Computer Communications xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Computer Communications

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

Towards cost-effective and low latency data center network architecture

Q1 Ting Wang[a,1,*], Zhiyang Su[b,2], Yu Xia[c,3], Bo Qin[b,4], Mounir Hamdi[d,5]

Q2 [a] Hong Kong University of Science and Technology, Hong Kong
[b] Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong
[c] College of Computer Science, Sichuan Normal University, China
Q3 [d] College of Science, Engineering and Technology in Hamad bin Khalifa University, Qatar

A R T I C L E I N F O

Article history:
Received 17 September 2015
Revised 25 February 2016
Accepted 27 February 2016
Available online xxx

Keywords:
Data center networks
Architecture
Torus topology
Probabilistic weighted routing
Deadlock-free

A B S T R A C T

This paper presents the design, analysis, and implementation of a novel data center network architecture, named *NovaCube*. Based on regular Torus topology, *NovaCube* is constructed by adding a number of most beneficial jump-over links, which offers many distinct advantages and practical benefits. Moreover, in order to enable *NovaCube* to achieve its maximum theoretical performance, a probabilistic oblivious routing algorithm PORA is carefully designed. PORA is a both deadlock and livelock free routing algorithm, which achieves near-optimal performance in terms of average routing path length with better load balancing thus leading to higher throughput. Theoretical derivation and mathematical analysis together with extensive simulations further prove the good performance of *NovaCube* and PORA.

1. Introduction

The data center network (DCN)[6] architecture is regarded as one of the most important determinants of network performance in data centers, and plays a significant role in meeting the requirements of could-based services as well as the agility and dynamic reconfigurability of the infrastructure for changing application demands. As a result, many novel proposals, such as Fat-Tree [1], VL2 [2], DCell [3], BCube [4], c-Through [5], Helios [6], SprintNet [7,8], CamCube [9], Small-World [10], NovaCube [11], CLOT [12], and so on, have been proposed aiming to efficiently interconnect the servers inside a data center to deliver peak performance to users.

Generally, DCN topologies can be classified into four categories: multi-rooted tree-based topology (e.g. Fat-Tree), server-centric topology (e.g. DCell, BCube, SprintNet), hybrid network (e.g. c-Through, Helios) and direct network (e.g. CamCube, Small-World) [13]. Each of these has their advantages and disadvantages. Tree-based topologies, like FatTree and Clos, can provide full bisection bandwidth, thus the any-to-any performance is good. However, their building cost and complexity is relatively high. The recursive-defined server-centric topology usually concentrates on the scalability and incremental extensibility with a lower building cost; however, the full bisection bandwidth may not be achieved and their performance guarantee is only limited to a small scope. The hybrid network is a hybrid packet and circuit switched network architecture. Compared with packet switching, optical circuit switching can provide higher bandwidth and lower latency in transmission with lower energy consumption. However, optical circuit switching cannot achieve full bisection bandwidth at packet granularity. Furthermore, the optics also suffers from slow switching speed which can take as high as tens of milliseconds.

The direct network topology, which directly connects servers to other servers, is a switchless network interconnection without any switches, or routers. It is usually constructed in a regular pattern, such as Torus (as show in Fig. 1). Besides being widely used in high-performance computing systems, Torus is also an attractive network architecture candidate for data centers. However, this design suffers consistently from poor routing efficiency compared with other designs due to its relatively long network diameter (the maximum shortest path between any node pairs), which is known

Q4 * Corresponding author. Tel.: +86 13681755836.
E-mail addresses: twangah@connect.ust.hk (T. Wang), zsuab@cse.ust.hk (Z. Su), rainsia@163.com (Y. Xia), bqin@cse.ust.hk (B. Qin), hamdi@cse.ust.hk (M. Hamdi).
[1] Student Member, IEEE
[2] Student Member, IEEE
[3] Member, IEEE
[4] Student Member, IEEE
[5] Fellow, IEEE
[6] The term "data center network" and "DCN" are used interchangeably in this paper.

http://dx.doi.org/10.1016/j.comcom.2016.02.016

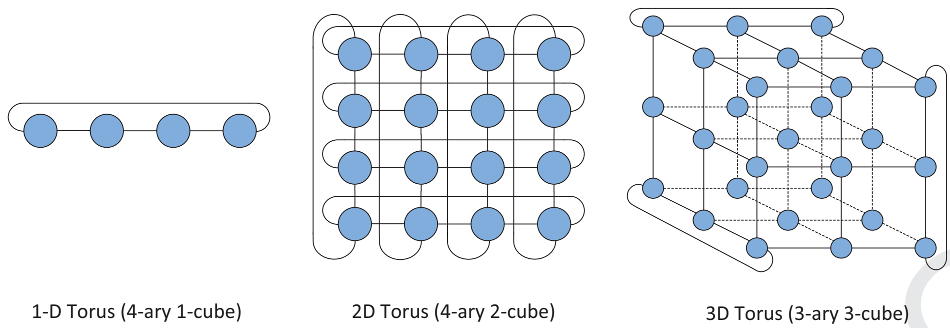

Fig. 1. Examples of 1D, 2D, 3D Torus topologies.

as $\lfloor \frac{k}{2} \rfloor n$ for a n-D Torus[7] with radix k. Besides, a long network diameter may also lead to high communication delay. Furthermore, its performance largely depends on the routing algorithms.

In order to deal with these imperfections, in this paper we propose a novel container level high-performance Torus-based DCN architecture named *NovaCube*. The key design principle of *NovaCube* is to connect the farthest node pairs by adding additional jump-over links. In this way, *NovaCube* can halve the network diameter and receive higher bisection bandwidth and throughput. Moreover, we design a new weighted probabilistic oblivious deadlock-free routing algorithm PORA for *NovaCube*, which achieves low average routing path length and good load-balancing by exploiting the path diversities.

The primary contributions of this paper can be summarized as follows:

(1) We propose a novel Torus-based DCN architecture *NovaCube*, which exhibits good performance in network latency, bisection bandwidth, throughput, path diversity and average path length.
(2) We carefully design a weighted probabilistic oblivious routing algorithm PORA, which is both deadlock-free and livelock-free, and helps *NovaCube* achieve good load balancing.
(3) We introduce a credit-based lossless flow control mechanism in *NovaCube* network.
(4) We design a practical geographical address assignment mechanism, which also can be applied to the traditional Torus network.
(5) We implement *NovaCube* architecture, PORA routing algorithm and the flow control mechanism in NS3. Extensive simulations are conducted to demonstrate the good performance of *NovaCube*.

The rest of the paper is organized as follows. First, we briefly review the related research literature in Section 2. Then Section 3 demonstrates the motivation. In Section 4, *NovaCube* architecture is introduced and analyzed in detail. Afterwards, the routing algorithm PORA is designed in Section 5. Section 6 introduces the credit-based flow control mechanism. Section 7 demonstrates a geographical address assignment mechanism. Section 8 presents the system evaluation and simulation results. Finally, Section 9 concludes this paper.

2. Related work

2.1. Network interconnection

The Torus-based topology well implements the network *locality* forming the servers in close proximity of each other, which increases the communication efficiency. Besides being widely used in supercomputing, Torus network has also been introduced to the data center networks. Three typical representatives are namely CamCube [9], Small-World [10] and CLOT [12].

CamCube was proposed by Abu-Libdeh et al., and the servers in CamCube are interconnected in a 3D Torus topology. The CamCube is designed target to shipping container-sized data centers, and is a server-only switchless network design. With the benefit of Torus architecture and the flexibility offered by a low-level link orientated CamCube API, CamCube allows applications to implement their own routing protocols so as to achieve better application-level performance. However, as aforementioned this design based on the regular Torus suffers long average routing path – $O(N^{1/3})$ hops, with N servers, which results in poor routing efficiency.

In order to overcome this limitation, Shin Ji-Yong, et al. proposed Small-World, which provides an unorthodox random data center network topology. It is constructed based on some regular topologies (such as ring, Torus or cube) with the addition of a large number of random links which can help reduce the network diameter and achieve higher routing efficiency. The degree of each node in Small-World is limited to six, taking realistic deployment and low cost into consideration. In addition to traditional routing methods, Small-World also provides content routing coupled with geographical address assignment, which in turn efficiently implements key-value stores. However, its shortest path routing suffers poor worst-case throughput and poor load balancing.

CLOT was also a DCN architecture designed based on Torus topology. CLOT shares the same goal with Small-World, which aims to reduce the routing path length and improve its overall network performance while retaining the Torus merits. Based on regular Torus topology, CLOT uses a number of most beneficial small low-end switches to connect each node and its most distant nodes in different dimensions. In this way, for a n-D CLOT, each switch will connect to 2^n nodes. By employing additional low-end switches, CLOT largely shortens the network diameter and the average path length. However, it also induces an extra expenditure on these switches.

2.2. Power savings in data centers

The energy cost of a data center accounts for a large portion of total budget [14–17]. There have emerged a considerable number of research and investigation to achieve a green data center. Generally, the existing proposals can be classified into four categories as below.

