults better than the respective routing algorithms in FiConn and DPillar, but when we make around 10% of the links faulty and compare it with the optimal scenario in GQ*, *GQ*-routing* provides around 95% connectivity and generates paths that are, on average, only around 10% longer than the shortest available paths. When we consider load balancing in GQ*, FiConn, and DPillar, with their respective routing algorithms *GQ*-routing, TOR*, and *DPillarSP* and under a variety of traffic patterns, we find that the situation in GQ* is generally improved over that in FiConn and DPillar. As we shall see, DPillar performs particularly poorly except as regards the butterfly and hot-region traffic patterns; indeed, for the hot-region traffic pattern, it performs best. The improved load balancing in GQ* in tandem with the generation of relatively short paths translates to potential latency savings.

However, we have only scratched the surface in terms of what might be possible as regards the translation of high-performance interconnection networks into dual-port server-centric DCNs in that we have applied our generic, stellar construction to only one family of interconnection networks so as to achieve encouraging results. In addition to our experiments, we demonstrate that there are numerous families of interconnection networks to which our construction might be applied. Whilst our results with generalized hypercubes are extremely positive, we feel that the generic nature of our construction has significant potential and scope for further application.

To summarise, the contributions of this paper are as follows:

- We propose the star-replaced server-centric DCN construction as a generic methodology in order to automatically convert graphs and interconnection networks into ‘stellar’ dual-port server-centric DCNs;
- We demonstrate how the properties of the base graph or interconnection network translate so that similar properties are

[1] We disregard the terminal nodes of indirect networks, which are not intrinsic to the topology.

inherited by the stellar DCN (consequently, we can often use existing interconnection networks research);
- We instantiate our stellar transformation using the well-studied generalized hypercube family of interconnection networks to obtain the stellar DCN GQ⋆; and
- We evaluate GQ⋆ against the state-of-the-art dual-port server-centric DCNs FiConn and DPillar and find that it yields excellent comparative performance.

The rest of the paper is organized as follows. In the next section, we give an overview of the design space for dual-port server-centric DCNs, along with related work, before defining our new generic construction in Section 3 and proving that good networking properties of the underlying interconnection network translate to good networking properties of the stellar DCN. We also instantiate our stellar construction in Section 3 so as to generate the DCNs GQ⋆, and in Sections 4 and 5 we describe the methodology of our empirical evaluation and the results of this investigation, respectively. We close with some concluding remarks and suggestions for future work in Section 7. We refer the reader: to [12] for all standard graph-theoretic concepts; to [24,47] for the interplay between graph theory and interconnection network design; and to [9] for an overview of interconnection networks and their implementation for distributed-memory multiprocessors. We implicitly refer to these references throughout.

2. The dual-port server-centric DCN design space

A dual-port server-centric DCN can be built from: COTS servers, each with (at most) two network interface card (NIC) ports; dumb "crossbar" switches; and the cables that connect these hardware components together. We define the capability of a dumb crossbar-switch (henceforth referred to as a switch) as being able to forward an incoming packet to a single port requested in the packet header and to handle all such traffic in a non-blocking manner. Such a switch only ever receives packets destined for servers directly attached to it. It is never the case that two switches in the network are directly connected by a cable, as otherwise a switch would necessarily have some routing capability which by design it does not have.

We take a (primarily) mathematical view of datacenters in order to systematically identify potential DCN topologies, and we abstract a DCN as an undirected graph so as to model only the major hardware components; namely, the servers and switches are abstracted as server-nodes and switch-nodes, respectively, and the interconnecting cables as edges or links, with each link modelling two oppositely-oriented and independent communication channels. As our server-centric DCNs are dual-port, our graphs are such that every server-node has degree at most 2 and the switch-nodes form an independent set in the graph.

2.1. Designing DCNs with good networking properties

There are well-established performance metrics for DCNs and their routing algorithms so that we might evaluate properties such as network throughput, latency, load balancing capability, fault-tolerance, and cost to build (we will return to these metrics later when we outline our methodology, in Section 4, and undertake our empirical analysis, in Section 5). Networks that perform well with respect to these or related metrics are said to have *good networking properties*. Maintaining a diverse pool of potential DCN topologies with good networking properties gives DCN designers greater flexibility. There is already such a pool of interconnection networks, developed over the past 50 years or so, and it is precisely from here that the switch-centric DCN fabrics of layer-2 switch-nodes in fat-trees and related topologies have been adapted (see, *e.g.*, [3,27]).

Adapting interconnection networks to build server-centric DCNs, which necessarily have a more sophisticated arrangement of server-nodes and switch-nodes, however, is more complicated. For example, BCube [19] is built from a generalized hypercube (see Definition 3.1) by replacing the edges of certain cliques, each with a switch-node connected to the nodes of the clique. In doing so, BCube inherits well-known routing algorithms for generalized hypercubes, as well as mean-distance, fault-tolerance, and other good networking properties. DPillar [31], which we discuss in detail in Section 2.4, is built in a similar manner from a wrapped butterfly network (see, *e.g.*, [26]) by replacing bicliques[2] with switch-nodes. The presence of these cliques and bicliques is inherent in the definitions of generalized hypercubes and wrapped butterfly networks, respectively, but are not properties of interconnection networks in general. Furthermore, the dual-port property of DPillar is not by design of the construction, but is a result of the fact that each node in a wrapped butterfly is in exactly two maximal bicliques.

In order to effectively capitalise on a wide range of interconnection networks, for the purpose of server-centric DCN design, we must devise new generic construction methods, similar to those used to construct BCube and DPillar but that do not impose such severe structural requirements on the interconnection network used as the starting point.

2.2. Related work

We briefly survey the origins of the dual-port server-centric DCNs proposed thus far within the literature [21,28–31], referring the reader to the original publications for definitions of topologies not given below. FiConn [28] is an adaptation of DCell and is unrelated to any particular interconnection network. DPillar's origins [31] were discussed above. The topologies HCN and BCN [21] are built by combining a 2-level DCell with another network, later discovered to be related to WK-recursive interconnection networks [11,43]. BCCC [30] is a tailored construction related to BCube and based on cube-connected-cycles and generalized hypercubes. Finally, SWKautz, SWCube, and SWdBruijn [29] employ a subdivision rule similar to ours, but the focus in [29] is not on the (generic) benefits of subdividing interconnection networks as much as it is on the evaluation of those two particular network topologies. The reader is referred to the surveys [23,40] for more on the current DCN landscape.

In Section 5 we compare an instantiation of our construction, namely the dual-port server-centric DCN GQ⋆, to FiConn and DPillar. The rationale for using these DCNs in our evaluation is that they are good representatives of the spectrum of dual-port server-centric DCNs mentioned above: FiConn is a good example of a DCN that includes both server-node-to-server-node and server-node-to-switch-node connections and is somewhat unstructured, whereas DPillar is server-node symmetric[3] [13] and features only server-node-to-switch-node connections. In addition, FiConn is arguably unrelated to any previously known interconnection network topology, whilst DPillar is built from, and inherits some of the properties of, the wrapped butterfly network. Various other dual-port server-centric DCNs lie somewhere between these two extremes. Notice that neither FiConn nor DPillar can be described as an instance of our generalised construction: FiConn has some server-nodes whose only connection is to a solitary switch-node, and in DPillar each server-node is connected only to 2 switch-nodes. We now describe the constructions of the DCNs FiConn and DPillar.

[2] A biclique is a graph formed from two independent sets so that every node in one independent set is joined to every node in the other independent set.

[3] Meaning that for every pair (u, v) of server-nodes, there is an automorphism of the network topology that maps u to v.

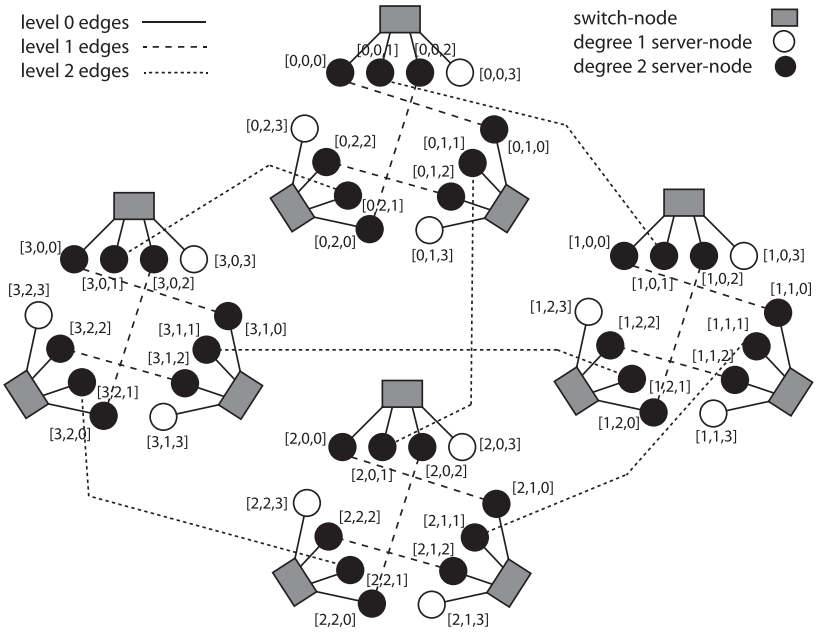

Fig. 1. A visualisation of $\text{FiConn}_{2,4}$.

2.3. The construction of FiConn

We start with FiConn, the first dual-port server-centric DCN to be proposed and, consequently, typically considered the baseline such DCN. For any even $n \geq 2$ and $k \geq 0$, $\text{FiConn}_{k,n}$ [28] is a recursively-defined DCN where k denotes the level of the recursive construction and n the number of server-nodes that are directly connected to a switch-node (so, all switch-nodes have degree n). $\text{FiConn}_{0,n}$ consists of n server-nodes and one switch-node, to which all the server-nodes are connected. Suppose that $\text{FiConn}_{k,n}$ has b server-nodes of degree 1 ($b = n$ when $k = 0$; moreover, no matter what the value of k, b can always be shown to be even). In order to build $\text{FiConn}_{k+1,n}$, we take $\frac{b}{2} + 1$ copies of $\text{FiConn}_{k,n}$ and for every copy we connect one server-node of degree 1 to each of the other $\frac{b}{2}$ copies (these additional links are called level k links). The actual construction of which server-node is connected to which is detailed precisely in [28] ($\text{FiConn}_{2,4}$, as constructed in [28], can be visualised in Fig. 1); in particular, there is a well-defined naming scheme where server-nodes of $\text{FiConn}_{k,n}$ are named as specific k-tuples of integers. In fact, although it is not made clear in [28], there is a multitude of connection schemes realising different versions of FiConn. Note that all of the DCNs we consider in this paper come in parameterized families; so, when we say "the DCN FiConn", what we really mean is "the family of DCNs $\{\text{FiConn}_{k,n}$: $k \geq 0$, $n \geq 2$ is even$\}$".

In [28], two routing algorithms are supplied: *TOR* (traffic-oblivious routing) and *TAR* (traffic-aware routing). *TAR* is intended as a routing algorithm that dynamically adapts routes given changing traffic conditions (it was remarked in [28] that it could be adapted to deal with link or port faults).

2.4. The construction of DPillar

The DCN $\text{DPillar}_{k,n}$ [31], where $n \geq 2$ is even and $k \geq 2$, is such that n denotes the number of ports of a switch-node and k denotes the level of the recursive construction; it can be imagined as k columns of server-nodes and k columns of switch-nodes, arranged alternately on the surface of a cylindrical pillar (see as an example $\text{DPillar}_{3,6}$ in Fig. 2). Each server-node in some server-column is adjacent to 2 switch-nodes, in different adjacent switch-columns. Each server-column has $(\frac{n}{2})^k$ server-nodes, named as $\{0, 1, \ldots, \frac{n}{2} - 1\}^k$, whereas each switch-column has $(\frac{n}{2})^{k-1}$ switch-nodes, named as $\{0, 1, \ldots, \frac{n}{2} - 1\}^{k-1}$. We remark that in the literature, our $\text{DPillar}_{k,n}$ is usually referred to as $\text{DPillar}_{n,k}$. However, we have adopted our notation so as to be consistent with other descriptions of DCNs.

