On Architecture Design, Congestion Notification, TCP Incast and Power Consumption in Data Centers

Yan Zhang, *Student Member, IEEE,* and Nirwan Ansari, *Fellow, IEEE*

***Abstract*—With rapid deployment of modern data center networks (DCNs), many problems with respect to DCN architecture, congestion notification, TCP Incast, and power consumption have been observed. In this paper, we provide a comprehensive survey of the most recent research activities in DCNs, with emphasis on the network architecture design, congestion notification algorithms, TCP Incast, and power consumption. We describe in detail the architecture of a typical modern DCN. Furthermore, we investigate the challenges of the typical tree-based hierarchical DCN architecture encountered today and the requirements for the optimal DCN architecture, and we classify the proposed DCN architectures into switch-centric topology and server-centric topology. A section is also devoted to describe some newly proposed DCN architectures. We present a brief overview of the TCP Incast problem along with previously proposed solutions. We also review the recent energy-efficient solutions to minimize the power consumption in DCNs. Finally, we outline possible future research topics in DCNs.**

***Index Terms*—Data center networks (DCNs), modular data center networks, architecture design, data center congestion notification, TCP Incast, TCP throughput collapse, power consumption, energy efficiency, green data centers.**

I. Introduction

DATA centers, facilities with communications network equipments and servers for data processing and/or storage, are prevalent and essential to provide a myriad of services and applications for various private, non-profit, and government systems. They are used and deployed in nearly every sector of the economy, e.g., financial services, media, universities, and government institutions. Many large data center networks (DCNs) have been built in recent years to host online services such as web search and online gaming, distributed file systems such as Google File System (GFS) [1], and distributed Storage System such as BigTable [2] and MapReduce [3]. DCNs are growing rapidly, and the number of servers in DCNs is expanding at an exponential rate. Google hosted approximately 450,000 low-cost commodity servers running across 20 datacenters in 2006 [4], and Microsoft is doubling the number of servers every 14 months.

With rapid deployment of modern high-speed low-latency large-scale DCNs, many problems have been observed in DCNs, such as DCN architecture design, congestion notification, TCP Incast, and power consumption. This survey/tutorial focuses on addressing the above four critical issues in a coherent manner. Readers are referred to [5]–[9] in addressing other issues such as virtual machine migration and routing in DCNs. The typical DCNs are constructed based on a tree-like hierarchical topology [10]. As the scale of modern DCNs is expanding, many problems have been observed in this tree-like hierarchical topology [11], such as scalability, reliability, utilization, and fault tolerance. Several new DCN architectures have been proposed to tackle challenges of typical tree-based architecture [12]–[31]. To support cloud services, existing solutions for the enterprise data center, which were originally developed for smaller data centers in tree-based hierarchical topology, are no longer effective. Typical enterprise data centers are quite different from cloud service data centers [11]. The cloud service data centers must support large economies of scale. The size of cloud DCNs presents an opportunity to leverage economies of scale not present in enterprise DCNs. Automation is a fundamental principle of the design requirement in cloud DCNs; this might be just partially true for enterprise data centers. The leading cost in running enterprise DCNs is the operational staff while such cost is very small (smaller than 5%) in the cloud data centers due to automation. The typical ratio of IT staff members to servers in enterprise data centers is 1:100 while it is 1:1000 for cloud data centers. Furthermore, cloud service DCNs are required to scale out to distribute workload over a large number of low-cost commodity servers and switches, while enterprise DCNs are often optimized for the physical space and number of switches in order to consolidate the workload onto a small number of high-end servers and switches. Here, our objective is to mainly survey the cloud service DCN architectures.

In this survey, we also discuss other problems in DCNs, including congestion notification algorithms [32]–[37], TCP Incast [38]–[40], as well as power consumptions. Congestion notification algorithms have been developed to reduce or remove packet drops at congested switches. Several congestion notification algorithms have been proposed, e.g., Backward Congestion Notification (BCN) [32], [33], Forward Explicit Congestion Notification (FECN) [34], the enhanced FECN (E-FECN) [35], and Quantized Congestion Notification (QCN) [36]. QCN has been developed probably for inclusion in the IEEE 802.1Qau standard to provision congestion notification at the Ethernet Layer or Layer 2 (L2) in DCNs by the IEEE Data Center Bridging Task Group.

Manuscript received 30 January 2011; revised 23 June 2011, 7 October 2011, 28 November 2011, and 10 December 2011.