(1) *Network level*: This scheme usually resorts to energy-aware routing, VM migration/placement, flow scheduling and workload management mechanism to consolidate traffic and turn off idle switches/servers [17–20].
(2) *Hardware level*: This scheme aims to design energy-efficient hardware (e.g. server, switch) by using certain

[7] n-D Torus with radix k is also called k-ary n-cube, which may be used interchangeably in this paper.

Please cite this article as: T. Wang et al., Towards cost-effective and low latency data center network architecture, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.02.016

energy-efficient techniques like DVFS, VOVO, PCPG,and so on [21–24].

(3) *Architectural level*: This scheme designs energy-efficient network architecture to achieve power savings, examples like flattened butterfly topology [25], Torus-based topology [9–11] and content-centric networking (CCN) based architectures which can reduce the content distribution energy costs [26,27].

(4) *Green energy resources*: This scheme makes use of green sources to reduce the energy budget such as wind, water, solar energy, heat pumps, and so on [28,29].

NovaCube can be considered as an architectural level approach, which avoids using energy hungry switches. Moreover, NovaCube would also save the cooling cost induced by cooling the heat generated by switches. More discussions about the energy efficiency performance of *NovaCube* are given in Section 4.2.7.

3. Motivation

3.1. Why Torus-based clusters

The Torus (or precisely k-ary n-cube) based intserconnection has been regarded as an attractive DCN architecture scheme for data centers because of its own unique advantages, some of which are listed below.

Firstly, it incurs lower infrastructure cost since it is a switchless architecture without needing any expensive switches. In addition, the power consumed by the switches and its associated cooling power can also be saved.

Secondly, it achieves better fault-tolerance. Traditional architecture is usually constructed with a large number of switches, any failure of which could greatly impact on the network performance and system reliability. For example, if a ToR switch fails, the whole rack of servers will lose connection with the servers in other racks. Comparatively, the rich interconnectivity and in-degree of Torus-based switchless architecture makes the network far less likely to be partitioned. The path diversity can also provide good load balance even on permutation traffic.

Thirdly, the architectural symmetry of Torus topology optimizes the scalability and granularity of Clusters. It allows systems to economically scale to tens of thousands of servers, which is well beyond the capacity of Fat-Tree switches. For an n-ary k-cube, the network can support up to k^n nodes, and scales at a high exponential speed which outperforms traditional switched networks, such as Fat-Tree's $O(p^3)$, and BCube's $O(p^2)$, where p denotes p-port switch.

Fourthly, Torus is also highlighted by its low cross-cluster latency. In traditional switched DCNs, an inevitably severe problem is that the switching latency (several μs in each hop) and the TCP/IP processing latency (tens of μs) are very high, which leads to a long RTT. Comparatively, TCP/IP stack is not needed in Torus network which saves the long TCP/IP processing time, and the NIC processing delay is also lower than switches (e.g., the processing delay of a real VirtualShare NIC engine is only 0.45 μ s). Besides, Torus also avoids the network oversubscription and provides many equal cost routing paths to avoid network congestion, which can help reduce the queuing delay due to network congestion. Consequently, Torus achieves a much lower end-to-end delay, which is very important in the data center environment.

Fifthly, its high network performance has already been proven in high-performance systems and supercomputers, such as Cray Inc.'s Cray SeaStar (2D Torus) [30], Cray Gemini (3D Torus) [31], IBM's Blue Gene/L (3D Torus) [32] and Blue Gene/Q (5D Torus) [33].

3.2. Routing issues in Torus

Any well qualified routing algorithm design in Torus network must take all important metrics (such as throughput, latency, average path length, load balancing, deadlock free) into consideration. However, the current existing routing algorithms in Torus are far from perfect, as when they improve some certain performance it is usually at the sacrifice of others. Generally the routings in Torus can be divided into two classes: deterministic routing and adaptive routing. A common example of deterministic routing is dimension-ordered routing (DOR) [34], where the message routes dimension by dimension and the routing is directly determined by the source address and destination address without considering the network state. DOR achieves a minimal routing path, but also eliminates any path diversity provided by Torus topology, which results in poor load balancing and low throughput. As an improved two-phase DOR algorithm, Valiant routing (VAL) [35] can achieve optimal worst-case throughput by adding a random intermediate node, but it destroys locality and suffers longer routing path. ROMM [36] and RLB [37] implements good locality, but cannot achieve optimal worst-case throughput. Comparatively, the adaptive routing (like MIN AD [38]) uses local network state information to make routing decisions, which achieves better load balancing and can be coupled with a flow control mechanism. However, using local information can lead to non-optimal choices while global information is more costly to obtain, and the network state may change rapidly. Besides, adaptive routing is not deadlock free, where a resource cycle can occur without routing restrictions which leads to a deadlock. Thus, adaptive routings have to apply some dedicated deadlock-avoiding techniques, such as Turn Model Routing (by eliminating certain turns in some dimensions) and Virtual Channels (by decomposing each unidirectional physical link into several logical channels with private buffer resources), to prevent deadlock.

To summarize, the good features of Torus conclusively demonstrate its superiority in constructing a cost-effective and high performance data center network. However, it also suffers some shortcomings, such as the relatively long routing path, and inefficient routing algorithm with low worst-case throughput. In response to these issues, in this paper we propose some practical and efficient solutions from the perspectives of physical interconnection and routing while inheriting and keeping the intrinsic advantages of Torus topology.

4. NovaCube network design

This section presents the network design and theoretical analysis of *NovaCube*. Before introducing the physical interconnection structure, we firstly provide a theorem with proof, which offers a theoretical basis of *NovaCube* design.

Theorem 4.1. *For any node $A(a_1, a_2, \dots, a_n)$ in a k-ary n-cube (when k is even) if $B(b_1, b_2, \dots, b_n)$ is assumed to be the farthest node from A, then B is unique and B's unique farthest node is exactly A.*

Proof. In a k-ary n-cube, if $B(b_1, b_2, \dots, b_n)$ is the farthest node from $A(a_1, a_2, \dots, a_n)$, where $a_i \in [0, k)$, $b_i \in [0, k)$, then there is:

$$b_1 = \left(a_1 + \frac{k}{2}\right) mod\ k, \dots, b_n = \left(a_n + \frac{k}{2}\right) mod\ k \tag{1}$$

Since the result of $(a_i + \frac{k}{2})\ mod\ k$ is unique, thus $\forall b_i$ is unique and $b_i \in [0, k)$. Hence, A's farthest node B is unique.

Next, assume B's farthest node is $A'(a'_1, a'_2, \dots, a'_n)$, similarly we have:

$$a'_1 = \left(b_1 + \frac{k}{2}\right) mod\ k, \dots, a'_n = \left(b_n + \frac{k}{2}\right) mod\ k \tag{2}$$

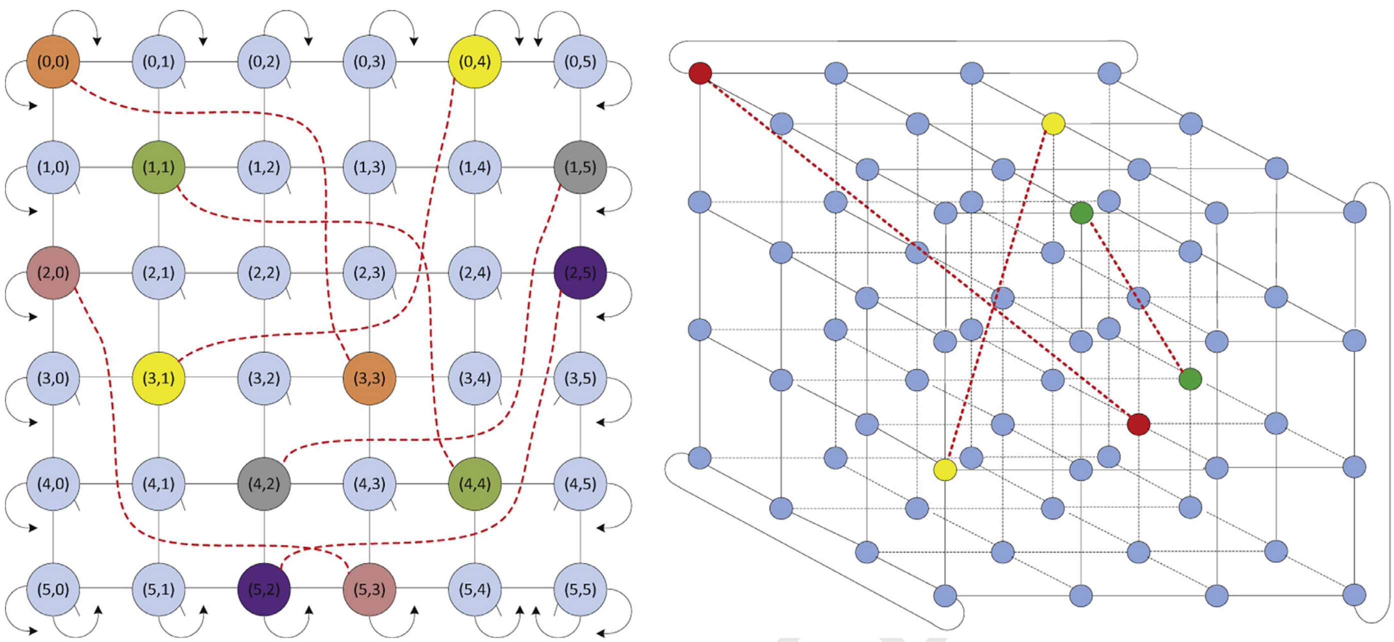

Fig. 2. A 2D 6 × 6-node and 3D 4 × 4 × 4-node *NovaCube* (for simplicity, not all jump-over links are shown).