Fix $c \in \{0, 1, \ldots, k-1\}$. The server-nodes in server-columns $c, c+1 \in \{0, 1, \ldots, k-1\}$ (with addition modulo k) are arranged into $(\frac{n}{2})^{k-1}$ groups of n server-nodes so that in server-columns c and $c+1$, the server-nodes in group $(u_{k-1}, \ldots, u_{c+1}, u_{c-1}, \ldots, u_0) \in \{0, 1, \ldots, \frac{n}{2} - 1\}^{k-1}$ are the server-nodes named $\{(u_{k-1}, \ldots, u_{c+1}, i, u_{c-1}, \ldots, u_0) : i \in \{0, 1, \ldots, \frac{n}{2} - 1\}\}$. The adjacencies between switch-nodes and server-nodes are such that any server-node in group $(u_{k-1}, \ldots, u_{c+1}, u_{c-1}, \ldots, u_0)$ in server-columns c and $c+1$ is adjacent to the switch-node of name $(u_{k-1}, \ldots, u_{c+1}, u_{c-1}, \ldots, u_0)$ in switch-column c.

In [31], two routing algorithms are supplied: *DPillarSP* and *DPillarMP*. The former is a single-path routing algorithm and the latter is a multi-path routing algorithm.

While all of the dual-port server-centric DCNs from the literature have merit, it is clear that a generic method of transforming interconnection networks into dual-port server-centric DCNs has not previously been proposed and analysed. Having justified the value in studying the dual-port restriction, and having discussed the benefits of tapping into a large pool of potentially useful topologies, we proceed by presenting our generic construction in detail in the next section.

3. Stellar DCNs: a new generic construction

In this section we present our generic method of transforming interconnection networks into potential dual-port server-centric DCNs. We then describe how networking properties of the DCN, including routing algorithms, are inherited from the interconnection network, and go on to identify a preliminary pool of interconnection networks that particularly suit the stellar transformation. Next, we apply our stellar transformation in detail to generalized hypercubes as a prelude to an extensive empirical evaluation in Sections 4 and 5. The key aspects of our stellar construction are its topological simplicity, its universal applicability,

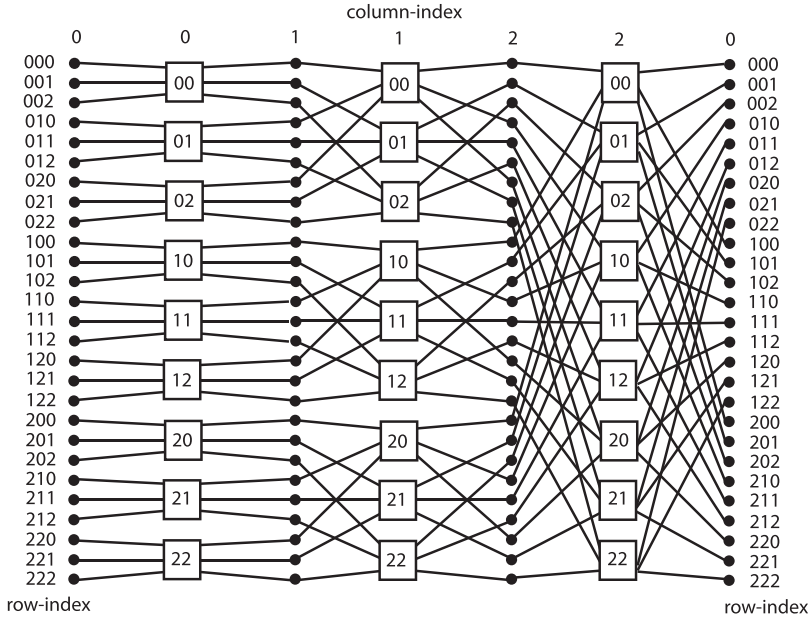

Fig. 2. A visualization of $\mathrm{DPillar}_{3,6}$. Squares represent switch-nodes, whereas dots represent server-nodes. For the sake of simplicity, the left-most and the right-most server-columns are the same (server-column 0).

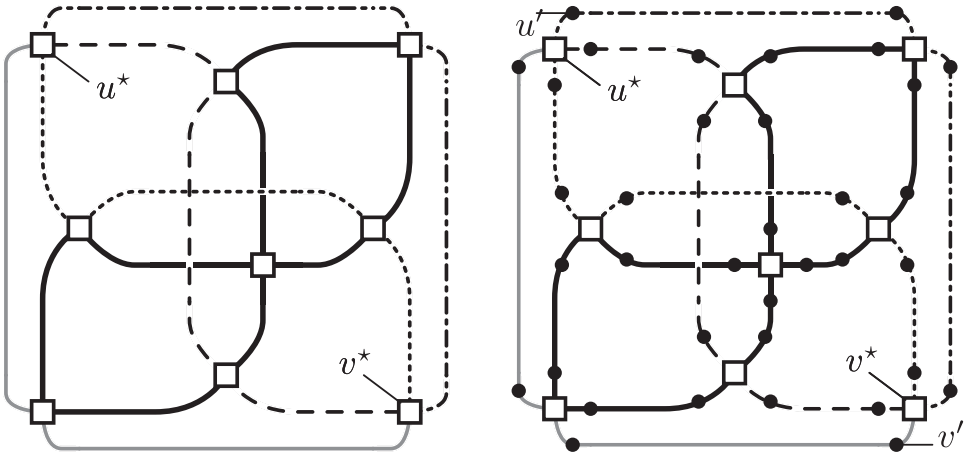

Fig. 3. Transforming 4 paths from u to v in G (*left*) into 4 paths from u' to v' in $G^\star$ (*right*).

and the tight relationship between the interconnection network and the resulting stellar DCN (in a practical networking sense). While we present our stellar construction within a graph-theoretic framework, we end this section by briefly discussing concrete networking aspects of our construction in relation to implementation. We remind the reader that we use [9,24,47] as our sources of information for the definitions and the networking properties of the families of interconnection networks mentioned below; we use these sources implicitly and only cite other sources when pertinent.

3.1. The stellar construction

An *interconnection network* is an undirected graph together with associated routing algorithms, packet-forwarding methodologies, fault-tolerance processes, and so on. However, it suffices for us to abstract an interconnection network as simply a graph $G = (V, E)$ that is undirected and without self-loops.

Let $G = (V, E)$ be any non-trivial connected graph, which we call the *base graph* of our construction. The *stellar* DCN $G^\star$ is obtained from G by placing 2 server-nodes on each link of G and identifying the original nodes of G as switch-nodes (see Fig. 3). We use the term "stellar" as we essentially replace every node of G and its incident links with a "star" subnetwork consisting of a hub switch-node and adjacent server-nodes. Clearly, $G^\star$ has $2|E|$ server-nodes and $|V|$ switch-nodes, with the degree of every server-node being 2 and the degree of every switch-node being identical to the degree of the corresponding node in G.

We propose placing 2 server-nodes on every link of G so as to ensure: uniformity, in that every server-node is adjacent to exactly 1 server-node and exactly 1 switch-node (uniformity, and its stronger counterpart symmetry, are widely accepted as beneficial properties in general interconnection networks); that there are no links incident only with switch-nodes (as this would violate the server-centric restriction, discussed in the opening paragraph of Section 2); and that we can incorporate as many server-nodes as needed within the construction (subject to the other conditions). In fact, *any* DCN in which every server-node is adjacent to exactly 1 server-node and 1 switch-node and where every switch-node is only adjacent to server-nodes can be realised as a stellar DCN $G^\star$, for some base graph G. In addition, the stellar transformation can be applied to *any* (non-trivial connected) base graph; that is, the

transformation does not rely on any non-trivial structural properties of the base graph.

3.2. Topological properties of stellar DCNs

The principal decision that must be taken when constructing a stellar DCN is in choosing an appropriate base graph G. The good networking properties discussed in Section 2.1 are underpinned by several graph-theoretic properties that are preserved under the stellar transformation: for example, low diameter, high connectivity, and efficient routing algorithms in the base graph G translate more-or-less directly into good (theoretical) networking properties of the stellar graph $G^\star$, as we now discuss. The DCN designer, having specific performance targets in mind, can use this information to facilitate the selection of a base graph G that meets the requirements of the desired stellar DCN.

3.2.1. Paths

A useful aspect of our construction is as regards the transformation of paths in G to paths in $G^\star$. As is usual in the analysis of server-centric DCNs (see, e.g., [19–21,28]), we measure a server-node-to-server-node path P by its *hop-length*, defined as one less than the number of server-nodes in P. Accordingly, we prefix other path-length-related measures with hop-; for example, the hop-length of a shortest path joining two given server-nodes in $G^\star$ is the *hop-distance* between the two server-nodes, and the *hop-diameter* of a server-centric DCN is the maximum over the hop-distances for every possible pair of server-nodes. Let $G = (V, E)$ be a connected graph and let $u, v \in V$. Let $u^\star$ and $v^\star$ be the switch-nodes of $G^\star$ corresponding to u and v, respectively. Let u' and v' be server-node neighbours of $u^\star$ and $v^\star$, respectively, in $G^\star$. Each (u,v)-path P in G, of length m, corresponds uniquely to a (u',v')-path in $G^\star$ of hop-length $2m-1$, $2m$, or $2m+1$. The details are straightforward.

3.2.2. Path-based sub-structures

The transformation of paths in G to paths in $G^\star$ is the basis for the transfer of potentially useful sub-structures in G to $G^\star$ so as to yield good DCN properties. Any useful (path-based) sub-structure in G, such as a spanning tree, a set of node-disjoint paths, or a Hamiltonian cycle, corresponds uniquely to a closely related sub-structure in $G^\star$. Swathes of research papers have uncovered these sub-structures in interconnection networks, and the stellar construction facilitates their usage in dual-port server-centric DCNs. It is impossible to cover this entire topic here, but we describe how a few of the more commonly sought-after sub-structures behave under the stellar transformation.

Foremost are internally node-disjoint paths, associated with fault-tolerance and load balancing. As the degree of any server-node in $G^\star$ is 2, one cannot hope to obtain more than 2 internally node-disjoint paths joining any 2 distinct server-nodes of $G^\star$. However, a set of c internally node-disjoint (u,v)-paths in G corresponds uniquely to a set of c internally (server- and switch-) node-disjoint $(u^\star,v^\star)$-paths in $G^\star$, where $u,v,u^\star$, $v^\star$, u', and v' are as defined above. This provides a set of c (u',v')-paths in $G^\star$, called *parallel paths*, that are internally node-disjoint apart from possibly $u^\star$ and $v^\star$ (see Fig. 3). It is trivial to show that the minimum number of parallel paths between any pair of server-nodes, not connected to the same switch-node, in $G^\star$ is equal to the connectivity of G.

By reasoning as above, it is easy to see that a set of c edge-disjoint (u,v)-paths in G becomes a set of c internally server-node-disjoint (u',v')-paths in $G^\star$, with $u,v,u^\star$, $v^\star$, u', and v' defined as above; we shall call these *server-parallel paths*. The implication is that as any two of these paths share only the links $(u',u^\star)$ and $(v^\star,v')$, a high value of c may be leveraged to alleviate network traffic congestion as well as fortify the network against server-node failures.

Table 1
Transformation of networking properties of a connected graph G.

Property	$G = (V, E)$	$G^\star$
Nodes/nodes	$\|V\|$	$\|V\|$ switch-nodes $2\|E\|$ server-nodes
Node degree/switch-node degree	d	d
Edges/links	$\|E\|$	$3\|E\|$ (bidirectional)
Path-length/hop-length	x	$2x-1 \le \cdot \le 2x+1$
Diameter/hop-diameter	D	$2D-1, 2D,$ or $2D+1$
Internally-disjoint paths/parallel paths	κ	κ
Edge-disjoint paths/server-parallel paths	γ	γ

On a more abstract level, consider any connected spanning sub-structure H of G, such as a Hamiltonian cycle or a spanning tree. Let $H^\star$ be the corresponding sub-structure in $G^\star$ (under the path-to-path mapping described above) and observe that each edge of G not contained in H corresponds to two adjacent server-nodes in $G^\star$ not contained in $H^\star$. On the other hand, every server-node not in $H^\star$ is exactly one hop away from a server-node that is in $H^\star$; so within an additive factor of one hop, $H^\star$ is just as "useful" in $G^\star$ as H is in G. In fact, if H is a spanning tree in G then we can extend $H^\star$ in $G^\star$ by augmenting it with pendant edges from switch-nodes so that what results is a spanning tree in $G^\star$ containing *all* server-nodes of $G^\star$ (and not just those in the original $H^\star$). By the same principle, non-spanning sub-structures of G, such as those used in one-to-many, many-to-many, and many-to-one communication patterns, also translate to useful sub-structures in $G^\star$.

We summarise the relationship between properties of G and $G^\star$ that we have discussed so far in Table 1 where corresponding properties for G and $G^\star$ are detailed. It should now be apparent that the simplicity of our stellar transformation enables us to import good networking properties from our base graphs to our stellar DCNs where these properties are crucial to the efficacy of a DCN.

