Future Generation Computer Systems 75 (2017) 58–71

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Improved routing algorithms in the dual-port datacenter networks HCN and BCN

Alejandro Erickson [a], Iain A. Stewart [a,*], Jose A. Pascual [b], Javier Navaridas [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

HIGHLIGHTS

- Improved routing algorithms for the datacenter networks HCN and BCN are proposed.
- New routing algorithms are derived from algorithms for WK-recursive networks.
- Routing algorithms are simulated for a variety of traffic patterns and workloads.
- Our routing algorithms massively improve on existing ones.

ARTICLE INFO

Article history:
Received 9 January 2017
Received in revised form
4 April 2017
Accepted 5 May 2017
Available online 11 May 2017

Keywords:
Datacenters
Datacenter networks
HCN
BCN
One-to-one routing
WK-recursive networks
Performance metrics

ABSTRACT

We present significantly improved one-to-one routing algorithms in the datacenter networks HCN and BCN in that our routing algorithms result in much shorter paths when compared with existing routing algorithms. We also present a much tighter analysis of HCN and BCN by observing that there is a very close relationship between the datacenter networks HCN and the interconnection networks known as WK-recursive networks. We use existing results concerning WK-recursive networks to prove the optimality of our new routing algorithm for HCN and also to significantly aid the implementation of our routing algorithms in both HCN and BCN. Furthermore, we empirically evaluate our new routing algorithms for BCN, against existing ones, across a range of metrics relating to path-length, throughput, and latency for the traffic patterns all-to-one, bisection, butterfly, hot-region, many-all-to-all, and uniform-random, and we also study the completion times of workloads relating to MapReduce, stencil and sweep, and unstructured applications. Not only do our results significantly improve routing in our datacenter networks for all of the different scenarios considered but they also emphasize that existing theoretical research can impact upon modern computational platforms.

1. Introduction

Datacenters are becoming pervasive within the global computational infrastructure and the sizes of these datacenters are expanding rapidly, with some of the largest operators managing over a million servers across multiple datacenters. As to how these servers are interconnected via the *datacenter network* (*DCN*) is a fundamental issue, the consideration of which involves a mix of mathematics, computer science, and engineering. Moreover, just as with the design of interconnection networks for distributed-memory multiprocessors or networks-on-chips, there is no 'silver bullet' solution, for there is a wide range of design parameters to consider, some of which are conflicting.

The traditional architecture of a DCN is 'switch-centric' whereby the primary structure is a topology (usually tree-based) of switches with the switches possessing interconnection intelligence. The DCNs Fat-Tree [1], VL2 [2], and Portland [3] are typical of such DCNs. A more recent and alternative architecture is 'server-centric' whereby the interconnection intelligence resides within the servers and the switches are dumb crossbars (so, there are no switch-to-switch links). The DCNs DCell [4], FiConn [5], BCube [6], MDCube [7], HCN and BCN [8], and GQ* [9] are typical of server-centric DCNs.

The server-centric architecture possesses a number of advantages when compared with the more traditional switch-centric architecture: tree-based switch-centric DCNs tend to be such

* Corresponding author.
E-mail addresses: alejandro.erickson@gmail.com (A. Erickson), i.a.stewart@durham.ac.uk (I.A. Stewart), jose.pascual@manchester.ac.uk (J.A. Pascual), javier.navaridas@manchester.ac.uk (J. Navaridas).

http://dx.doi.org/10.1016/j.future.2017.05.004

that 'root' switches quickly become a bottleneck; the underlying topologies of server-centric topologies are better suited to support traffic patterns prevalent in datacenters (such as one-to-all and all-to-all); the switches in server-centric DCNs can be chosen to be commodity switches as they require no intelligence; and multiple network interface controller (NIC) ports on servers in server-centric DCNs can be utilized so that more varied topologies can be constructed (see, for example, [10,8,11] for more information).

Whilst multiple NIC ports can be used when building server-centric DCNs, commodity servers usually only have a small number of NIC ports, often only two. This can be problematic as a primary aim of DCN design is to incorporate a large number of servers within the datacenter. For example, when one builds the DCNs DCell, BCube, and MDCube, one finds that the number of NIC ports required increases as the number of servers rises. On the other hand, FiConn and GQ*, for example, is such that no matter how many servers there are, each server needs only two NIC ports; such server-centric DCNs are referred to as *dual-port*.

Motivated by the need to limit the number of NIC ports on servers (so that commodity servers might be used), Guo et al. introduced and evaluated the dual-port DCNs HCN and BCN [8]. The general construction is that the DCN HCN is a recursively-defined family of networks, with the DCN BCN built using (copies of) the DCN HCN by including an additional layer of interconnecting links. After defining the DCNs HCN and BCN, Guo et al. developed a number of routing algorithms (including one-to-one, multipath, and fault-tolerant algorithms) and evaluated HCN and BCN, primarily in comparison with FiConn and according to a number of basic metrics.