By combining (1) and (2), we can get:

$$a_i' = \left(b_i + \frac{k}{2}\right) mod\ k = \left\{\left[\left(a_i + \frac{k}{2}\right) mod\ k\right] + \frac{k}{2}\right\} mod\ k$$

$$\because a_i \in [0, k), \therefore a_i + \frac{k}{2} \in \left[\frac{k}{2}, k + \frac{k}{2}\right)$$

(1) For the case of $a_i + \frac{k}{2} \in [\frac{k}{2}, k)$, we have

$$a_i' = \left\{\left[\left(a_i + \frac{k}{2}\right) mod\ k\right] + \frac{k}{2}\right\} mod\ k$$
$$= \left(a_i + \frac{k}{2} + \frac{k}{2}\right) mod\ k = (a_i + k) mod\ k = a_i$$

(2) For the case of $a_i + \frac{k}{2} \in [k, k + \frac{k}{2})$, we have

$$a_i' = \left\{\left[\left(a_i + \frac{k}{2}\right) mod\ k\right] + \frac{k}{2}\right\} mod\ k$$
$$= \left(a_i + \frac{k}{2} - k + \frac{k}{2}\right) mod\ k = a_i mod\ k = a_i$$

As a consequence of the above, $a_i' = a_i$ for $\forall i \in [1, n]$. Therefore, A'(a_1', a_2', …, a_n') = A(a_1, a_2, …, a_n), which means the farthest node from B is exactly A. This ends the proof. □

4.1. NovaCube physical structure

As aforementioned, one critical drawback of *k*-ary *n*-cube topology is its relatively long network diameter, which is as high as $\lfloor\frac{k}{2}\rfloor n$. In order to decrease the network diameter and make routing packets to far away destinations more efficiently, based on the regular *k*-ary *n*-cube, *NovaCube* is constructed by adding some jump-over links connecting the farthest node pairs throughout the network. In a *n*-D Torus the most distant node of (a_1, a_2, …, a_n) can be computed as $((a_1 + \lfloor\frac{k}{2}\rfloor)\ mod\ k,\ (a_2 + \lfloor\frac{k}{2}\rfloor)\ mod\ k, \ldots, (a_n + \lfloor\frac{k}{2}\rfloor)\ mod\ k)$, which guides the construction of *NovaCube*. In brief, the key principle of *NovaCube* is to connect the most distant node pairs by adding one jump-over link. More precisely, there are two construction cases with tiny differences.

4.1.1. Case #1: k is even

When *k* is even, then according to Theorem 3.1 any node's farthest node is unique to each other, and there are $\frac{k^n}{2}$ farthest node pairs, where k^n is the total number of nodes. In this case, all the $\frac{k^n}{2}$ farthest node pairs are connected to each other by one jump-over link. As a result, the degree of each node will be increased from original $2n$ to $2n + 1$. Fig. 2 presents two examples of 2D and 3D *NovaCube*.

4.1.2. Case #2: k is odd

When *k* is odd, one node's farthest node cannot be guaranteed to be unique, nor is the number of node pairs $\frac{k^n}{2}$ an integer either since k^n is odd. In consideration of this fact, we have no alternative but to settle for the second-best choice. The eclectic way is to only construct (*k*-1)-ary *n-NovaCube*, and keep the *k*th node in each dimension unchanged. Noticing that k ($k \geq 1$) is odd, then $k - 1$ is even. Therefore, the construction of *n*-D *NovaCube* with radix $k - 1$ is the same as Case 1. Consequently, there are $\frac{(k-1)^n}{2}$ node pairs with node degree of $2n + 1$ that are connected, and $n^k - n^{k-1}$ nodes with node degree of $2n$ remain unchanged. This way makes a trade-off, however a small one.

4.2. Properties of NovaCube

As with any network, the performance of the *NovaCube* network is characterized by its network diameter, bisection bandwidth, throughput, path diversity and physical cost.

4.2.1. Network diameter

After connecting the most distant node pairs by additional jump-over links, the *NovaCube* network architecture halves the diameter, where the diameter is reduced from original $D_{Torus} = \lfloor\frac{k}{2}\rfloor n$ to be current

$$D_{NovaCube} = \left\lceil \frac{\lfloor\frac{k}{2}\rfloor n}{2} \right\rceil \quad (3)$$

Proof. The network diameter of a regular *k*-ary *n*-cube (*n*-D Torus) is $D_{Torus} = \lfloor\frac{k}{2}\rfloor n$, which means that any node inside the network can reach all the other nodes within $\lfloor\frac{k}{2}\rfloor n$ hops. For any node A

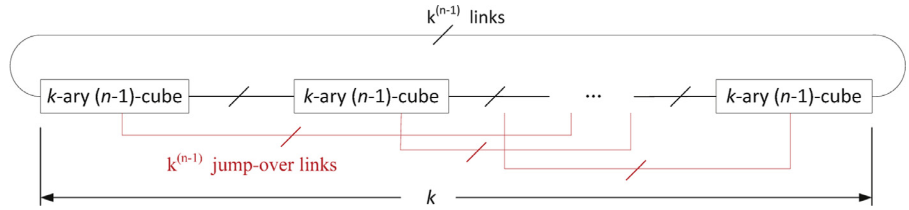

Fig. 3. A *k*-ary *n*-*NovaCube*.

in Torus, we assume that node B is the farthest node from node A. Next, we assume set S_i to denote the nodes that can be reached from node A at the *i*-th hop in Torus, where $i \in [0, \lfloor \frac{k}{2} \rfloor n]$. Then the universal set of all nodes in the network can be expressed as $S = \sum_{i=0}^{\lfloor \frac{k}{2} \rfloor n} S_i$.

After linking all the most distant node pairs (e.g. A and B) in *NovaCube*, if we define S'_i as the set of nodes that are *i* hops from node A in *NovaCube*, then: (1) for the case of $\lfloor \frac{k}{2} \rfloor n$ is even, we have $S'_0 = S_0, S'_1 = S_1 + S_{\lfloor \frac{k}{2} \rfloor n}, S'_2 = S_2 + S_{\lfloor \frac{k}{2} \rfloor n-1}, \ldots, S'_{\lfloor \frac{k}{2} \rfloor n/2} = S_{\lfloor \frac{k}{2} \rfloor n/2} + S_{\lfloor \frac{k}{2} \rfloor n/2+1}$; (2) for the case of $\lfloor \frac{k}{2} \rfloor n$ is odd, we have $S'_0 = S_0, S'_1 = S_1 + S_{\lfloor \frac{k}{2} \rfloor n}, S'_2 = S_2 + S_{\lfloor \frac{k}{2} \rfloor n-1}, \ldots, S'_{\lceil \lfloor \frac{k}{2} \rfloor n/2 \rceil -1} = S_{\lceil \lfloor \frac{k}{2} \rfloor n/2 \rceil -1} + S_{\lceil \lfloor \frac{k}{2} \rfloor n/2 \rceil +1}, S'_{\lceil \lfloor \frac{k}{2} \rfloor n/2 \rceil} = S_{\lceil \lfloor \frac{k}{2} \rfloor n/2 \rceil}$. This demonstrates that in *NovaCube* any node A can reach all nodes of the entire network within $\lfloor \frac{k}{2} \rfloor n/2$ or $\lceil \lfloor \frac{k}{2} \rfloor n/2 \rceil$ hops. Consequently, the network diameter of *NovaCube* is $\lceil \lfloor \frac{k}{2} \rfloor n/2 \rceil$. This ends the proof. □

4.2.2. Bisection bandwidth

The bisection bandwidth can be calculated by summing up the link capacities between two equally-sized parts which the network is partitioned into. It can be used to evaluate the worst-case network capacity [39]. Assume the *NovaCube* network $T(N_1, N_2)$ is partitioned into two equal disjoint sets N_1 and N_2, each element of $T(N_1, N_2)$ is a bidirectional channel with a node in N_1 and another node in N_2. Then the number of bidirectional channels in the partition is $|T(N_1, N_2)|$, or $2|T(N_1, N_2)|$ channels in total, thus the bisection bandwidth is $B_T = 2 \mid T(N_1, N_2) \mid$. For a *k*-ary *n*-*NovaCube* as shown in Fig. 3, when *k* is even, there is even number of *k* *k*-ary (*n*-1)-cube, which can be divided by the minimum bisection into two equal sets with $2k^{n-1}$ regular bidirectional links and $k * \frac{k^{n-1}}{2}$ jump-over bidirectional links. Therefore, the channel bisection bandwidth of *k*-ary *n*-*NovaCube* is computed as:

$$B_T = 2 * \left(2k^{n-1} + k * \frac{k^{n-1}}{2} \right) = k^n + 4k^{n-1} \tag{4}$$

According to the result in [40], the bisection bandwidth of a regular *n*-dimensional Torus with radix *k* is $B_C = 4k^{n-1}$. Therefore, *NovaCube* effectively increases the bisection bandwidth by at least $\frac{B_T - B_C}{B_C} = \frac{k^n + 4k^{n-1} - 4k^{n-1}}{4k^{n-1}} = \frac{k}{4} \geq 25\%$ (k ≥ 1) and the ratio increases accordingly as *k* increases.