We close this sub-section with a brief discussion of the transferral of routing algorithms under the stellar transformation and of the as yet unexplored potential of the stellar transformation as regards other important aspects of DCNs. A routing algorithm for an interconnection network G is effectively concerned with an efficient computation over some communication sub-structures. For example, in the case of unicast routing from u to v, we may compute one or more (u,v)-paths (and route packets over them), or for a broadcast we may compute one or more spanning trees. Routing algorithms can be executed at the source node or in a distributed fashion, and they can be deterministic or non-deterministic; whatsoever the process, the resulting output is a communication sub-structure over which packets are sent from node to node. We discussed above the correspondence between communication sub-structures in G and those in $G^\star$; we now observe that, in addition, any routing algorithm on G can be simulated on $G^\star$ with the same time complexity. We leave the details to the reader (but we will instantiate this later when we build the stellar DCNs GQ*).

While we undertake an extensive investigation into the concept of a stellar transformation in this paper, there are numerous other aspects of DCNs that intuitively might benefit from the stellar construction. Consider the *bisection width* which is the minimum number of links that must be removed to partition the network into two (roughly) equal halves. The bisection width is primarily used in order to evaluate throughput but is also relevant to fault-tolerance (we say a little more about its relevance to throughput and our own experiments at the end of Section 4.5). Calculating the bisection width of networks is notoriously difficult; algorithmically, it is **NP**-hard in general, and the precise analyti-

cal value for many relatively simple networks remains unknown (see [4] and the references therein). Given the nature of the stellar construction, it seems plausible that we might be able to use known bisection width values of the base graph to analytically determine (estimates of) the bisection width of the stellar DCN. In addition, consider the *over-subscription ratio*, defined in [3] as 'the ratio of the worst-case achievable aggregate bandwidth among the end hosts to the total bisection bandwidth of a particular communication topology'. Roughly speaking, over-subscription is concerned with the factor by which the available throughput of links or servers falls below the required throughput (on occasion, it is used imprecisely within the literature so that its intended definition is unclear). An important practical aspect of a DCN is that there should be some control over the over-subscription ratio. Our stellar transformation is parameterized by the base graph G, and the choice of this parameter provides considerable scope as to the choice of bisection width (which controls the over-subscription ratio, according to the above definition). The key point is that as well as some demonstrably provable advantages of the stellar transformation (in relation to hop-length, hop-diameter, paths, routing algorithms, and so on), the nature of the construction makes many other advantages plausible too. We say more about bisection width and over-subscription in Sections 6 and 7.

3.3. A pool of suitable base graphs

So far, we have referred to an interconnection network as a solitary object. However, interconnection networks (almost always) come in families where there are parameters the values for which precisely delineate the family members. For example the hypercube Q_n is parameterized by the degree n of the nodes, and so really by "the hypercube Q_n" we mean "the family of hypercubes $\{Q_n : n = 1, 2, \ldots, \}$". For the rest of this sub-section we will be precise and speak of families of interconnection networks as we need to focus on the parameters involved. To ease understanding, when there is more than one parameter involved in some definition of a family of interconnection networks and these parameters appear as subscripts or in tuples in the denotation, we list parameters relating to the dimension of tuples or the depth of recursion first with parameters referring to the size of some component-set coming afterwards (we have done this so far with $\text{FiConn}_{k,n}$ and $\text{DPillar}_{k,n}$). We remark that this is sometimes at odds with standard practice in the literature.

We validate our claim that many families of interconnection networks suit the stellar construction by highlighting several that, first, have parameters flexible enough to yield interconnection networks of varying and appropriate size and degree, and, second, are known to possess good networking properties. The first goal is to identify families of interconnection networks that have suitable combinations of degree and size, bearing in mind that today's DCN COTS switches have up to tens of ports, with 48 being typical, while conceivable (but not necessarily in production) sizes of DCNs range from tens of server-nodes up to, perhaps, 5 million in the near future. An illustration of a family of interconnection networks *lacking* this flexibility is the family of hypercubes, where the hypercube Q_n necessarily has 2^n nodes when the degree is fixed at n; this translates to a stellar DCN with n-port switch-nodes and, necessarily, $n2^n$ server-nodes. As such, there is a lack of flexibility, in terms of the possible numbers of server-nodes, and if we were to build our stellar DCNs using commodity switches with 48 ports then we would have to have 48×2^{48} servers which is clearly impossible. Another illustration of a family of interconnection networks lacking flexibility is the family of cube-connected cycles $\{CCC(n)\colon n \geq 3\}$, where $CCC(n)$ is obtained from a hypercube Q_n via a transformation similar to our stellar transformation: 2 new nodes are placed on each edge; the new nodes adjacent to some old node are joined (systematically) in a cycle of length n; and the old nodes, and any adjacent edges, are removed. So, $CCC(n)$ is regular of degree 3 and consequently unsuitable for our stellar transformation.

We now look at some families of interconnection networks that *are* suitable for our stellar transformation. It is too much to list all of the good networking properties of the interconnection networks discussed below. However, it should be remembered that, from above, any path, path-based sub-structure, and routing algorithm is immediately inherited by the stellar DCN; consequently, we focus on the flexibility of the parameterized definition in what follows and refer the reader to other sources (including [9,24,47]) for more details as regards good networking properties. Besides: the fact that these families of interconnection networks have featured so strongly within the research literature is testament to their good networking properties. Also, the families of interconnection networks mentioned below are simply illustrations of interconnection networks for which our stellar transformation has potential and there are many others not mentioned (see, *e.g.*, [24,47]).

Tori (also known as toroidal meshes) have been widely studied as interconnection networks; indeed, tori form the interconnection networks of a range of distributed-memory multiprocessor computers (see, *e.g.*, [9]). The uniform version of a torus is the n-ary k-cube $Q_{k,n}$, where $k \geq 1$ and $n \geq 3$, whose node-set is $\{0, 1, \ldots, n-1\}^k$ and where there is an edge joining two nodes if, and only if, the nodes differ in exactly one component and the values in this component differ by 1 modulo n; hence, $Q_{k,n}$ has n^k nodes and kn^k edges, and every node has degree $2k$. There is some, though limited, scope for using n-ary k-cubes in our stellar construction. For example, if we use switch-nodes with 16 ports to build our DCN then this means that $k = 8$; choosing $n = 3$, 4, or 5 results in our stellar DCN having 104,976 server-nodes, 1,048,576 server-nodes, or 6,250,000 server-nodes, respectively. We get more variation if we allow the sets of values in different components to differ; that is, we use mixed-radix tori. However, it is not really feasible to use switch-nodes with more than 16 ports in a torus-based stellar construction because of how torus topologies scale; not unless one were to use, for example, switch-nodes with 64 ports in order to implement 4 switch-nodes with 16 ports. This is an interesting and as yet unexplored methodology that we expand upon in our conclusions.

Circulant graphs have been studied extensively in a networking context, where they are often called multi-loop networks. Let S be a set of integers, called *jumps*, with $1 \leq s \leq \lfloor N/2 \rfloor$, for each $s \in S$, and where $N \geq 2$. A circulant $G(N; S)$ has node set $\{0, 1, \ldots, N-1\}$, where node i is connected to nodes $i \pm s \pmod N$, for each $s \in S$. A circulant has N nodes and at most $N|S|$ edges, and the degree of every node is approximately $2|S|$ (depending upon the relative values of N and the integers in S); consequently, the parameters provide significant general flexibility. Illustrations of good networking properties of circulants can be found in, for example, [7,25,34].

The wrapped butterfly network $BF(k,n)$ can be obtained from $\text{DPillar}_{k,n}$ by replacing all switch-nodes with bicliques (joining server-nodes in adjacent columns); consequently, $B(k,n)$ has $k(\frac{n}{2})^k$ nodes and $k(\frac{n}{2})^{k+1}$ edges, and each node has degree n. Thus, by varying k and n, there is reasonable scope for flexibility in terms of the sizes of stellar DCNs. Illustrations of the good networking properties of wrapped butterfly networks can be found in, for example, [17,44]. Note that transforming a wrapped butterfly network to obtain DPillar is different to transforming it according to the stellar transformation; the two resulting DCNs are combinatorially very distinct.

The de Bruijn digraph $dB(k,n)$, where $k \geq 1$ and $n \geq 2$ is even, is a graph with node-set $\{0, 1, \ldots, \frac{n}{2} - 1\}^k$. There is a directed edge from $(s_{k-1}, s_{k-2}, \ldots, s_0)$ to $(s_{k-2}, s_{k-3}, \ldots, s_0, \alpha)$, for

each $\alpha \in \{0, 1, \ldots, \frac{n}{2} - 1\}$. Undirected de Bruijn graphs are obtained by regarding all directed edges as undirected and removing self-loops and multiple edges; such graphs are not regular but nearly so, with most of the $(\frac{n}{2})^k$ nodes having degree n although some have degree $n-1$ or $n-2$. Consequently, by varying the values of k and n, there is good flexibility in terms of the sizes of stellar DCNs. Illustrations of the good networking properties of de Bruijn graphs can be found in, for example, [15,38]. Note that de Bruijn graphs have been studied as server-centric DCNs in [37] but these DCNs are not dual-port.

The arrangement graph $A_{k,n}$, where $n \geq 2$ and $1 \leq k \leq n-1$, has node-set $\{(s_{k-1}, s_{k-2}, \ldots, s_1, s_0) : s_i \in \{0, 1, \ldots, n-1\}, s_i \neq s_j, i, j = 0, 1, \ldots, k-1\}$. There is an edge joining two nodes if, and only if, the nodes are identical in all but one component. Hence, the arrangement graph $A_{k,n}$ has $\frac{n!}{(n-k)!}$ nodes and $\frac{k(n-k)n!}{2(n-k)!}$ edges, and is regular of degree $k(n-k)$. The family of arrangement graphs includes the well-known star graphs as a sub-family, and there is clearly considerable flexibility in their degree and size.

3.4. The stellar DCNs GQ⋆

Having hinted that there are various families of interconnection networks to which our stellar transformation might sensibly be applied, we now apply the stellar transformation to one specific family in detail: the family of generalized hypercubes [6]. We provide below more details as regards the topological properties of and routing algorithms for generalized hypercubes as we will use these properties and algorithms in our experiments in Sections 4 and 5. We choose generalized hypercubes because of their flexibility as regards the stellar construction, their good networking properties, and the fact that they have already featured in DCN design as templates for BCube.

Definition 3.1. The *generalized hypercube* $GQ_{k,n}$, where $k \geq 1$ and $n \geq 2$, has node-set $\{0, 1, \ldots, n-1\}^k$ and there is an edge joining two nodes if, and only if, the names of the two nodes differ in exactly one component.

Consequently, $GQ_{k,n}$ has n^k nodes and $\frac{1}{2}k(n-1)n^k$ edges, and is regular of degree $k(n-1)$. Also, $GQ_{k,n}$ has diameter k and connectivity $k(n-1)$. Hence, $\mathrm{GQ}^{\star}_{k,n}$ has hop-diameter $2k+1$ and there are $k(n-1)$ parallel paths between any two distinct server-nodes.

Suppose that we wished to use 48-port switch-nodes (and utilize all switch-ports) in a stellar DCN built from $GQ_{k,n}$. We might choose (k,n) as (2, 25), (3, 17), or (4, 13) with the result that the number of server-nodes is 30,000 for $\mathrm{GQ}^{\star}_{2,25}$, 235,824 for $\mathrm{GQ}^{\star}_{3,17}$, or 1,370,928 for $\mathrm{GQ}^{\star}_{4,13}$, respectively (of course, we can vary this number of server-nodes if we do not use all switch-ports or if we use switch-nodes with less than 48 ports).

The stellar construction allows us to transform existing routing algorithms for the base graph $GQ_{k,n}$ into routing algorithms for $\mathrm{GQ}^{\star}_{k,n}$. We describe this process using the routing algorithms for $GQ_{k,n}$ surveyed in [48]. Let $u = (u_{k-1}, u_{k-2}, \ldots, u_0)$ and $v = (v_{k-1}, v_{k-2}, \ldots, v_0)$ be two distinct nodes of $GQ_{k,n}$. The basic routing algorithm for $GQ_{k,n}$ is *dimension-order* (or *e-cube*) routing where the path from u to v is constructed by sequentially replacing each u_i by v_i, for some predetermined ordering of the coordinates, say $i = 0, 1 \ldots, k-1$. As we mentioned above, dimension-order routing translates into a shortest-path routing algorithm for $\mathrm{GQ}^{\star}_{k,n}$ with unchanged time complexity, namely $O(k)$.