We pursue the analysis of the DCNs HCN and BCN in this paper. In particular, we present significantly improved one-to-one routing algorithms in both HCN and BCN, in that our routing algorithms result in much shorter paths than those in [8] (our analysis is both theoretical and empirical). We also present a much tighter analysis of HCN and BCN by observing that there is close relationship between the DCN HCN and the interconnection networks known as *WK-recursive networks*, which originated in [12] and which have been well studied as general interconnection networks. We use existing theoretical results concerning WK-recursive networks to develop our routing algorithms and prove the optimality of our new routing algorithm for HCN (in terms of path length), as well as to significantly aid the implementation of our routing algorithms in both HCN and BCN. Not only do we develop routing algorithms for HCN and BCN that are theoretical improvements over existing routing algorithms but we undertake an extensive empirical evaluation of our algorithms, against existing ones, for DCNs of a range of realistic sizes, under a range of traffic patterns and workloads, and across a range of metrics. In particular, we consider metrics relating to hop-length, throughput, and latency for the ('static') traffic patterns all-to-one, bisection, butterfly, hot-region, many-all-to-all, and uniform-random, and we also study the completion times of ('dynamic') workloads relating to MapReduce, stencil and sweep, and unstructured applications, where these workloads have data associated with flows and might involve some causality between flows. We also study how the connection rule used to build BCN out of copies of HCN, of which there are currently two in the literature (though potentially many more), impacts upon the resulting DCN BCN, in terms of the above empirical analysis. Our simulations are undertaken with our own purpose-built flow-based simulator `INRFlow` [13]. A novel aspect of our simulations is that whereas the 'static' simulation of routing algorithms on the above traffic patterns is the norm within the server-centric research community, `INRFlow` allows us to simulate our routing algorithms on the above 'dynamic' workloads (insofar as we are aware, this paper contains the first such 'dynamic' simulations on server-centric DCNs).

Our results are extremely encouraging, for we almost universally obtain improvements. Not only do we obtain theoretically-improved algorithms but our empirical analysis suggests that there are significant gains to be made by the practical deployment of our new routing algorithms in HCN and BCN. For example, when compared with the routing algorithm *BdimRouting* for BCN (from [8]), our primary new routing algorithm for BCN, namely $NewBdimRouting_\gamma$, achieves hop-length savings for all DCNs studied and across all traffic patterns, averaging at around a 25% improvement. What is more, a practical version of $NewBdimRouting_\gamma$, namely $NewBdimRouting_1$, where we curtail the inherent search for shorter paths within $NewBdimRouting_\gamma$, is shown to give a performance comparable with that of $NewBdimRouting_\gamma$. Our algorithm $NewBdimRouting_1$ also achieves a significant improvement in both throughput and latency when compared with *BdimRouting* in the different scenarios: as regards throughput, on average this improvement is by 36% and 55% for the two throughput metrics we consider; and as regards latency, on average this improvement is by 10%. Our algorithm $NewBdimRouting_1$ also obtains improvements for all the different 'dynamic' workloads mentioned above.

This paper is structured as follows. In the next section, after detailing the essential concepts of server-centric DCNs, we give precise definitions of the DCN HCN, and exhibit the link with WK-recursive networks, and the DCN BCN. In Section 3, we develop new one-to-one routing algorithms for HCN, prove their optimality, and explain how they can be very easily implemented. Our new one-to-one routing algorithms for BCN are developed in Section 4. In Section 5, we explain the framework for and reasoning behind our experiments, and in Section 6, we supply and evaluate the results we obtain. Our conclusions and directions for further research are presented in Section 7. A preliminary version of this paper where the analysis only considered HCN appeared as [14].

2. Server-centric datacenter networks

In this section we define the graph-theoretic abstractions that we use to obtain our results on server-centric DCNs. A server-centric DCN is built from commodity off-the-shelf (COTS) switches and servers, interconnected by cable links. It is distinguished from other types of datacenters in that very low capability is required of the switches, which act as simple, non-blocking crossbars, and any routing algorithms and network protocols are implemented within the servers. Thus, we abstract a server-centric DCN as a graph $G = (S \cup W, E)$, where $u \in S$ is a *server-node*, representing a server, and $w \in W$ is a *switch-node*, representing a switch, and each link in E represents a physical link of the DCN. The only requirement, imposed by the simplicity of the switches we are modelling, is that no two switch-nodes are connected by a link; as such, $E \cap (W \times W) = \varnothing$. As we shall see, our DCNs come in parameterized families. Henceforth, we use the term DCN to refer to both a family member and the family itself.

A routing algorithm[1] takes a pair of server-nodes, (src, dst), as input and outputs a path, P, in G from src to dst. The *path-length* of P is equal to the number of links P contains, and the *hop-length* of P is equal to the number of hops it contains, where a *hop* is a link joining two server-nodes or a path of path-length 2 from a server-node to another server-node through a switch-node. Hop-length is the primary distance-related performance metric used in evaluations of server-centric DCNs (see, *e.g.*, HCN and BCN [8], DCell [4], FiConn [5], BCube [6], MDCube [7], and GQ* [9]), for the reason that packets must travel up and down the protocol stack of each intermediate server to reach the service that will route them to the next server, rendering negligible the time spent at each switch. We work with hop-length in this paper.

[1] Strictly speaking, this is a unicast routing algorithm, but we do not discuss any other sort in this paper.