4.2.3. Throughput

Throughput is a key indicator of the network capacity to measure a topology. It not only largely depends on the bisection bandwidth, but is also determined by the routing algorithm and flow control mechanism. However, we can evaluate the ideal throughput of a topology under the assumed perfect routing and flow control. The maximum throughput means some channel in the network is saturated and the network cannot carry more traffic. Thus, the throughput is closely related to the channel load. We assume the bandwidth of each channel is b_c and the workload on a channel *c* is ω_c. Then the maximum channel load $\omega_{max} = Max\{\omega_c, c \in C\}$. The ideal throughput occurs when the bottleneck channel is saturated and equal to the channel bandwidth b_c. Thus, the ideal throughput Θ_{ideal} of a topology is

$$\Theta_{ideal} = \frac{b_c}{\omega_{max}} \tag{5}$$

Under uniform traffic pattern, the maximum channel load ω_{max} at the bisection channel has a lower bound, which in turn gives an upper bound on throughput. For a uniform traffic pattern, on average $\frac{k^n}{2}$ packets must go through the B_T bisection channels. If the routing and flow control are optimal, then the packets will be distributed evenly among all bisection channels which results in the best throughput. Thus, the load on each bisection channel load is at least

$$\omega_{max} \geq \frac{\frac{k^n}{2}}{B_T} = \frac{\frac{k^n}{2}}{k^n + 4k^{n-1}} = \frac{k}{2k+8} \tag{6}$$

Consequently, the upper bound on an ideal throughput under uniform traffic can be derived from Eqs. (5) and (6):

$$\Theta_{ideal} = \frac{b_c}{\omega_{max}} \leq \frac{2k+8}{k} b_c \tag{7}$$

This exhibits that *NovaCube* achieves better performance than regular Torus topology in the network capacity with respect to throughput, where the ideal throughput of Torus is only $8b_c/k$ [40]. Here we normalize the worst-case throughput $\widehat{\Theta}_{nw}$ to the network capacity: $\widehat{\Theta}_{nw} = \frac{\omega_{max}}{\omega_{nw}(\widehat{R})}$, where $\widehat{R}$ indicate a routing algorithm. Valiant routing (VAL) [35] is a known worst-case throughput optimal routing algorithm in Torus which obtains $\widehat{\Theta}_{nw} = 50\%$. Thus, an optimal routing algorithm in *NovaCube* can achieve normalized worst-case throughput of at least $\widehat{\Theta}_{nw} = 62.5\%$.

4.2.4. Path diversity

Inherited from Torus topology, *NovaCube* provides a diversity of paths, which can be exploited in routing algorithm by selectively distributing traffic over these paths to achieve load balancing. Besides, the network reliability also greatly benefits much from the path diversity, where the traffic can route around the faulty nodes/links by taking alternative paths.

The number of distinct paths existing in *NovaCube* is too huge to be calculated exactly, for simplicity, we first compute the number of shortest paths in a regular Torus without any jump-over links. Assume two nodes $A(a_1, a_2, \ldots, a_n)$ and $B(b_1, b_2, \ldots, b_n)$ in an *n*-dimensional Torus, and the coordinate distance between A and B in the i^{th} dimension is $\Delta_i = |a_i - b_i|$. Then the total number of shortest paths P_{ab} between A and B is:

$$P_{ab} = \prod_{i=1}^{n} \binom{\sum_{j=i}^{n} \Delta_j}{\Delta_i} = \frac{(\sum_{i=1}^{n} \Delta_i)!}{\prod_{i=1}^{n} \Delta_i !} \tag{8}$$

where the term $\binom{\sum_{j=i}^{n} \Delta_j}{\Delta_i}$ computes the number of ways to choose where to take the Δ_i hops in dimension *i* out of all the remaining hops. It can be seen that a longer distance and a higher dimension result in a larger number P_{ab} of shortest paths. For instance, if given $\Delta_x = 3$, $\Delta_y = 4$, $\Delta_z = 5$ in a 3D Torus, the number of shortest paths is as high as 27720. If we further add a larger

Please cite this article as: T. Wang et al., Towards cost-effective and low latency data center network architecture, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.02.016

number of additional jump-over links in *NovaCube*, the number of paths will be larger. If taking the non-minimal paths into consideration as designed in some routing algorithms, the number of feasible paths is nearly unlimited. The great path diversity of *NovaCube* offers many benefits as aforementioned, but it is also confronted with great challenges in designing an efficient, deadlock free and load balanced routing algorithm.

4.2.5. Average path length

In this subsection, we derive the average path length (APL) for n-D *NovaCube* with an even radix k, and calculate its value for $n = 2$. Due to the symmetry, every node is the same in the k-ary n-*NovaCube*. Hence, we can only consider the APL from a fixed source s to any possible destination d. Here we denote m as the jump-over neighbour of s.

Denote source $s = (0, \ldots, 0)$ and destination $d = (x_1, \ldots, x_n)$, where $x_i \in [0, k-1]$. Then we have $m = (\frac{k}{2}, \ldots, \frac{k}{2})$. Thus, the s-d minimal distance in k-ary n-*NovaCube* is given as:

$$\Delta(s, d) \stackrel{\text{def}}{=} \min\{||s-d||_1, ||m-d||_1 + 1\} \tag{9}$$

where $||\cdot||_p$ is p-norm of the vector meaning that $||x||_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$ for the n-dimensional vector x. Then, the APL is $E[\Delta(s, d)]$. Without loss of generality, for the case $n = 2$, we can have

$$E[\Delta(s,d)] = \frac{1*5 + \sum_{i=2}^{k/2-1}(8i-4)*i + (4k-6)*\frac{k}{2}}{k^2-1} = \frac{\frac{k^3}{3} + \frac{k^2}{2} - \frac{4k}{3} + 1}{k^2-1} \tag{10}$$

Thus, the APL of *NovaCube* approaches to $\frac{k}{3}$ when k is large, which is superior to 2D Torus's $\frac{k}{2}$ [40], and as the dimension increase *NovaCube* reduces more. In the similar way, we can compute the APL for the k-ary n-D *NovaCube*.

4.2.6. Cost-effectiveness

The total number of nodes of a n-dimensional Torus with radix k is k^n and the degree of each node is $2n$, thus the number of links is given as $\frac{2nk^n}{2} = nk^n$. Therefore, the total number of links N_{links} in *NovaCube* can be calculated by summing up nk^n regular links and the number of jump-over links. When k is even, there are $\frac{k^n}{2}$ node pairs connected with jump-over links, so there are $nk^n + \frac{k^n}{2} = (n + \frac{1}{2})k^n$ links in total. Likewise, when k is odd, there are $nk^n + \frac{(k-1)^n}{2}$ links altogether, of which $\frac{(k-1)^n}{2}$ is the number of jump-over links. The calculation can be summarized as below:

$$N_{links} = \begin{cases} \left(n + \frac{1}{2}\right) * k^n & : k is even \\ nk^n + \frac{(k-1)^n}{2} & : k is odd \end{cases} \tag{11}$$

The number of links per server in *NovaCube* is $\widetilde{N}_{links} \le n + \frac{1}{2}$, where it is $n + \frac{1}{2}$ for even k and $n + \frac{(k-1)^n}{2k^n}$ for odd k. Comparatively, the number of links in FatTree[1] is relative to the number of ports p on switches, which is $N_{links} = 3p^3/4$, and the number of links per server in FatTree is 3. Thus, when the dimension $n \le 3$, the cost-effectiveness of *NovaCube* $(n + \frac{1}{2})$ is almost the same as FatTree (3) or even better than FatTree for $n \le 2$. For example, for a 4096-node topology, FatTree uses 12288 links while *NovaCube* has 10240 links for 2D topology and 14336 links for 3D topology. Moreover, *NovaCube* is a switchless architecture, which can save the high expenditure of expensive switches and racks with related cooling costs. Therefore, drawn from the above analysis, *NovaCube* can be regarded as a cost-effective architecture for data centers.

4.2.7. Power savings

The power savings of data center is related to many factors, which can be divided into several categories including hardware level (server/switch using energy-efficient techniques like DVFS, VOVO, PCPG, etc.), network level (energy-aware routing and flow scheduling, job placement, energy-aware VM migration, etc.), architectural level (e.g. switchless architecture), cooling system design (cooling techniques) and the energy resources (e.g. renewable or green resources like wind, water, solar energy, heat pumps) [17,41,42]. NovaCube can be considered as an architectural level approach, which avoids using energy hungry switches. According to the findings in previous studies [19,43,44], the power consumed by switches accounts for around 15% of total power budget. As a switchless architecture, NovaCube will save this portion of power consumption. Besides, intuitively the cooling cost originally induced by cooling the heat emitted by the switches will be saved as well. From this perspective, NovaCube can be regarded as an energy-efficient architecture. Moreover, if some other levels of power saving techniques (e.g. power-aware routings, energy-efficient work placement and VM migration, energy-saving hardware) are employed to NovaCube, more power savings can be achieved.