We introduce a fault-tolerant mechanism called *intra-dimensional* routing by allowing the path to replace u_i by v_i in two steps, using a *local proxy*, rather than in one step, as described in dimension-order routing. Suppose, for example, that one of the edges in the dimension-order route from u to v is faulty; say, the one from $u = (u_{k-1}, u_{k-2}, \ldots, u_1, u_0)$ to $x = (u_{k-1}, u_{k-2}, \ldots, u_1, v_0)$ (assuming that u_0 and v_0 are distinct). In this case we can try to hop from u to $(u_{k-1}, u_{k-2}, \ldots, u_1, x_0)$, where $u_0 \neq x_0 \neq v_0$, and then to x.

Inter-dimensional routing is a routing algorithm that extends intra-dimensional routing so that if intra-dimensional routing fails, because a local proxy within a specific dimension cannot be used to re-route round a faulty link, an alternative dimension is chosen. For example, suppose that in $GQ_{k,n}$ intra-dimensional routing has successfully built a route over dimensions 1 and 2 but has failed to re-route via a local proxy in dimension 3. We might try and build the route instead over dimension 4 and then return and try again with dimension 3. Note that if a non-trivial path extension was made in dimension 4 then this yields an entirely different locality within $\mathrm{GQ}_{k,n}$ when trying again over dimension 3.

However, in our upcoming experiments we implement the most extensive fault-tolerant, inter-dimensional routing algorithm possible, called *GQ⋆-routing*, for the stellar DCN $\mathrm{GQ}^{\star}_{k,n}$, whereby we perform a depth-first search of the dimensions and we use intra-dimensional routing to cross each dimension wherever necessary (and possible). In addition, if *GQ⋆-routing* fails to route directly in this fashion then it attempts four more times to route (as above) from the source to a randomly chosen server-node, and from there to the destination. We have chosen to make this extensive search of possible routes in order to test the maximum capability of *GQ⋆-routing*; however, we expect that in practice the best performance will be obtained by limiting the search in order to avoid certain worst-case scenarios. The precise implementation details of *GQ⋆-routing* can be found in the software release of INRFlow [14] (see Section 4.6). Finally, it is easy to see that *GQ⋆-routing* can be implemented as a distributed algorithm if a small amount of extra header information is attached to a path-probing packet, similarly to the suggestion in [28] for implementing *TAR* (Traffic Aware Routing) in FiConn.

3.5. Implementing stellar DCNs

Implementing the software suite from scratch would require a software infrastructure that supports through-server end-to-end communications. This could be implemented either on top of the transport layer (TCP) so as to simplify development, since most network-level mechanisms (congestion control, fault-tolerance, quality of service) would be provided by the lower layers. Alternatively, it could be implemented on top of the data-link layer to improve the performance, since a lower protocol stack will result in the faster processing of packets. The latter would require a much higher implementation effort in order to deal with congestion and reliability issues. At any rate, the design and development of a software suite for server-centric DCNs is outside the scope of this paper, but may be considered in the future.

4. Methodology

The good networking properties discussed in Section 2.1 guide our evaluation methodology; they are network throughput, latency, load balancing capability, fault-tolerance, and cost to build. These properties are reflected through performance metrics, and in this section we explain how we use aggregate bottleneck throughput, distance metrics, connectivity, and paths and their congestion, combined with a selection of traffic patterns, in order to evaluate the performance of our DCNs and routing algorithms. In particular, we describe and justify the use of our simulation tool in Section 4.6.

Our methodological framework is as follows. First, we take the position, similar to Popa et al. [37], that the cost of a network is of fundamental importance. No matter what purpose a network is intended for, the primary objective is to maximise the return on

Table 2
Basic properties of the selected DCNs.

Topology	$GQ^{\star}_{3,10}$	$GQ^{\star}_{4,6}$	$FiConn_{2,24}$	$DPillar_{4,18}$
Server-nodes	27,000	25,920	24,648	26,244
Switch-nodes	1000	1296	1027	2916
Switch-ports	27	20	24	18
Links	81,000	77,760	67,782	104,976
Diameter	7	9	7	7
Parallel paths	27	20	unknown	9 (see [31])

the cost of a DCN. While there are several elements that factor into the cost of a DCN, including operational costs, our concern is with the capital costs of purchasing and installing the components we are modelling: servers, switches, and cables. Having calculated these costs (in Section 4.1 below), where appropriate (in our evaluation in Section 5.1) we normalise with respect to cost and proceed by both quantitatively and qualitatively interpreting the resulting multi-dimensional metric-based comparison. Subsequently, (from Section 5.2 onwards) we focus on 4 carefully chosen DCNs, namely $GQ^{\star}_{3,10}$, $GQ^{\star}_{4,6}$, $FiConn_{2,24}$, and $DPillar_{4,18}$, and evaluate these DCNs in some detail. We have selected these DCNs as their properties are relevant to the construction of large-scale DCNs: they each have around 25,000 server-nodes and use switch-nodes of around 24 ports. Table 2 details some of their topological properties.

4.1. Network cost

We follow Popa et al. [37] and assume that the cost of a switch is proportional to its radix (this is justified in [37] for switches of radix up to around 100–150 ports). Let c_s be the cost of a server, let c_p be the cost of a switch-port, and let c_c be the average cost of a cable. We make the simplifying assumption that the average cost of a cable c_c is uniform across DCNs with N servers within the families $GQ^{\star}$, FiConn, and DPillar, and, furthermore, that the average cost of a cable connected only to servers is similar to that of a cable connected to a switch. Thus, the cost of a DCN $GQ^{\star}$ with N server-nodes is $N(c_p + c_c + c_s + \frac{c_c}{2})$; the cost of a DCN $FiConn_{k,n}$ with N server-nodes is $N(c_p + c_c + c_s + \frac{c_c}{2} - \frac{c_c}{2^{k+1}})$, since it contains $\frac{N}{2^k}$ server-nodes of degree 1 [28]; and the cost of a DCN DPillar with N server-nodes is $N(2(c_p + c_c) + c_s)$. Next, we express $c_p = \rho c_s$ and $c_c = \gamma c_s$ so that the costs of the server-nodes become $Nc_s(\rho + \gamma + 1 + \frac{\gamma}{2})$, $Nc_s(\rho + \gamma + 1 + \frac{\gamma}{2} - \frac{\gamma}{2^{k+1}})$, and $Nc_s(2(\rho + \gamma) + 1)$, respectively. A rough estimate is that realistic values for ρ lie in the range [0.01, 0.1], and that realistic values for γ lie in the range [0.01, 0.6]; we choose the ranges conservatively because there is great variation in the cost of components, *e.g.*, between copper and optical cables, as well as how we account for the labour involved in installing them. Our estimates stem from [37] where a typical server is quoted at around $2500, a typical cable at around $50, and the cost of a switch-port at around $100, giving $\rho = 0.04$ and $\gamma = 0.02$. Consequently, we normalise with respect to the aggregated component cost per server-node in $GQ^{\star}$, letting $c_s(\rho + \gamma + 1 + \frac{\gamma}{2}) = 1$, and plot component costs per server-node against γ in Fig. 4, for the representative selection $\rho \in \{0.01, 0.02, 0.4, 0.8, 1.6\}$; in Fig. 4, there is one graph for each DCN and for each of the 5 values for ρ, with the 5 graphs corresponding to FiConn being almost indistinguishable from one another. The upshot is that the higher the value for ρ, the higher the cost of DPillar, and for the specific choices of ρ and γ mentioned above, DPillar could be up to 20% more expensive and FiConn around 4% less expensive than $GQ^{\star}$ when all DCNs have the same number of server-nodes. Perhaps the most realistic values of ρ and γ, however, yield a DPillar that is only about 10% more expensive and FiConn that is only about 2% less expensive.

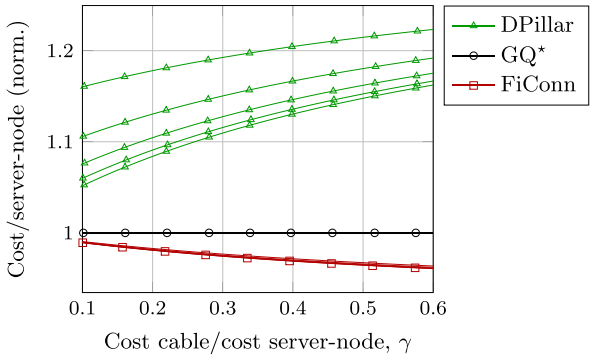

Fig. 4. The component costs per server-node of FiConn and DPillar, relative to that of $GQ^{\star}$, for $\rho \in \{0.01, 0.02, 0.4, 0.8, 1.6\}$.

4.2. Hop-distance metrics

The number of servers a packet flow needs to travel through significantly affects the flow's latency. In addition, for each server on the path, the compute and memory overheads are impacted upon: in a server-centric DCN (with currently available COTS hardware), the whole of the protocol stack, up to the application level, needs to be processed at each server which can make message transmission noticeably slower than in a switch-centric network where lower layers of the protocol stack are employed and use optimised implementations.

The paths over which flows travel are computed by routing algorithms, and it may not be the case that shortest-paths are achievable by available routing algorithms and without global fault-knowledge; large-scale networks like DCNs are typically restricted to routing algorithms that use only local knowledge of fault locations. As such, the performance of the routing algorithm is perhaps more important than the hop-diameter or mean hop-distance of the topology itself. Therefore, we use distance-related metrics that reveal the performance of the topology and the routing algorithm combined, namely *routed hop-diameter* and *routed mean hop-distance*, as well as for the topology alone (where appropriate), namely hop-diameter and mean hop-distance (see Section 2.1). This allows us to (more realistically) assess both the potential of the topologies and the actual performance that can be extracted from them when implemented with currently available routing algorithms.

4.3. Aggregate bottleneck throughput

The *aggregate bottleneck throughput* (*ABT*) is a metric introduced in [19] and is of primary interest to DCN designers due to its suitability for evaluating the worst-case throughput in the all-to-all traffic pattern, which is extremely significant in the context of DCNs (see Section 4.5). The reasoning behind ABT is that the performance of an all-to-all operation is limited by its slowest flow, *i.e.*, the flow with the lowest throughput. The ABT is defined as the total number of flows times the throughput of the *bottleneck flow*; that is, of the *bottleneck link* sustaining the most flows (here, a link is a directed link (x,y); the link (y,x) might sustain a different number of flows). Formally, the ABT of a network of size N is equal to

$$\frac{N(N-1)b}{F} \tag{1}$$

where F is the number of flows in the bottleneck link and b is the bandwidth of a link (which we simply assume to be 1 throughout).

In our experiments, the bottleneck flow is determined experimentally using the implementations of actual routing algorithms;

this is atypical of ABT calculations (*e.g.*, see [32]), where ordinarily shortest-paths are used, but our approach facilitates a more realistic evaluation. We measure ABT using *GQ*-routing* for GQ*, *TOR* for FiConn, and *DPillarSP* for *DPillar*, assuming $N(N-1)$ flows and a bandwidth of 1 unit per directional link, where N is the number of server-nodes. Since datacenters are most commonly used as a stream processing platform, and are therefore bandwidth limited, this is an extremely important performance metric in the context of DCNs. Given that ABT is only defined in the context of all-to-all communications, for other traffic patterns we focus on the number of flows in the bottleneck as an indicator of congestion propensity (we say more about these traffic patterns and our experiments in Section 4.5).

We should explain our choice of routing algorithm in FiConn and DPillar as regards our ABT analysis. In [28], it was shown that *TOR* yields better performance for all-to-all traffic patterns than *TAR*. In [31], the all-to-all analysis (actually, it is a many all-to-all analysis) showed that *DPillarSP* performs better than *DPillarMP*. We have chosen *TOR* and *DPillarSP* so as not to disadvantage FiConn and DPillar when we compare against GQ* and *GQ*-routing*.

4.4. Fault-tolerance

High reliability is of the utmost importance in datacenters, as it impacts upon the business volume that can be attracted and sustained. When scaling out to tens of thousands of servers or more, failures are common, with the mean-time between failures (MTBF) being as short as hours or even minutes. As an example, consider a datacenter with 25,000 servers, 1000 switches, and 75,000 links, each with an optimistic average lifespan of 5 years. Based upon a very rough estimate that the number of elements divided by the average lifespan results in the numbers of failures per day, the system will have an average of about 13 server faults per day, 40 link faults per day, and 1 switch fault every 2 days. In other words, failures are ubiquitous and so the DCN should be able to deal with them in order to remain competitively operational. Any network whose performance degrades rapidly with the number of failures is unacceptable, even if it does provide the best performance in a fault-free environment.

We investigate how network-level failures affect *routed connectivity*, defined as the proportion of server-node-pairs that remain connected by a path computable by a given routing algorithm, as well as how they affect routed mean hop-distance. Our study focuses on uniform random link failures in particular, because both server-node and switch-node failures induce link failures, and also because the sheer number of links (and NICs) in a DCN implies that link-failures will be the most common event. A more detailed study of failure events will be conducted in follow-up research, in which we will consider correlated link, server-node, and switch-node failures. We consider failure configurations with up to a 15% network degradation, where we randomly select, with uniform probability, 15% of the links to have a fault. Furthermore, we consider only bidirectional failures, *i.e.*, where links will either work in both directions or in neither. The rationale for this is that the bidirectional link-failure model is more realistic than the unidirectional one: failures affecting the whole of a link (*e.g.*, NIC failure, unplugged or cut link, or switch-port failure) are more frequent than the fine-grained failures that would affect a single direction. In addition, once unidirectional faults have been detected they will typically be dealt with by disabling the other direction of the failed link (according to the IEEE 802.3ah EFM-OAM standard). We also investigate *unrouted connectivity*, namely the proportion of source-destination pairs, from some given set of pairs, that are connected by a path, as ascertained by BFS. Note that BFS will always find a path if it exists (even in the presence of faults), whilst most routing algorithms do not provide this guarantee.

As regards routed connectivity and routed mean hop-distance, we consider GQ* with *GQ*-routing*, FiConn with breadth-first search (BFS), and DPillar with *DPillarMP*. Again, we explain our choice of routing algorithms. As regards FiConn, *TAR* is a distributed "heuristic" algorithm devised so as to improve network load balancing with bursty and irregular traffic patterns, and was neither optimised for nor tested on outright faulty links. In addition, *TAR* computes paths that are 15–30% longer in these scenarios than *TOR* does. However, *TOR* is not fault-tolerant and so we simply use BFS. In short, we have given FiConn preferential treatment (this makes the performance of GQ* against FiConn, described in Section 5.2, all the more impressive). As regards DPillar, *DPillarMP* is fault-tolerant whereas *DPillarSP* is not.

4.5. Traffic patterns

We now describe the traffic patterns used in our evaluation, the primary one being the all-to-all traffic pattern. All-to-all communications are extremely relevant as they are intrinsic to MapReduce, the preferred paradigm for data-oriented application development; see, for example, [10,32,45]. In addition, all-to-all can be considered a worst-case traffic pattern for two reasons: (*a*) the lack of spatial locality; and (*b*) the high levels of contention for the use of resources.

Our second set of experiments focuses on specific networks hosting around 25,000 server-nodes and evaluates them with a wider collection of traffic patterns; we use the routing algorithms *GQ*-routing, TOR*, and *DPillarSP*. Apart from all-to-all, we also consider the five other traffic patterns many all-to-all, butterfly, random, hot-region, and hot-spot. In *many all-to-all*, the network is split into disjoint groups of a fixed number of server-nodes with server-nodes within a group performing an all-to-all operation. Our evaluation shows results for groups of 1000 server-nodes but these results are consistent with ones for groups of sizes 500 and 5000. This workload is less demanding than the system-wide all-to-all, but can still generate a great deal of congestion. It aims to emulate a typical tenanted cloud datacenter in which there are many independent applications running concurrently. We assume a typical topology-agnostic scheduler and randomly assign server-nodes to groups. The *butterfly* traffic pattern is a "logarithmic implementation" of a pattern such as all-to-all as each server-node only communicates with other server-nodes at hop-distance 2^k, for each $k \in \{0, \ldots, \lceil log(N) \rceil - 1\}$ (see [35] for more details). This workload significantly reduces the overall utilization of the network when compared with the all-to-all traffic pattern and aims to evaluate the behaviour of networks when the traffic pattern is well-structured. We consider a *random* traffic pattern in which we generate one million flows (we also studied other numbers of flows but the results turn out to be very similar to those with one million flows). For each flow, the source and destination are selected uniformly at random. Our *hot-region* traffic pattern is such that we generate traffic so that $\frac{1}{4}$ of the traffic goes to $\frac{1}{8}$ of the server-nodes, with the rest uniform. Finally, flows in our *hot-spot* traffic pattern are generated uniformly at random with the exception that a predetermined hot-spot server-node is $100\times$ more likely to be the destination of each flow than any other given server-node. These additional collections of experiments provide further insights into the performance achievable with each of the networks and allow a more detailed evaluation of propensity to congestion, load balancing, and latency. Fig. 5 depicts example trace matrices for the traffic patterns many all-to-all, butterfly, hot-region, and hot-spot in a DCN with 96 server-nodes.

We close with a remark related to throughput and the role of the bisection width. The bisection width is used to obtain a upper bound on the throughput of a bisection channel, under the assumption of perfectly distributed, uniform traffic; see, *e.g.*,

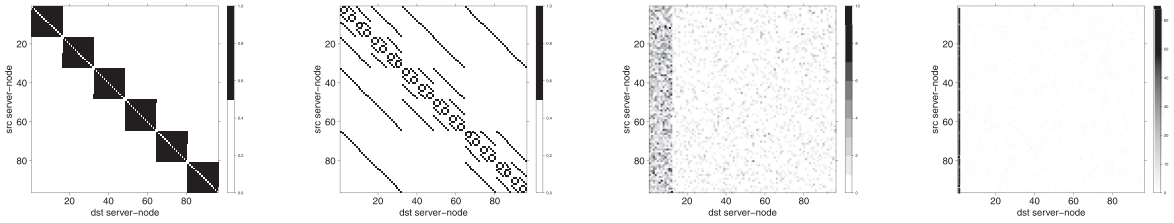

Fig. 5. Representative trace matrices for a 96-server-node DCN. From *left* to *right*: many all-to-all, butterfly, hot-region, and hot-spot. The many all-to-all traffic pattern is applied to a random permutation of the server-nodes, and is drawn unshuffled here only for clarity.

[9, Section 3.3.1]. This bottleneck throughout can then be immediately used to obtain the ABT (under the all-to-all traffic pattern) or the ideal throughput (under the random traffic pattern). This analytical methodology does not consider all of the practical characteristics of the design nor can it incorporate all the intricacies and subtleties of how traffic flows across a network. For this reason, it is common practice to bypass bisection width and determine throughput figures empirically, which is exactly what we do in our experimental work.

4.6. Software tools

Our software tool, Interconnection Networks Research Flow Evaluation Framework (INRFlow) [14] is specifically designed for testing large-scale, flow-based systems such as DCNs with tens or hundreds of thousands of nodes, which would prohibit the use of packet-level simulations. The results obtained from INRFlow inform a detailed evaluation within the intended scope of our paper.

INRFlow is capable of evaluating network topologies in two ways. Within INRFlow we can undertake a BFS for each server-node; this allows us to compute the hop-length of the *shortest path* between any two server-nodes and also to examine whether two server-nodes become disconnected in the presence of link failures. As we have noted in Section 4.2, results on shortest paths are of limited use when not studied in conjunction with a routing algorithm. Consequently, INRFlow also provides path and connectivity information about a given routing algorithm. We use the different routing algorithms within our DCNs as we have described so far in this section. The operation of the tool is as follows: for each flow in the workload, it computes the route using the required routing algorithm and updates link utilization accordingly. Then it reports a large number of statistics of interest, including the metrics discussed above.

Simulation. Simulation is the accepted methodology as regards the empirical investigation of DCNs. For example, as regards the DCNs FiConn, HCN, BCN, SWKautz, SWCube, and SWdBruijn, all empirical analysis is undertaken by simulation; on the other hand, DCell uses a test-bed of only 20 servers, BCube uses a test-bed of only 16 servers, and CamCube [1] uses a test-bed of only 27 servers. We argue that for the scenarios for which server-centric DCNs are intended, where the DCN will be expected to have thousands (if not millions) of servers (in future), experiments with a small test-bed cluster will not be too useful (except to establish proof-of-concept) and that simulation is the best way to proceed. Moreover, the uniformity and structured design of server-centric DCNs ameliorates against performance discrepancies that might arise in "more random" networks.

Error bars. The absence of error bars in our evaluation is by design. In our paper, random sampling occurs in two different ways: the first is where a random set of faulty links is chosen and properties of the faulty topology are plotted, as in Figs. 9–12; the second is with regards to randomised traffic patterns, as in Figs. 13–16. For each set of randomised link failures we plot statistics, either on connectivity or path length, for the all-to-all traffic pattern (*i.e.,* the whole population of server-node-pairs).

In Figs. 9–12 we sample the mean of two statistics over the set of all possible sets of m randomised link failures based on only one trial for each network and statistic, and therefore it does not make sense to compute estimated standard error for these plots. The true error clearly remains very small, however, because of the high level of uniformity of the DCNs we are studying, including the non-homogeneous DCN FiConn. The uniformity effectively simulates a large number of trials, since, for each choice of faulty link there are hundreds or thousands of other links in the DCN whose failure would have almost exactly the same effect on the overall experiment. Quantifying this error is outside the scope of our paper; however, it is evident from the low amount of noise in our plots that the true error is negligible in the context of the conclusions we are making. Figs. 13–16 sample flows to find the mean number of links with a certain proportion of utilisation, and to find the mean hop-lengths of the flows. Our sample sizes, given in Section 4.5, are exceedingly large for this purpose, and thus error bars would be all but invisible in these plots. We leave the explicit calculation to the reader.

5. Evaluation

In this section we perform an empirical evaluation of the DCN GQ⋆ and compare its performance with that of the DCNs FiConn and DPillar using the methodology and framework as detailed in Section 4. We begin by comparing various different versions of the three DCNs as regards ABT (as defined in Eq. (1)) and a coarse-grained analysis of latency[4]. Next, we focus on 4 comparable large-scale DCNs, namely $GQ^{\star}_{3,10}$, $GQ^{\star}_{4,6}$, $FiConn_{2,24}$, and $DPillar_{4,18}$, and we examine them in more detail with regard to fault-tolerance, latency, and load balancing, under different traffic patterns. Interspersed is an examination of the fault-tolerance capabilities of *GQ⋆-routing* in comparison with what might happen in the optimal scenario.

5.1. Aggregate bottleneck throughput

We begin by comparing GQ⋆, FiConn, and DPillar as regards aggregate bottleneck throughput, following our framework as outlined in Section 4.3; in particular, we use the routing algorithms *GQ⋆-routing, TOR*, and *DPillarSP*. We work with 3 different parameterized versions of GQ⋆, 2 of FiConn, and 3 of DPillar. Not only do we look at the relative ABT of different DCNs but we look at the scalability of each DCN in terms of ABT as the number of servers or component cost grows.

[4] The points plotted in Figs. 6–12 represent actual DCNs. Lines segments connect adjacent points, including some points outside the range of the plot, in order to reveal trends as the number of server-nodes changes.

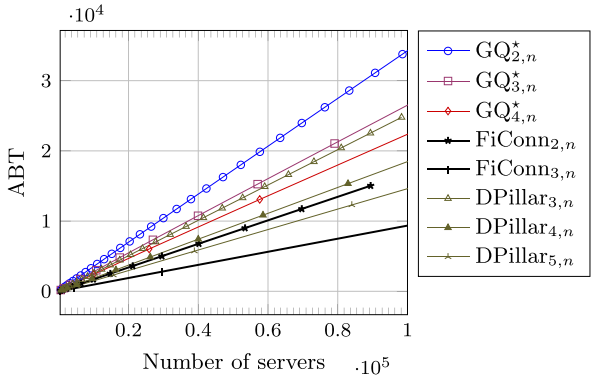

Fig. 6. ABT using *GQ*-routing*, *TOR*, and *DPillarSP*.

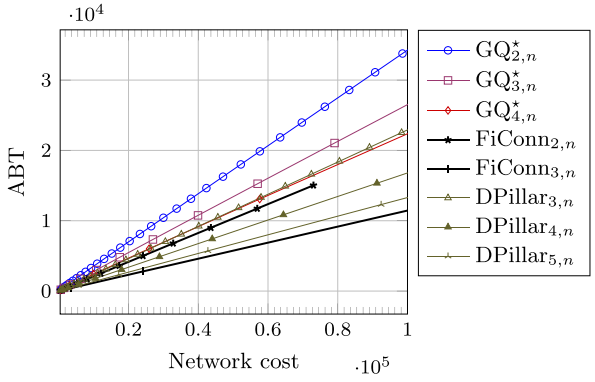

Fig. 7. ABT in terms of network cost for *GQ*-routing*, *TOR*, and *DPillarSP*, where a DCN DPillar is 110% the cost of a DCN GQ* with the same number of server nodes, whilst a DCN FiConn is 98% of the cost of a DCN GQ*. Network cost is normalised by the aggregated component cost per server in GQ*.