5. Routing scheme

This section presents the specially designed routing algorithms named PORA for *NovaCube*, which aims to help *NovaCube* achieve its maximum theoretical performance. PORA is a probabilistic weighted oblivious routing algorithm. Besides, PORA is also livelock and deadlock free.

5.1. PORA routing algorithm

Notation 1. The distance between node $A(a_1, a_2, \ldots, a_n)$ and node $B(b_1, b_2, \ldots, b_n)$ in the i-th dimension is denoted as $\Delta_i = ||a_i - b_i||_1$. The distance between A and B is given as $\Delta_{AB} = \sum_{i=1}^{n} \Delta_i$.

Generally, the PORA procedure can be divided into two steps. The first step is to choose routing direction according to the given probability, while the second step is to route within the designated quadrant. Without loss of generality, for simplicity we use 2D *NovaCube* to illustrate PORA.

5.1.1. Direction determination

As shown in Fig. 4, assume a packet needs to route from the source node S to the destination node D, then firstly it needs to decide the direction of its first hop. Since S has five neighbour nodes S_1, S_2, S_3, S_4, M, where M is its jump-over neighbour (although in the case of odd k some special nodes may have no jump-over links, PORA still works correctly), thus it has five directions to route the packet. In order to choose the most beneficial next-hop, each direction is assigned a probability based on the distance between S's next-hop node and destination node. Then PORA chooses the next-hop according to their probabilities, where the probabilistic mechanism can help PORA achieve a good load balancing. The normalized probability function is given as below:

$$p_i = \frac{\frac{1}{\Delta_i^2}}{\sum_{i=1}^{\psi} \frac{1}{\Delta_i^2}} \tag{12}$$

where ψ is the number of neighbour nodes of the source. Take Fig. 4 as an example, the distances between S's neighbour nodes and destination node D are $\Delta_{S_1D} = 4$, $\Delta_{S_2D} = 4$, $\Delta_{S_3D} = 6$, $\Delta_{S_4D} = 6$, $\Delta_{MD} = 3$, thus the probability of choosing S_1 as the next-hop

Please cite this article as: T. Wang et al., Towards cost-effective and low latency data center network architecture, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.02.016

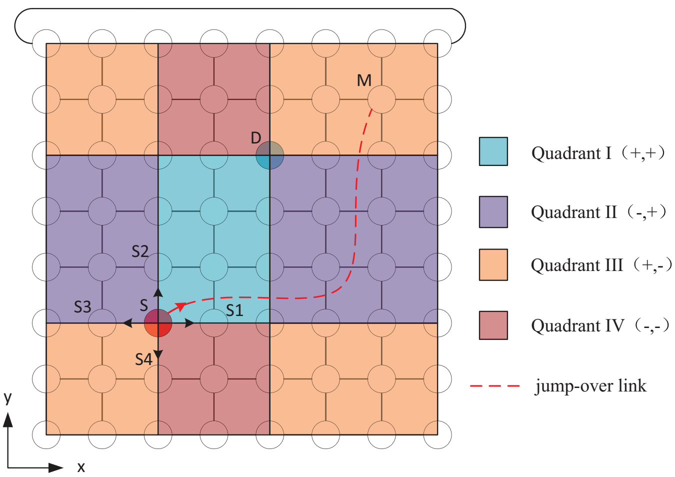

Fig. 4. PORA in an 8 × 8 *NovaCube* (for simplicity, not all wraparound links and jump-over links are displayed).

is $p_{S_1} = \frac{\frac{1}{\Delta^2_{S_1 D}}}{\sum \frac{1}{\Delta_i^2}} = 21.43\%$, and likewise $p_{S_2} = 21.43\%$, $p_{S_3} = 9.524\%$, $p_{S_4} = 9.524\%$, $p_M = 38.10\%$, respectively. Clearly, PORA prefers to choose the shorter path with a higher probability. Each neighbour node (except the jump-over neighbour) corresponds to one quadrant in the Cartesian coordinate system as shown in Fig. 4. For example, in Fig. 4 S_1, S_2, S_3, S_4 correspond to Quadrant I, II, III, and IV, respectively. The division of quadrants is only determined by the source node and destination node.

5.1.2. Routing within quadrant

There are two cases for this step. For the first, if Step 1 finally selects the regular neighbour node other than the jump-over neighbour as the next hop, then all the following routing decisions towards the destination must be restricted within the corresponding quadrant. For the second, in case Step 1 chooses the jump-over neighbour node M (e.g. if it has a smaller distance thus with a higher probability to be chosen) as the next hop, then repeat Step 1 to determine the quadrant by taking M as the source node. This time, in Step 1 PORA will only compute the probability of its regular neighbours without considering its jump-over neighbour, which can avoid jumping back to the original source node that may result in livelock issue.

Once the quadrant is determined, then PORA routes the packet only within the chosen quadrant, where the routing mechanism applied is also probabilistic. At each hop, PORA firstly check if the jump-over link can be used. The jump-over hop can be taken as the candidate next-hop route if and only if it satisfies the following two requirements:

- The jump-over neighbour node is also located within the same quadrant.
- The distance between jump-over neighbour node and destination node is smaller than the distance between regular neighbour node and destination node.

If the requirements cannot be satisfied, then PORA will take the regular neighbour node as its next-hop using traditional DOR (Dimension-Ordered Routing [34]) algorithm, which routes the packet dimension by dimension. Otherwise, if the jump-over link meets the requirements, then the next-hop is selected from the jump-over node and DOR node according to the probability as computed in Eq. (12). This process is repeated at each hop until it reaches the destination.

5.2. Livelock prevention

Livelock occurs when a packet is denied routing to its destination forever even though it is never blocked permanently. It may be travelling around its destination node, never reaching it because the channels it requires are always occupied by other packets. This can occur if non-greedy adaptive routing is allowed (packets are misrouted, but are not able to get closer to the destination). In PORA, once the routing direction is determined at the first step, each of the following hops of PORA will be restricted within the selected quadrant. Moreover, the routing method within the quadrant enables the packet to find its next hop, whose distance to the destination node is always smaller than that from the current node, which guarantees packet delivery. Thus, we claim that PORA is a livelock-free routing algorithm.

5.3. Deadlock-free implementation

As an another notorious problem in Torus networks, deadlock is the situation where packets are allowed to hold some resources while requesting others, so that the dependency chain forms a cycle. Then all these packets must wait forever without reaching the destination, and the throughput will also collapse. The DOR algorithm is proven to be deadlock-free in a mesh network, since there will be no message loop in the network. However, the Torus message loops by itself, thus simply using DOR cannot prevent deadlock. Virtual channels are proposed as a very effective means to prevent deadlock from happening. Virtual channels are used in the loops in a network to cut the loop into two different logical channels, so no cyclic dependency will be formed. Virtual channel is easy to implement by using multiple logical queues and effective in solving deadlock.

To prevent deadlock in our architecture, we first make sure that jump-over links cannot form loops in the routing. We ensure that any jump-over links we choose in the quadrant must be nearer than the previous hop and regular Torus links towards the destination; otherwise, regular Torus links are used. This ensures that the packet will never jump back through the jump-over links. Then, we use the DOR routing to prevent the deadlock in the mesh sub-network. Finally, if the packets have to pass through the wraparound links in the Torus network, we use two virtual channels to cut a Torus loop into two different logical paths. Thus, only two virtual channels are needed in each direction, which is still cost-effective, since the hardware cost increases as the number of virtual channels increases.

6. Flow control

6.1. Credit-based flow control mechanism

Flow control, or known as congestion control, is designed to manage the rate of data transmission between devices or nodes in a network to prevent the network buffers being overwhelmed. Too much data arrives exceeding the device capacity results in data overflow, meaning the data is either lost or must be retransmitted. Thus, the main objective of flow control is to limit packet delay and avoid buffer overflow. In traditional Internet, the protocols with flow control functionality like TCP usually implement the speed matching between fast transmitter and slow receiver by packet discarding and retransmission. More specifically, a node has no alternative but to drop the packet when its buffer space is full (in some special scenarios, packets may be discarded even when the buffer is still available if the packet priority is low or the flow has occupied more than their fair share of resources regarding to QoS restrictions). After packet loss occurs, the network then applies an acknowledgment mechanism to keep track of lost packets,

Please cite this article as: T. Wang et al., Towards cost-effective and low latency data center network architecture, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.02.016

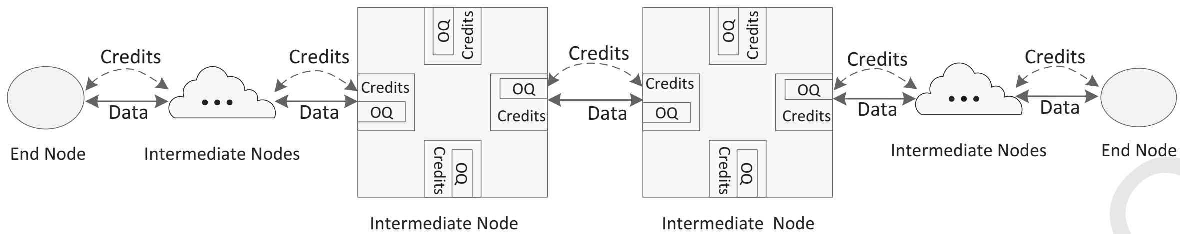

Fig. 5. The credit based flow control in *NovaCube*.

and the sender carries out fast retransmission after receiving three duplicate ACKs or go back to slow start phase after a reasonable timeout RTO (around 200 ms). However, this kind of packet discarding based flow control is unsuitable in the Torus based latency sensitive data center network because of its relatively long routing path with high number of hops. For example, if the congestion point is far from the sender (e.g. the TCP incast usually happens at last hop), then its previous long time transmission will be of waste and the retransmission (or even timeout to slow start) not only brings additional latency but also increases network overhead which may result in a more congested network.