We first consider ABT vs. the number of servers in each network. Fig. 6 shows that ABT scales much better in GQ* than in FiConn. For the largest systems considered, GQ* supports up to around three times the ABT of $\text{FiConn}_{3,n}$. The difference between the 3 versions of GQ* and $\text{FiConn}_{2,n}$ is not as large but is still substantial. We can see that although the DCNs GQ* are constructed using far fewer switch-nodes and links than DPillar (when the two DCNs have the same number of server-nodes), their maximum sustainable ABT is broadly better; however, while the DCNs $\text{GQ}^\star_{k,n}$ with $k = 2$ and $k = 3$ consistently outperform all DPillar DCNs, $\text{DPillar}_{3,n}$ does slightly outperform $\text{GQ}^\star_{4,n}$.

Fig. 7 shows a plot of ABT vs. network cost under the most plausible assumptions discussed in Section 4.1, namely that the aggregated cost of components for DPillar is around 10% more and that of FiConn is around 2% less than that of GQ*. When we normalize by network cost, we can see a similar shape to Fig. 6 except that FiConn has a slightly improved scaling whereas DPillar has a slightly degraded one.

Let us focus on the increase in ABT for $\text{GQ}^\star_{k,n}$ as k decreases, which can be explained as follows. First, for a fixed number of server-nodes, reducing k results in an increased switch-node radix, which translates into higher locality. Second, reducing k results in lower routed mean hop-distance (see Fig. 8), which lowers the total utilization of the DCN and, when combined with good load balancing properties, yields a bottleneck link with fewer flows. As regards routed mean hop-distances for each of the DCNs, we can see that for each topology these increase very slowly with network size (apart from, perhaps, $\text{FiConn}_{3,n}$) and are, of course, bounded by the routed hop-diameter, which is dependent on k for all 3 topologies: $2k + 1$ for *GQ*-routing*; $2^{k+1} - 1$ for *TOR*; and $2k - 1$ for

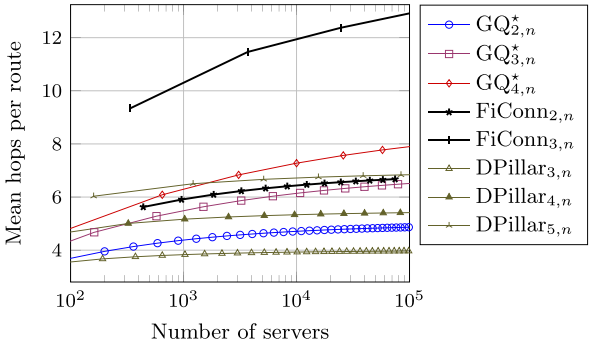

Fig. 8. Routed mean hop-distances for *GQ*-routing*, *TOR*, and *DPillarSP*.

DPillarSP. The "exponential nature" of FiConn discourages building this topology for any k larger than 2. However, note that in terms of routed mean hop-distance, DPillar is slightly better than GQ*, broadly speaking. However, such a metric cannot be taken in isolation and we take a closer look at this metric in relation to load balancing in a more detailed evaluation of our three DCNs in Section 5.4 (things are not what they might appear here).

Although we forgo a simulation of packet injections, our experiments do allow for a coarse-grained latency analysis. Network latency is brought on by routing packets over long paths and spending additional time processing (*e.g.*, buffering) the packets at intermediate nodes, due to network congestion. These scenarios have various causes, but they are generally affected by a DCN's ability to simultaneously balance network traffic and route it over short paths efficiently. Figs. 6–8 show that *GQ*-routing* scales well with respect to load balancing (high ABT) and routed mean hop-distance, from which we infer that in many situations $\text{GQ}^\star_{3,n}$ has lower latency than $\text{GQ}^\star_{4,n}$ and all FiConn DCNs, and likely performs at least similarly to $\text{DPillar}_{3,n}$.

In summary, GQ* has better ABT properties than FiConn and also broadly outperforms the denser DPillar; as discussed in Section 4.3, ABT is a performance metric of primary interest in the context of datacenters. We can also infer from our experiments a coarse-grained latency analysis, namely that *GQ*-routing* is likely to be at least as good as DPillar and better than FiConn.

5.2. Fault-tolerance

We now turn our attention to four concrete instances of the topologies and their routing algorithms: $\text{GQ}^\star_{3,10}$ and $\text{GQ}^\star_{4,6}$ with *GQ*-routing*; $\text{FiConn}_{2,24}$ with BFS; and $\text{DPillar}_{4,18}$ with *DPillarMP* (though we shall also consider DPillar with *DPillarSP* in the non-fault-tolerant environment of Section 5.4). As stated in Section 4.4, these DCNs were chosen as each has around 25,000 server-nodes and use switch-nodes with around 24 ports.

A priori, GQ* has a provably high number of parallel paths and server-parallel paths compared to FiConn and DPillar of similar size (see Table 2). Thus, if *GQ*-routing* utilises these paths, we expect strong performance in degraded networks. Fig. 9 shows the routed connectivity under failures[5] of *GQ*-routing* and *DPillarMP*. The plot indicates that *DPillarMP* underutilises the network, since the unrouted connectivity of DPillar (not plotted) is slightly stronger than that of GQ*. This highlights the fact that there is a close and complex relationship between topology, path-lengths, routing, fault-tolerance, and so on; ensuring that all aspects dovetail together is of primary importance. These observations also motivate a more detailed evaluation of *GQ*-routing* (and indeed fault-tolerant rout-

[5] Routed and unrouted data computed for other DCNs GQ* was very similar and is not plotted for the sake of clarity.

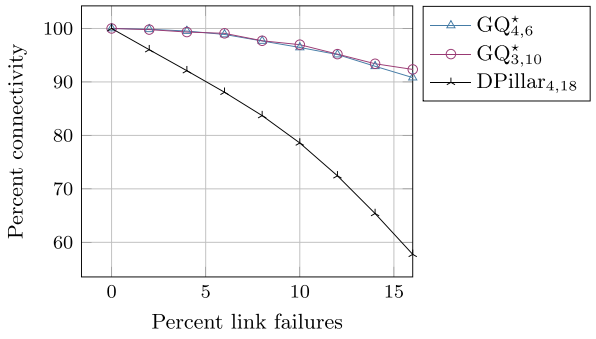

Fig. 9. Routed connectivity of *GQ*-routing* and *DPillarMP*.

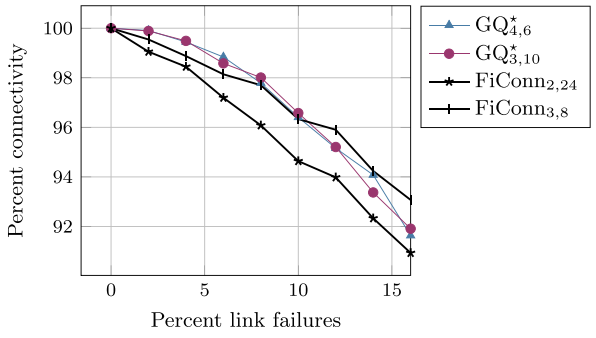

Fig. 10. Unrouted connectivity of GQ* and FiConn.

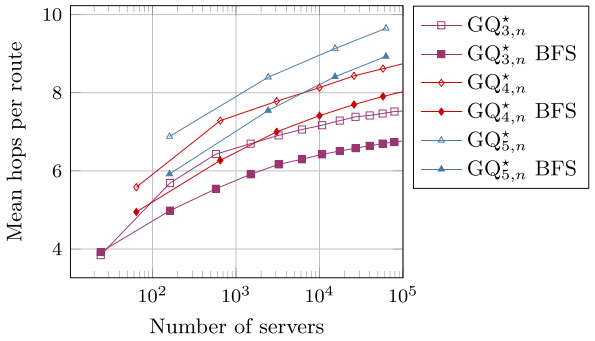

Fig. 11. Routed (*GQ*-routing*) and unrouted mean-distance in GQ* with 10% link failures.

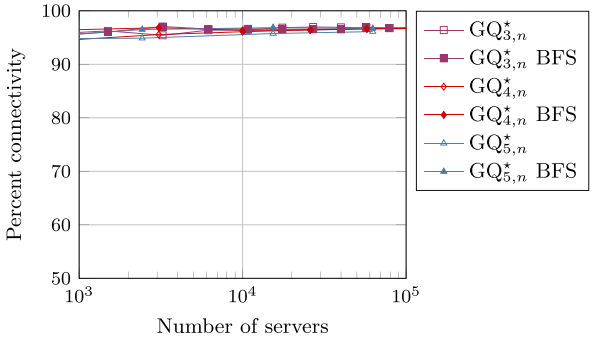

Fig. 12. Routed (*GQ*-routing*) and unrouted connectivity of GQ* with 10% link failures.

ing for DPillar). Note that the evaluation of *DPillarMP* in Liao et al. [31] is with respect to server-node faults, in which the performance of *DPillarMP* looks stronger than it does in our experiments with link-failures. This is because the failed server-nodes do not send messages and therefore do not factor into the connectivity of the faulty DPillar.

5.3. *Assessment of* GQ*-routing

With FiConn not having a fault-tolerant algorithm comparable to *GQ*-routing* (see Section 4.6), in Fig. 10 we plot the unrouted connectivity of GQ* with that of FiConn. As we can see, *GQ*-routing* performs similarly to FiConn in an optimal scenario. To our knowledge there is no fault-tolerant routing algorithm for FiConn that achieves anything close to the optimal performance of BFS (however, Fig. 12 shows that *GQ*-routing* very nearly achieves the optimum unrouted connectivity of GQ*).

In summary, we have shown that GQ* and *GQ*-routing* are very competitive when compared with both FiConn and DPillar in terms of fault-tolerance.

We assess the performance of *GQ*-routing* by comparing it with optimum performance, obtained by computing a BFS which finds a shortest-path (if it exists). Notice that since dimensional routing yields a shortest-path algorithm on $GQ_{k,n}$, it is straightforward to modify *GQ*-routing* so as to be a shortest path algorithm on $GQ^{\star}_{k,n}$; however, due to simplifications in our implementation there is a discrepancy of about 2% between shortest paths and *GQ*-routing* in a fault-free GQ*.

Of interest to us here is the relative performance of *GQ*-routing* and BFS in faulty networks. Fig. 11 plots the routed and unrouted mean hop-distances in networks with a 10% link failure rate; as can be seen, the difference between *GQ*-routing* and BFS in mean hop-distance is close to 10%. This is a reasonable overhead for a fault-tolerant routing algorithm, especially given the algorithm's high success rate at connecting pairs of servers in faulty networks: Fig. 12 plots the unrouted connectivity, which is optimum and achieved by a BFS, and the routed connectivity, achieved by *GQ*-routing*, for the same (10%) failure rate[6]. As it is currently implemented, *GQ*-routing* is optimised for maintaining connectivity at the cost of routing over longer paths if necessary. A different mix of features might reduce the 10% gap in Fig. 11 but increase the gap in Fig. 12. In any case, the performance of *GQ*-routing* is very close to the optimum.

5.4. *Detailed evaluation of large-scale DCNs*

We now return to our four concrete instances of the topologies and their basic routing algorithms: $GQ^{\star}_{3,10}$ and $GQ^{\star}_{4,6}$ with *GQ*-routing*; $FiConn_{2,24}$ with *TOR*; and $DPillar_{4,18}$ with *DPillarSP*. Our intention is to look at throughput, how loads are balanced, and the impact on latency.

Fig. 13 shows the number of flows in the bottleneck for the different traffic patterns considered in our study. We can see that these results follow those described above in that not only can GQ* broadly outperform FiConn and DPillar in terms of ABT, cost, latency, and fault-tolerance, but it does likewise in terms of throughput in that it can significantly reduce the number of flows in the bottleneck. The only exceptions are that $DPillar_{4,18}$ does best with the butterfly and hot-region traffic patterns: the rationale for the butterfly results is that the butterfly pattern matches perfectly the DPillar topology and, thus, it allows a very good balancing of the network, reducing the flows in the bottleneck; and although DPillar does best, the hot-region results for GQ* and DPillar are not very different. For the rest of the patterns, DPillar is clearly the worst performing in terms of bottleneck flow. Fig. 14 shows the routed mean hop-distance for the different patterns and topologies, and shows that DPillar, due to the higher number of switches,

[6] *GQ*-routing* appears to be better than BFS for certain numbers of servers, but this is because the faults were generated randomly for each test.

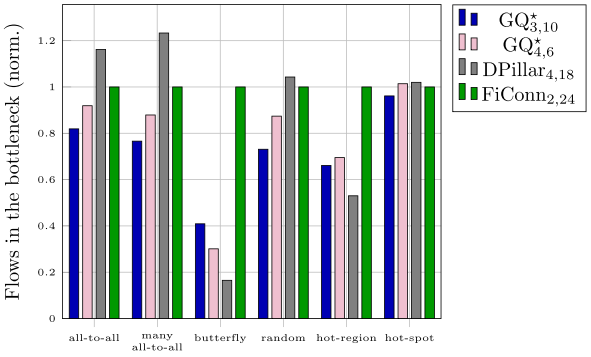

Fig. 13. Relative number of flows in the bottleneck for the different traffic patterns, normalised to FiConn and *TOR*.

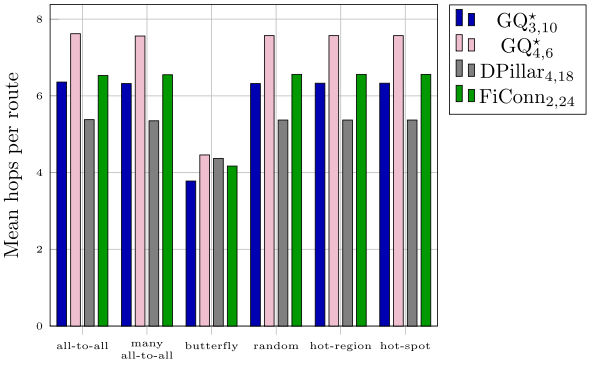

Fig. 14. Routed hop-distance for the different traffic patterns.

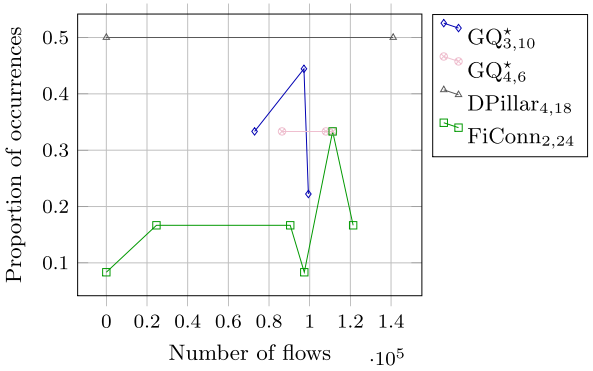

Fig. 15. Histogram of proportion of flows per link under the all-to-all traffic pattern. The mean number of flows per link are 89,567, 101,953, 84,615, and 141,187, for $GQ^{\star}_{3,10}$, $GQ^{\star}_{4,6}$, $FiConn_{2,24}$ and $DPillar_{4,18}$, respectively. Connecting lines are drawn for clarity.

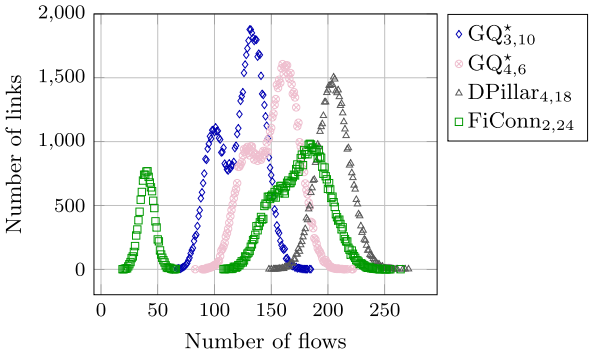

Fig. 16. Distribution of number of flows per link for the random traffic pattern. Not plotted are 52,488 unused links in DPillar and 6,162 unused links in FiConn.

can generally reach its destination using the shortest paths. Note that even with the clear advantage of having higher availability of shorter paths, $DPillar_{4,18}$ still has the highest number of flows in the bottleneck for the all-to-all, many all-to-all, random, and hot-spot traffic patterns, and, therefore, is the most prone to congestion. On the other hand $GQ^{\star}_{4,6}$, which uses the longest paths, has the second lowest number of flows in the bottleneck after $GQ^{\star}_{3,10}$. It should be noted that for the hot-region traffic pattern, the combination of a low number of bottleneck flows and short paths means that DPillar has the best performance.

The results we have obtained as regards bottleneck flows and routed hop-distances might appear surprising. However, a closer analysis helps to better appreciate the situation. Fig. 15 shows the distribution of flows across links in the all-to-all traffic pattern: for a given number of flows, we show the proportion of links carrying that number of flows. We can see that both GQ*s are much better balanced than both $FiConn_{2,24}$ and $DPillar_{4,18}$. For example, in $GQ^{\star}_{3,10}$ all of the links carry between 60,000 and 100,000 flows, and in $GQ^{\star}_{4,6}$ all of the links carry between 80,000 and 120,000 flows. However, nearly 25% of the links in $FiConn_{2,24}$ have less than 40,000 flows, whereas the other 75% of the links have between 80,000 and 140,000 flows. Even worse, in $DPillar_{4,18}$ half of the links have more than 100,000 flows while the other half are barely used. The imbalances present in $FiConn_{2,24}$ and $DPillar_{4,18}$ result in parts of the networks being significantly underutilised and other parts being overly congested.

A more detailed distribution obtained using the random traffic pattern is shown in Fig. 16. Here, we can see how both GQ*s are clearly better balanced than $FiConn_{2,24}$, as the latter has two pinnacles: one of low-load with about 30% of the links, and another of high-load with the rest of the links. We can also see that choosing the bottleneck link as the figure of merit is reasonable as it would yield similar results as if we had chosen the peaks in the plot.

Just as we did in Section 5.1, we can infer that $GQ^{\star}_{3,10}$ will provide better latency figures than $GQ^{\star}_{4,6}$ and $FiConn_{2,24}$ as it has fewer flows in the bottleneck link and uses shorter paths. The shorter paths in $DPillar_{4,18}$ do suggest that with low-intensity communication workloads it should have lower latency than $GQ^{\star}_{3,10}$, but since $DPillar_{4,18}$ is much poorer at balancing loads than $GQ^{\star}_{3,10}$, we can infer that it may have higher latency under higher-intensity communication workloads such as the ones typically used in datacenters.

6. Practical aspects of DCNs

Before we present our conclusions, let us take this opportunity to comment on the current place of the server-centric paradigm within the DCN landscape. There is still much to do as regards server-centric DCNs. As yet, insofar as we know, a substantial server-centric DCN has yet to be physically built. Given the costs of building a DCN, this will only happen when industry is convinced of the practical benefits that will accrue; and this will only happen when academic researchers have explored, proposed, and extensively evaluated many potential server-centric DCN topologies so as to provide a convincing argument. At present, we are nowhere near this point. If one reflects on the situation for general interconnection networks, one sees a plethora of different topologies together with a significant associated research effort (spanning engineering, mathematics, and computer science), with a substantial time lag before designs make it through to production.

There are many downstream challenges in order to move research on the server-centric paradigm forward from its current position, whereby conceptual and theoretical design ideas are subjected to an initial analysis as to their essential efficacy (indeed,

this constitutes the content of this paper). The more established switch-centric paradigm has made it through to production and is far more mature, with practical evaluations being undertaken upon real production datacenters. Consequently, there are a number of more pragmatic issues relating to switch-centric DCNs that have yet to be fully understood within the server-centric context, one of which is over-subscription (as we mentioned earlier). The key point as regards over-subscription is that the 'tree-like', hierarchical, layered nature of switch-centric DCNs, with the servers as 'leaves', makes over-subscription relatively straightforward to analyse: the focus is on the switches, and the combinatorial properties of the topologies (*e.g.*, fat-trees) makes the calculation of over-subscription ratios possible. The situation in server-centric DCNs is by no means as sharply defined: the switch is no longer the focus for loads, and the underlying topologies are far more complex. Furthermore, the combinatorial properties of these topologies (regarding bisection width, for example) are as yet generally not well understood. Added to this is the additional complication that in order to truly evaluate over-subscription, one needs the technical specifications of the intrinsic components, such as the actual bandwidths of links and servers. This data is readily available for switch-centric DCNs but not for server-centric DCNs (for obvious reasons).

It should also be noted that there are other approaches taken as regards (future) DCN design. An interesting account as regards the direction followed by Google can be found in [41], where it is noted (albeit briefly) that the server-centric paradigm has both pros and cons in comparison with their Clos topology-based approach: the pros are in relation to bandwidth; and the cons as regards the challenges and complexity posed by cabling, management, and routing. Let us briefly mention these latter issues in relation to the research in this paper. The cons of the server-centric paradigm are not so much definitive but are challenges remaining to be undertaken. Cabling is certainly an overlooked concern and there has only been limited mention of cabling in the context of server-centric DCNs; in fact, the only paper we are aware of is [2] and even then server-centric DCNs only feature obliquely. It is clear that cabling needs to be tackled within the context of server-centric DCNs. In this regard, the uniformity of our generic methodology provides an advantage as it enables us to potentially use cabling aspects related to the base graphs to assist us with cabling within the resulting stellar DCN. Issues relating to network management have, understandably, not been extensively studied due to the current 'theoretical' status of server-centric DCNs. As regards routing, as we have shown, our approach enables us to utilize established routing algorithms for the underlying base graphs in routing algorithms within our stellar DCNs, and we have demonstrated that such routing algorithms for GQ* compare favourably with those in DPillar and FiConn. Consequently, although much further research needs to be undertaken within the server-centric landscape, our stellar transformation has significant potential. What Singh et al. [41] alerts us to is that there needs to be a wider evaluation of server-centric DCNs across a range of metrics and that, ultimately, the server-centric paradigm will only succeed if such DCNs make it through to production and can compete in practical terms (at least as regards some performance aspects).

7. Conclusion

This paper proposes a new, generic construction that can be used to automatically convert existing interconnection networks, and their properties in relation to routing, path length, node-disjoint paths, and so on, into dual-port server-centric DCNs, that inherit the properties of the interconnection network. A range of interconnection networks has been identified to which our construction might be applied. A particular instantiation of our construction, the DCN GQ* where the base interconnection network is the generalized hypercube, has been empirically validated as regards network throughput, latency, load balancing capability, fault-tolerance, and cost to build. In particular, we have shown how GQ*, with its routing algorithm *GQ*-routing*, that is inherited from an existing routing algorithm for the generalized hypercube, consistently outperforms the established DCNs FiConn, with its routing algorithm *TOR*, and DPillar, with its routing algorithms *DPillarSP* and *DPillarMP*. As regards FiConn, the improved performance of GQ* was across all of the metrics we studied, apart from aggregated component cost where the two DCNs were approximately equal. As regards DPillar, the improved performance was across all metrics, apart from mean routed hop-distance and as regards bottleneck flows in the butterfly and hot-region traffic pattern. However, in mitigation against DPillar's improved mean routed hop-distance, our experiments as regards load balancing enable us to infer that although DPillar will exhibit lower latency in the case of low traffic congestion, when there is average to high traffic congestion DPillar's propensity to unbalanced loads on its links will mean that GQ* will have the better latency. Particularly marked improvements of GQ* against DPillar are as regards the fault-tolerant performance of the respective routing algorithms in link-degraded DCNs and also the aggregated component cost which in DPillar is around 10% higher than in GQ*. When we compare the performance of *GQ*-routing* within GQ* against what is optimally possible, in terms of path length, we find that *GQ*-routing* finds paths that are within 2% of the optimal length (0% is realistically possible) and within around 10% for degraded networks with 10% faulty links. This is a relatively small overhead for our routing algorithm which achieves very high connectivity, typically 95% connectivity when 10% of links are chosen to be faulty (uniformly at random).

There are a number of open questions immediately arising from this paper that we will investigate in the future. A non-comprehensive list is as follows.

- Analyse the practicalities (floor planning, wiring, availability of local routing, and so on) of packaging the DCNs GQ* and investigate a generic packaging methodology for DCNs formed using the stellar transformation (as we noted earlier, packaging issues relating to server-centric DCNs in general have hardly been considered).
- Apply the stellar transformation to other well-understood interconnection networks (some of which we have already highlighted) and undertake a more extensive empirical analysis. This empirical analysis should involve a wider range of DCN architectures and traffic models, and additional performance metrics, relating to, for example, multicast routing and energy efficiency.
- Explore the effect of the stellar construction on formal notions of symmetry in the base graph and in relation to metrics such as bisection width. We have actually made some initial progress on how the bisection widths of the base graph and the stellar DCNs are related, which will be reported elsewhere.
- Further investigate the routing algorithm *GQ*-routing*. For example, we should develop it so as to produce minimal paths for fault-free networks and compare the resulting performance with the near-optimal algorithm used in this paper.