Ideally, a lossless transport protocol is desired. However, it is very difficult to guarantee zero packet loss using sliding window based flow control. Based on this observation, similar to [45] a packet lossless credit based flow control mechanism is adopted in *NovaCube*. The key principle is that each node maintains buffer state of its direct downlink neighbour node for each virtual channel, and only if its downstream node has available space, the sender could get some certain number of credits and transfer corresponding amount of packets.

As illustrated in Fig. 5, a prerequisite for one node to send data from its output queue (OQ) to its next hop node is that its corresponding output port must have enough credits. The default value of credits maintained on one output port equals to the number of packets that can be accepted by its downstream node. The value of credits will be decreased by one whenever its port sends out one packet. Likewise, once the downstream node has new room to accept a new packet, it will send one credit to its upstream node whose relevant port correspondingly increases its credits by one. Usually, there is a very small delay between packet transmission and credit feedback, thus the downstream node should have a bit larger buffer than expected to avoid overflow and achieve maximum throughput, or the downstream node can reserve some safety space when granting credits to its upstream node, for example, to set a threshold (e.g. 80% of total space) that cannot be exceeded. The credit can be transmitted either in-band or out-of-band, where in-band means packets and credits feedback are transmitted over the same channel while out-of-band uses two different channels to transmit packets and credits separately. The in-band feedback is more complicated in implementation but behaves more cost-effective. Comparatively, the out-of-band fashion is an expensive way but easier to be implemented. Nevertheless, these two possible methods can achieve the same effect. Thus, for the sake of simplicity, in our simulations we implement the credit feedback using out-of-band signal.

6.2. Internal structure of nodes

Compared to the traditional network, in *NovaCube* the number of ports on each node is relatively small, thus output queue switching mechanism can be applied, which not only can help achieve better performance but also can simplify the internal structure of nodes. However, it is difficult to implement the flow control adopting output queue switching mechanism. As illustrated in Fig. 5, the credits of a upstream node indicate the queue availability of its downstream node. In order to determine the value of credits that downstream node can accept, the upstream node must first determine the output port of the downstream node that the current packet will go through, which increases the difficulty in implementation.

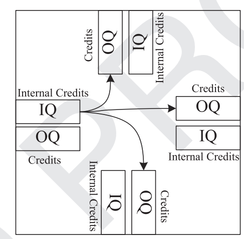

Fig. 6. Internal structure of *NovaCube* node.

In response to this issue, an input queue is introduced at each port as buffering space, as illustrated Fig. 6. The credits of the upstream node denote the available space of input queue in downstream node. Each output queue assigns each input queue a certain number of credits, named as internal credits. Each input queue can forward its packets to the corresponding output queue as long as the input queue has enough assigned credits for this output queue. Besides, each output queue can receive packets from multiple input queues. In fact, if the input/output queues are implemented using centralized shared memory, then the division of input and output queues is merely a logical thing, and the packet scheduling from input queue to output queue is just an action of moving packet pointers. As for the issue of Head of Line (HoL) bocking, all the input queues can be organized as a shared virtual queue, and one virtual input queue can forward certain number of packets to output queue if it has corresponding number of credits. Once packets of an output queue are transmitted to the next hop node, the internal credits of its corresponding input queue will be increased accordingly so that the packets buffered in the input queue can be forwarded to this output queue properly.

7. Geographical address assignment mechanism

7.1. Network layering

Similar to the traditional internet, the protocol stack of *NovaCube* is also divided into five abstraction layers which are application layer, transport layer, network layer, link layer and physical layer. The only difference lies in the network layer, where the traditional internet uses IP address to locate different hosts while *NovaCube* uses coordinates to direct the data transmission with the benefit of topology's symmetry, which can improve the routing efficiency greatly. Except for network layer, the other layers are kept the same without any changes. However, IP addresses must be provided when creating TCP/UDP sockets, yet *NovaCube* only has coordinates. Thus, an adaptation layer, which converts coordinates to IP address format, is need at the network

Please cite this article as: T. Wang et al., Towards cost-effective and low latency data center network architecture, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.02.016

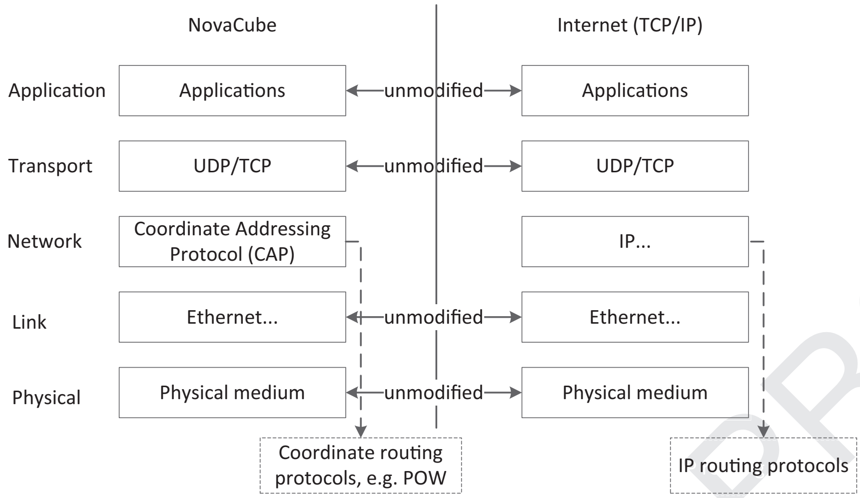

Fig. 7. The NovaCube network abstraction layers.

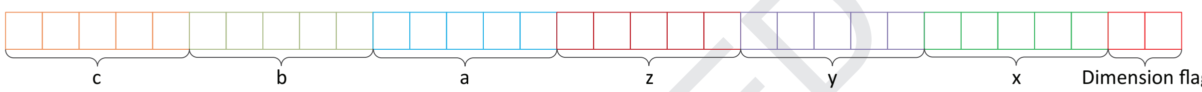

Fig. 8. Coordinated address assignment.

layer. In this way, the transport layer and its upper layers keep unmodified, so that *NovaCube* network can be compatible with legacy TCP/IP protocols and the TCP/IP based application services can run without any changes (Fig. 7).

7.2. Coordinate based geographical address assignment

An address translation mechanism is designed to implement the convention between IPv4 address and *NovaCube* coordinates. As illustrated in Fig. 8, in order to support a NovaCube network with maximum 6 dimensions, the traditional 32-bit IPv4 address is divided into seven segments consisting of six pieces with five bits and one piece with two bits. The coordinate of each dimension is denoted by the five-bit slice, and the remaining two-bit slice is a dimension flag which is used to indicate the number of dimensions of the network. In this way, a 32-bit IPv4 address can support up to six dimensions, where a 6-D *NovaCube* can hold up to 2^{30}= 1,073,741,824 (1 billion) servers, thus this kind of division is reasonable and adequate even for a large scale data center. However, normally the two-bit dimension flag only can support up to 2^2=4 dimensions other than six dimensions. In response to this issue, here we define that only the dimension flag with binary "11" indicates a 6D network address. When the number of dimensions is less than six, the address space of last dimension will not be used. Therefore, when the dimension flag is "10", we make use of the first three bits of the last dimension's address space to represent the specific dimension. The rule of dimension correspondence is illustrated in Table 1, and other values are currently considered illegal.

8. Evaluation

In this section, we evaluate the performance of *NovaCube* and PORA routing algorithm under various network conditions by using network simulator 3 (NS-3) [46]. The link bandwidth is 1 Gbps and each link is capable of bidirectional communications. The default maximum transmission unit (MTU) of a link is 1500 bytes. The propagation delay of a link and the processing time for a packet at a node are set to be 4 μ s and 1.5 μs, respectively. Besides, Weibull

Table 1
The representation of different dimensions.

Dimension flag (binary)	First 3 bits of *c* (binary)	Dimension number (decimal)
11	xxx	6
10	101	5
10	100	4
10	011	3
10	010	2
10	001	1

Distribution is adopted to determine the packet inter-arrival time, and a random permutation traffic matrix is used in our simulation where each node sends/receives traffic to/from exactly one other server.

8.1. Average path length

One of the biggest advantages of *NovaCube* resides in its small average path length (APL). With the benefit of jump-over links, the APL of routing in Torus is significantly reduced. Fig. 9 exhibits the simulation results of average path length using PORA in 2D *NovaCube* and DOR (known to be a shortest path routing) in 2D Torus. The result reveals that PORA indeed achieves a smaller APL in *NovaCube* than DOR in regular Torus, where a smaller APL implies a lower network latency. Even the network diameter of *NovaCube* is also slightly smaller than the achieved APL by DOR in Torus. Moreover, the APL achieved by PORA is already very close to the theoretical analysis, which demonstrates the optimality of PORA.

8.2. Network latency

Generally, the network latency consists of queuing delay at each hop, transmission delay and propagation delay. In order to actually evaluate the overall packet delivery delay in the network, we use the global packet lifetime (the time from packet's generation to the arrival at its destination) to measure the network latency. We simulated the network latency in different sized *NovaCube* and regular Q5

Please cite this article as: T. Wang et al., Towards cost-effective and low latency data center network architecture, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.02.016

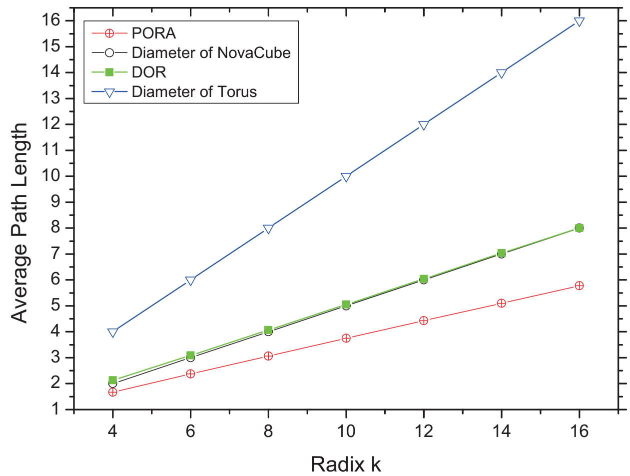

Fig. 9. The average path length using PORA and DOR.

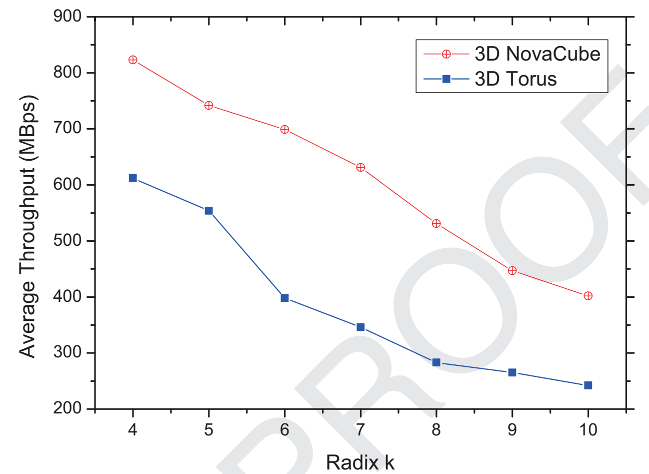

Fig. 11. The throughput of different sized *NovaCube* and Torus.

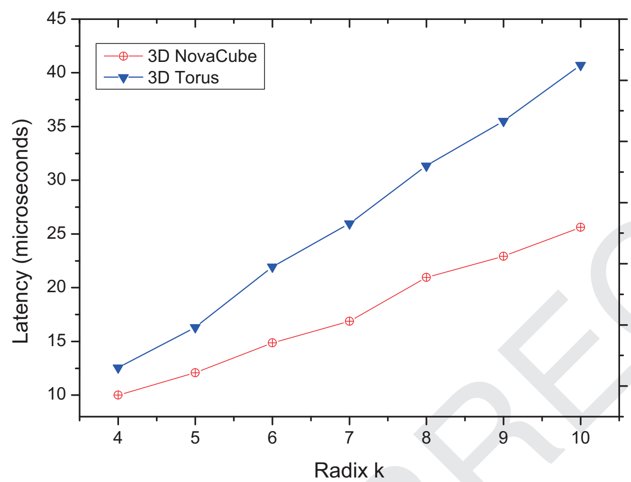

Fig. 10. The network latency of different sized *NovaCube* and Torus.

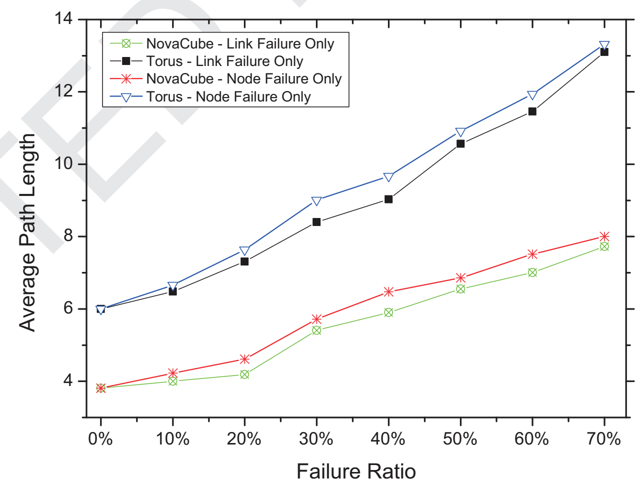

Fig. 12. The average path length under different failures.

Torus networks, varying from $k = 4$ (64 servers) to $k = 10$ (1000 servers), as shown in Fig. 10. It can be seen from the simulation results that compared to the regular Torus network the network latency of *NovaCube* network is reduced by around 40%.

8.3. Throughput

The throughput can be used to measure the network capacity of an architecture, and it is usually limited by bisection bandwidth and also impacted by the routing algorithm. Fig. 11 shows the achieved average throughput in different sized *NovaCube* and Torus network. The result reveals that the average throughput decreases with the increase of network size. *NovaCube* improves the throughput of regular Torus network by up to 90%.

8.4. Fault tolerance

The rich connectivity of Torus-based topologies guarantees the network with high reliability. The node/link failures unlikely cause network disconnections, only may lead to a higher routing path length. Fig. 12 exhibits the simulation results of the average path lengths under different kinds of failure ratios in 512-server (k=8) NovaCube network using PORA routing algorithm and Torus network using DOR routing algorithm. The results show that the average path length increases as the link/node failure ratio increases. NovaCube network with a richer connectivity demonstrates a better performance in fault tolerance than regular Torus. Another finding is that the node failure has a slightly higher impact on the average routing path length.

Table 2
sComparison between 2D NovaCube and 3D NovaCube.

NovaCube	64-server		729-server	
	2D	3D	2D	3D
Average path length	3.06	1.82	9.46	4.17
Latency (ms)	16.12	10.03	49.87	22.92
Throughput (MBps)	616.28	823.42	355.31	447.92

8.5. Comparison between 2D and 3D NovaCube

NovaCubes with different dimensions have different advantages. A lower dimensional NovaCube has lower wiring and routing complexity, and can be easier to be constructed. However, if the network size is large, it is better to be constructed with higher dimensions, and the average path length will also be lower for higher dimensional NovaCube. The network can easily scale up with a higher dimension, thus NovaCube can well support network's future expansion. Table 2 gives some simple simulation results about the performance comparison between 2D NovaCube and 3D Novacube with respect to average path length, latency and throughput. As it can be seen, for the same sized network, 3D

NovaCube achieves lower average path length, lower latency and higher throughput than 2D NovaCube. Therefore, if not consider the wiring and routing complexity, a higher dimensional NovaCube is a better choice.

9. Conclusion

In this paper, we proposed a novel data center architecture named *NovaCube*, and presented its design and key properties. As a switchless architecture, *NovaCube*'s cost-effectiveness is highlighted with regard to its energy consumption and infrastructure cost. As proved, *NovaCube* is also superior to other candidate architectures in terms of network diameter, throughput, average path length, bisection bandwidth, path diversity and fault tolerance. Furthermore, the specially designed probabilistic weighted oblivious routing algorithm PORA helps *NovaCube* achieve near-optimal average path length with better load balancing which can result in a better throughput. Moreover, PORA is also free of livelock and deadlock. The simulation results further prove the good performance of *NovaCube*.

Acknowledgment

This research has been supported by a Grant from Huawei Technologies Co., Ltd. The authors also would like to express their thanks and gratitudes to the anonymous reviewers whose constructive comments helped improve the manuscript.