Finally, let us remark upon two observations made by the anonymous reviewers that might lead to new research investigations, and not just in the context of stellar transformations. The first observation is that one could use, for example, 64-port switch-nodes in order to implement 4 16-port switch-nodes. Insofar as we are aware, this approach has not previously been systematically considered. It has a number of interesting mathematical ramifications; for example, as regards our stellar transformations, proceeding in this way would mean that the resulting stellar DCN inherited properties not from the base graph but from the

base graph where groups of nodes had been identified. A judicious choice of which nodes to identify could have a marked impact upon properties such as the diameter and connectivity. The second observation is that there has been no significant consideration of how one might use the hardware from an existing switch-centric DCN in order to build a server-centric DCN. One idea might be to have multiple, smaller server-centric DCNs connected by means of an aggregation/core infrastructure to form some flavour of hybrid between server- and switch-centric DCNs. For example, we might replace the lowest layer of a tree-like switch-centric DCN with small server-centric DCNs; thus, some ports from the switch-nodes of server-centric DCNs will be connected to the aggregation layer of the infrastructure. This might help alleviate cabling and large hop-count issues relating to server-centric DCNs.

Acknowledgements

This work has been funded by the Engineering and Physical Sciences Research Council (EPSRC) through grants EP/K015680/1 and EP/K015699/1. Dr. Javier Navaridas is also supported by the European Union's Horizon 2020 programme under grant agreement No. 671553 'ExaNeSt'. The authors gratefully acknowledge their support. The authors also gratefully acknowledge the incisive comments of anonymous reviewers which substantially improved the exposition in this paper.

References

[1] H. Abu-Libdeh, P. Costa, A. Rowstron, G. O'Shea, A. Donnelly, Symbiotic routing in future data centers, SIGCOMM Comput. Commun. Rev. 40 (4) (2010) 51–62.
[2] R. Agarwal, J. Mudigonda, P. Yalagandula, J.C. Mogul, An Algorithmic Approach to Datacenter Cabling, Technical Report, HP Laboratories, Palo Alto, USA, 2015.
[3] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, SIGCOMM Comput. Commun. Rev. 38 (4) (2008) 63–74.
[4] J. Arjona Aroca, A. Fernandez Anta, Bisection (band)width of product networks with application to data centers, IEEE Trans. Parallel. Distrib. Syst. 25 (3) (2014) 570–580.
[5] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, A view of cloud computing, Commun ACM 53 (4) (2010) 50–58.
[6] L.N. Bhuyan, D.P. Agrawal, Generalized hypercube and hyperbus structures for a computer network, IEEE Trans. Comput. C-33 (4) (1984) 323–333.
[7] J.-Y. Cai, G. Havas, B. Mans, A. Nerurkar, J.-P. Seifert, I. Shparlinski, On routing in circulant graphs, in: Proc. of 5th Int. Conf. on Computing and Combinatorics, Lecture Notes in Computer Science, vol. 1627, Springer, 1999, pp. 360–369.
[8] T. Chen, X. Gao, G. Chen, The features, hardware, and architectures of data center networks: a survey, J. Parallel Distrib. Comput. 96 (2016) 45–74.
[9] W.J. Dally, B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2003.
[10] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[11] G. Della Vecchia, C. Sanges, A recursively scalable network VLSI implementation, Future Gener. Comput. Syst. 4 (3) (1988) 235–243.
[12] R. Diestel, Graph Theory, Graduate Texts in Mathematics, vol. 173, Springer, 2012.
[13] A. Erickson, A. Kiasari, J. Navaridas, I.A. Stewart, An efficient shortest-path routing algorithm in the data centre network DPillar, in: Proc. of 9th Int. Conf. on Combinatorial Optimization and Applications, Lecture Notes in Computer Science, vol. 9486, Springer, 2015, pp. 209–220.
[14] A. Erickson, A. Kiasari, J. Pascual Saiz, J. Navaridas, I.A. Stewart, Interconnection Networks Research Flow Evaluation Framework (INRFlow), 2016. [Software] https://bitbucket.org/alejandroerickson/inrflow.
[15] A.-H. Esfahanian, S.L. Hakimi, Fault-tolerant routing in de bruijn communication networks, IEEE Trans. Comput. 34 (9) (1985) 777–788.
[16] N. Farrington, G. Porter, S. Radhakrishnan, H.H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, A. Vahdat, Helios: a hybrid electrical/optical switch architecture for modular data centers, SIGCOMM Comput. Commun. Rev. 40 (4) (2010) 339–350.
[17] A.W.-C. Fu, S.-C. Chau, Cyclic-cubes: a new family of interconnection networks of even fixed-degrees, IEEE Trans. Parallel. Distrib. Syst. 9 (12) (1998) 1253–1268.
[18] A. Greenberg, J.R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D.A. Maltz, P. Patel, S. Sengupta, VL2: a scalable and flexible data center network, SIGCOMM Comput. Commun. Rev. 39 (4) (2009) 51–62.
[19] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, S. Lu, BCube: a high performance, server-centric network architecture for modular data centers, SIGCOMM Comput. Commun. Rev. 39 (4) (2009) 63–74.
[20] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, S. Lu, DCell: a scalable and fault-tolerant network structure for data centers, SIGCOMM Comput. Commun. Rev. 38 (4) (2008) 75–86.
[21] D. Guo, T. Chen, D. Li, M. Li, Y. Liu, G. Chen, Expandable and cost-effective network structures for data centers using dual-port servers, IEEE Trans. Comput. 62 (7) (2013) 1303–1317.
[22] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S.R. Das, J.P. Longtin, H. Shah, A. Tanwer, Firefly: a reconfigurable wireless data center fabric using free-space optics, SIGCOMM Comput. Commun. Rev. 44 (4) (2014) 319–330.
[23] A. Hammadi, L. Mhamdi, A survey on architectures and energy efficiency in data center networks, Comput. Commun. 40 (2014) 1–21.
[24] L.-H. Hsu, C.-K. Lin, Graph Theory and Interconnection Networks, CRC Press, 2009.
[25] F.K. Hwang, A survey on multi-loop networks, Theor. Comput. Sci. 299 (1–3) (2003) 107–121.
[26] F.T. Leighton, Introduction to Parallel Algorithms and Architectures: Array, Trees, Hypercubes, Morgan Kaufmann, 1992.
[27] C.E. Leiserson, Fat-trees: universal networks for hardware-efficient supercomputing, IEEE Trans. Comput. 34 (10) (1985) 892–901.
[28] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, S. Lu, J. Wu, Scalable and cost-effective interconnection of data-center servers using dual server ports, IEEE/ACM Trans. Netw. 19 (1) (2011) 102–114.
[29] D. Li, J. Wu, On data center network architectures for interconnecting dual-port servers, IEEE Trans. Comput. 64 (11) (2015) 3210–3222.
[30] Z. Li, Z. Guo, Y. Yang, BCCC: an expandable network for data centers, in: Proc. of 10th ACM/IEEE Symp. on Architectures for Networking and Communications Systems, ACM, 2014, pp. 77–88.
[31] Y. Liao, J. Yin, D. Yin, L. Gao, DPillar: dual-port server interconnection network for large scale data centers, Comput. Netw. 56 (8) (2012) 2132–2147.
[32] Y. Liu, J.K. Muppala, M. Veeraraghavan, D. Lin, M. Hamdi, Data Center Networks: Topologies, Architectures and Fault-Tolerance Characteristics, Springer, 2013.
[33] Y.J. Liu, P.X. Gao, B. Wong, S. Keshav, Quartz: a new design element for low-latency DCNs, SIGCOMM Comput. Commun. Rev. 44 (4) (2014) 283–294.
[34] E.A. Monakhova, A survey on undirected circulant graphs, Discrete Math. Algorithms Appl. 4 (1) (2012) 1250002. (30 pages).
[35] J. Navaridas, J. Miguel-Alonso, F.J. Ridruejo, On synthesizing workloads emulating MPI applications, in: Proc. of IEEE Int. Symp. on Parallel and Distributed Processing, IEEE, 2008, pp. 1–8.
[36] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, A. Vahdat, Portland: a scalable fault-tolerant layer-2 data center network fabric, SIGCOMM Comput. Commun. Rev. 39 (4) (2009) 39–50.
[37] L. Popa, S. Ratnasamy, G. Iannaccone, A. Krishnamurthy, I. Stoica, A cost comparison of datacenter network architectures, in: Proc. of 6th Int. Conf. on Emerging Networking Experiments and Technologies, ACM, 2010. article no. 16
[38] D.K. Pradhan, S.M. Reddy, A fault-tolerant communication architecture for distributed systems, IEEE Trans. Comput. C-31 (9) (1982) 863–870.
[39] G. Qu, Z. Fang, J. Zhang, S.-Q. Zheng, Switch-centric data center network structures based on hypergraphs and combinatorial block designs, IEEE Trans. Parallel Distrib. Syst. 26 (4) (2015) 1154–1164.
[40] J. Shuja, K. Bilal, S.A. Madani, M. Othman, R. Ranjan, P. Balaji, S.U. Khan, Survey of techniques and architectures for designing energy-efficient data centers, IEEE Syst. J. 10 (2) (2016) 507–519.
[41] 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. HÈµlzle, S. Stuart, A. Vahdat, Jupiter rising: a decade of clos topologies and centralized control in google's datacenter network, SIGCOMM Comput. Commun. Rev. 45 (5) (2015) 183–197.
[42] A. Singla, C.-Y. Hong, L. Popa, P.B. Godfrey, Jellyfish: networking data centers randomly, in: Proc. of 9th USENIX Symp. on Networked Systems Design and Implementation, USENIX Association, 2012.
[43] I.A. Stewart, Improved routing in the data centre networks HCN and BCN, in: Proc. of 2nd Int. Symp. on Computing and Networking, IEEE, 2014, pp. 212–218.
[44] A. Touzene, K. Day, B. Monien, Edge-disjoint spanning trees for the generalized butterfly networks and their applications, J. Parallel Distrib. Comput. 65 (11) (2005) 1384–1396.
[45] T. White, Hadoop: The Definitive Guide, O'Reilly Media, 2009.
[46] C. Wu, R. Buyya, Cloud Datacenters and Cost Modeling, Elsevier, 2015.
[47] J. Xu, Topological Structure and Analysis of Interconnection Networks, Springer, 2010.
[48] S. Young, S. Yalamanchili, Adaptive routing in generalized hypercube architectures, in: Proc. of 3rd IEEE Symp. on Parallel and Distributed Processing, IEEE, 1991, pp. 564–571.

Alejandro Erickson completed a 3-year postdoctoral research position at Durham University, United Kingdom in 2016, where he did research on various topological aspects of interconnection networks, with an emphasis on applications in datacenter networks. He received his Ph.D. in Computer Science from the University of Victoria, Canada in 2013 and his M.Math in Combinatorics and Optimization from the University of Waterloo, Canada in 2008. Dr. Erickson has published in a broad range of topics, including data centre networks, computational geometry, graph and matroid theory, enumerative combinatorics, education, and mathematical art.

Iain A. Stewart received the MA and PhD degrees in mathematics from the University of Oxford, United Kingdom in 1983 and the University of London, United Kingdom in 1986. He is a professor in the School of Engineering and Computing Sciences, Durham University, United Kingdom. His research interests include interconnection networks for parallel and distributed computing, computational complexity and finite model theory, algorithmic and structural graph theory, theoretical aspects of artificial intelligence, GPGPU computing and computational aspects of group theory.

Dr. Javier Navaridas is a lecturer in computer architecture in the Advanced Processors Technologies group of the University of Manchester. He obtained his PhD in computer engineering in 2009 from the University of the Basque Country which was rewarded with an Extraordinary Doctorate Award (Top 5% of theses in the academic year – 16 out of 306). During that period he held a pre-doctoral and a post-doctoral grant with the Intelligent Systems Group led by Prof. J.A. Lozano. He joined the University of Manchester with a prestigious Newton Fellowship in 2010. His research interests include interconnection networks for parallel and distributed systems, and performance evaluation of parallel architectures, with special emphasis on simulation and characterization of application's behaviour.

Abbas E. Kiasari is a research associate with the Advanced Processor Technologies group at the University of Manchester, UK, where he does research on various aspects of interconnection networks. He got his Ph.D. in electronic and computer systems from KTH, Sweden, in 2013, his M.Sc. in computer engineering from Sharif University of Technology, Iran, in 2005 and his B.Sc. in electrical engineering from Ferdowsi University of Mashhad, Iran, in 2003.