References

[1] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, in: Proceedings of the ACM SIGCOMM Computer Communication Review, vol. 38, ACM, 2008, pp. 63–74.
[2] 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.
[3] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, S. Lu, DCell: A scalable and fault-tolerant network structure for data centers, ACM SIGCOMM Comput. Commun. Rev. 38 (4) (2008) 75–86.
[4] 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, ACM SIGCOMM Comput. Commun. Rev. 39 (4) (2009) 63–74.
[5] G. Wang, D. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, M. Ryan, c-Through: Part-time optics in data centers, in: Proceedings of the ACM SIGCOMM Computer Communication Review, vol. 40, ACM, 2010, pp. 327–338.
[6] N. Farrington, G. Porter, S. Radhakrishnan, H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, A. Vahdat, Helios: a hybrid electrical/optical switch architecture for modular data centers, in: Proceedings of the ACM SIGCOMM Computer Communication Review, vol. 40, ACM, 2010, pp. 339–350.
[7] T. Wang, Z. Su, Y. Xia, Y. Liu, J. Muppala, M. Hamdi, Sprintnet: a high performance server-centric network architecture for data centers, in: Proceedings of the 2014 IEEE International Conference on Communications (ICC), IEEE, 2014, pp. 4005–4010.
[8] T. Wang, Z. Su, Y. Xia, J. Muppala, M. Hamdi, Designing efficient high performance server-centric data center network architecture, Comput. Netw. 79 (2015) 283–296.
[9] H. Abu-Libdeh, P. Costa, A. Rowstron, G. O'Shea, A. Donnelly, Symbiotic routing in future data centers, ACM SIGCOMM Comput. Commun. Rev. 40 (4) (2010) 51–62.
[10] J.-Y. Shin, B. Wong, E.G. Sirer, Small-world datacenters, in: Proceedings of the 2nd ACM Symposium on Cloud Computing, ACM, 2011, p. 2.
[11] T. Wang, Z. Su, Y. Xia, B. Qin, M. Hamdi, Novacube: a low latency torus-based network architecture for data centers, in: Proceedings of the 2014 IEEE Global Communications Conference (GLOBECOM), IEEE, 2014, pp. 2252–2257.
[12] T. Wang, Z. Su, Y. Xia, M. Hamdi, Clot: a cost-effective low-latency overlaid torus-based network architecture for data centers, in: Proceedings of the 2015 IEEE International Conference on Communications (ICC), IEEE, 2015, pp. 5479–5484.
[13] T. Wang, Z. Su, Y. Xia, M. Hamdi, Rethinking the data center networking: architecture, network protocols, and resource sharing, Access, IEEE 2 (2014) 1481–1496.
[14] Z. Guo, Z. Duan, Y. Xu, H.J. Chao, Cutting the electricity cost of distributed datacenters through smart workload dispatching, IEEE Commun. Lett. 17 (12) (2013) 2384–2387.
[15] Z. Guo, Z. Duan, Y. Xu, H.J. Chao, Jet: electricity cost-aware dynamic workload management in geographically distributed datacenters, Comput. Commun. 50 (2014) 162–174.
[16] L. Rao, X. Liu, L. Xie, W. Liu, Minimizing electricity cost: optimization of distributed internet data centers in a multi-electricity-market environment, in: Proceedings of the INFOCOM, 2010, IEEE, 2010, pp. 1–9.
[17] T. Wang, Y. Xia, J. Muppala, M. Hamdi, S. Foufou, A general framework for performance guaranteed green data center networking, in: Proceedings of the 2014 IEEE Global Communications Conference (GLOBECOM), IEEE, 2014, pp. 2510–2515.
[18] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, N. McKeown, Elastictree: saving energy in data center networks., in: Proceedings of the NSDI, vol. 3, 2010, pp. 19–21.
[19] T. Wang, B. Qin, Z. Su, Y. Xia, M. Hamdi, S. Foufou, R. Hamila, Towards bandwidth guaranteed energy efficient data center networking, J. Cloud Comput. 4 (1) (2015) 1–15.
[20] V. Mann, A. Kumar, P. Dutta, S. Kalyanaraman, Vmflow: leveraging vm mobility to reduce network power costs in data centers, in: Proceedings of the NETWORKING 2011, Springer, 2011, pp. 198–211.
[21] D. Meisner, B.T. Gold, T.F. Wenisch, Powernap: eliminating server idle power, in: Proceedings of the ACM Sigplan Notices, vol. 44, ACM, 2009, pp. 205–216.
[22] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, C. Kozyrakis, Power management of datacenter workloads using per-core power gating, Comput. Archit. Lett. 8 (2) (2009) 48–51.
[23] H. David, C. Fallin, E. Gorbatov, U.R. Hanebutte, O. Mutlu, Memory power management via dynamic voltage/frequency scaling, in: Proceedings of the 8th ACM international conference on Autonomic computing, ACM, 2011, pp. 31–40.
[24] Y. Xia, T. Wang, Z. Su, M. Hamdi, Fine-grain power control for combined input-crosspoint queued switches, in: Proceedings of the Globecom, IEEE, 2014.
[25] D. Abts, M.R. Marty, P.M. Wells, P. Klausler, H. Liu, Energy proportional datacenter networks, in: Proceedings of the ACM SIGARCH Computer Architecture News, vol. 38, ACM, 2010, pp. 338–347.
[26] U. Lee, I. Rimac, V. Hilt, Greening the internet with content-centric networking, in: Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking, ACM, 2010, pp. 179–182.
[27] K. Guan, G. Atkinson, D.C. Kilper, E. Gulsen, On the energy efficiency of content delivery architectures, in: Proceedings of the 2011 IEEE International Conference on Communications Workshops (ICC), IEEE, 2011, pp. 1–6.
[28] Í. Goiri, K. Le, T.D. Nguyen, J. Guitart, J. Torres, R. Bianchini, Greenhadoop: leveraging green energy in data-processing frameworks, in: Proceedings of the 7th ACM European Conference on Computer Systems, ACM, 2012, pp. 57–70.
[29] M. Arlitt, C. Bash, S. Blagodurov, Y. Chen, T. Christian, D. Gmach, C. Hyser, N. Kumari, Z. Liu, M. Marwah, et al., Towards the design and operation of net-zero energy data centers, in: Proceedings of the 2012 13th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), IEEE, 2012, pp. 552–561.
[30] R. Brightwell, K. Pedretti, K.D. Underwood, Initial performance evaluation of the cray seastar interconnect, in: Proceedings of the 13th Symposium on High Performance Interconnects, 2005, IEEE, 2005, pp. 51–57.
[31] R. Alverson, D. Roweth, L. Kaplan, The gemini system interconnect, in: Proceedings of the 2010 IEEE 18th Annual Symposium on High Performance Interconnects (HOTI), IEEE, 2010, pp. 83–87.
[32] N.R. Adiga, M.A. Blumrich, D. Chen, P. Coteus, A. Gara, M.E. Giampapa, P. Heidelberger, S. Singh, B.D. Steinmacher-Burow, T. Takken, et al., Blue gene/l torus interconnection network, IBM J. Res. Develop. 49 (2.3) (2005) 265–276.
[33] D. Chen, N.A. Eisley, P. Heidelberger, R.M. Senger, Y. Sugawara, S. Kumar, V. Salapura, D. Satterfield, B. Steinmacher-Burow, J. Parker, The IBM blue gene/q interconnection fabric, Micro, IEEE 32 (1) (2012) 32–43.
[34] W.J. Dally, H. Aoki, Deadlock-free adaptive routing in multicomputer networks using virtual channels, IEEE Trans. Parallel Distrib. Syst. 4 (4) (1993) 466–475.
[35] L.G. Valiant, G.J. Brebner, Universal schemes for parallel communication, in: Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing, ACM, 1981, pp. 263–277.
[36] T. Nesson, S.L. Johnsson, Romm routing on mesh and torus networks, in: Proceedings of the Seventh Annual ACM Symposium on Parallel Algorithms and Architectures, ACM, 1995, pp. 275–287.
[37] A. Singh, W.J. Dally, B. Towles, A.K. Gupta, Locality-preserving randomized oblivious routing on torus networks, in: Proceedings of the Fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures, ACM, 2002, pp. 9–13.
[38] L. Gravano, G.D. Pifarre, P.E. Berman, J.L. Sanz, Adaptive deadlock-and livelock-free routing with all minimal paths in torus networks, IEEE Trans. Parallel Distrib. Syst. 5 (12) (1994) 1233–1251.
[39] N. Farrington, E. Rubow, A. Vahdat, Data center switch architecture in the age of merchant silicon, in: Proceedings of the IEEE Hot Interconnects, New York, 2009.
[40] W. Dally, B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2004.
[41] Y. Zhang, N. Ansari, Hero: hierarchical energy optimization for data center networks, in: Proceedings of the 2012 IEEE International Conference on Communications (ICC), IEEE, 2012, pp. 2924–2928.

Please cite this article as: T. Wang et al., Towards cost-effective and low latency data center network architecture, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.02.016

[42] Y. Zhang, N. Ansari, On architecture design, congestion notification, TCP incast and power consumption in data centers, Commun. Surv. Tutor. IEEE 15 (1) (2013) 39–64.
[43] A. Greenberg, J. Hamilton, D. Maltz, P. Patel, The cost of a cloud: research problems in data center networks, ACM SIGCOMM Comput. Commun. Rev. 39 (1) (2008) 68–73.
[44] S. Pelley, D. Meisner, T.F. Wenisch, J.W. VanGilder, Understanding and abstracting total data center power, in: Proceedings of the Workshop on Energy-Efficient Design, 2009.
[45] H. Kung, R. Morris, Credit-based flow control for ATM networks, IEEE Netw. 9 (2) (1995) 40–48.
[46] http://www.nsnam.org.