Y. Zhang and N. Ansari are with the Advanced Networking Lab., Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, 07102 USA. (e-mail:yz45@njit.edu and Nirwan.Ansari@njit.edu)

Digital Object Identifier 10.1109/SURV.2011.122211.00017

TCP throughput collapse has been observed during synchronous data transfers in early parallel storage networks [41]. This phenomenon is referred to as TCP Incast and is attributed to having multiple senders overwhelming a switch buffer, thus resulting in TCP timeout due to packet drops at the congestion switch. TCP Incast has also been observed by others in distributed cluster storage [38], MapReduce [3], and web-search workloads. The main root cause of the TCP Incast analyzed in [38]–[40] is due to the packets drops at the congestion switch that result in TCP timeout.

To provide reliable and scalable computing infrastructure, the high network capacity of DCNs is especially provisioned for worst-case or busy-hour load, and thus data centers consume a huge amount of energy. Report to Congress on Server and Data Center Energy Efficiency [42] assessed trends in the energy use and energy costs of data centers and servers in U.S., and outlined existing and emerging opportunities for improved energy efficiency. It is estimated that data centers and servers consumed about 61 billion kilowatt-hours (kWh) in 2006 (1.5% of the total U.S. electricity consumption) for a total electricity cost of about $4.5 billion. The energy use of data centers and servers in 2006 is estimated to be more than double the electricity that was consumed for this purpose in 2000 and could nearly double again in 2011 to more than 100 billion kWh, representing a $7.4 billion annual electricity cost. However, numerous studies have shown that data center servers rarely operate at full utilization, and it has been well established in the research literature that the average server utilization is often below 30% of the maximum utilization in data centers [43], [44] and a great number of network devices work in the idle state in these richly-connected data networks. At low levels of workload, servers are highly energy-inefficient because even at the idle state, the power consumed is over 50% of its peak power for an energy-efficient server [44] and often over 80% for a commodity server [45]. Barroso and Hölzle [44] showed a mismatch between common server workload profiles and server energy efficiency. Thus, these servers are not power-proportional, as the amount of power used is proportional to the provisioned capacity, not to the workload. They proposed the concept of energy proportional computing systems that ideally consume almost no power when idle and gradually consume more power as the activity level increases. The high operational costs and the mismatch between the data center utilization and power consumption have spurred great interest in improving DCN energy efficiency. Currently, power consumption and energy efficiency is also a major concern for other communication networks, such as wireless cellular networks, optical networks, and future Internet. Several survey papers on power consumption for these different networks [46]–[49] have been reported, and readers are referred to these references for further exploration.

The rest of the paper is organized as follows. We describe a typical modern DCN architecture, challenges encountered in DCNs today, and main requirements for emerging DCN architectures in Section II. Section III categorizes the DCN architectures into switch-center as well as server-center topologies, and introduces some newly proposed DCN architectures. Section IV compares various congestion notification algorithms proposed for DCNs. Section V provides a brief overview of the TCP Incast problem and proposed solutions to mitigate TCP Incast. Section VI summarizes current solutions to minimize power consumptions in DCNs. Finally, Section VII concludes the paper, outlining possible future research topics in DCNs.

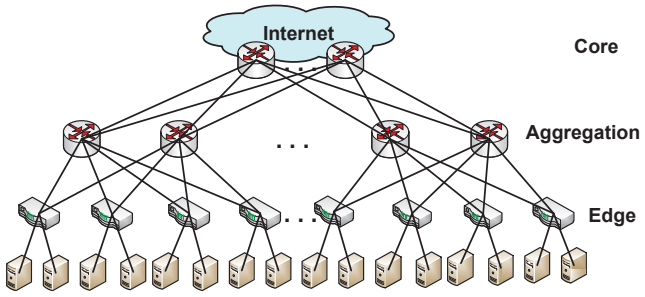


Fig. 1. The conventional DCN architecture for data centers (adapted from Cisco [10]).

II. Typical Data Center Network Architecture

Conventional DCNs are typically constructed based on a tree-based hierarchical topology. In this section, we review the construction of conventional DCNs, summarize the major problems of conventional DCNs, and present requirements for new DCN architectures.

A. Typical DCN Construction

Recommended by Cisco data center infrastructure design guide version 2.5 [10], conventional DCNs typically consist of a two- or three-tier data center switching infrastructure. Two-tier architectures are typically comprised of edge tier switches located at top-of-rack or end-of-row switches, and core tier switches. Two-tier DCNs are usually sufficient to support up to several thousand hosts. For some very large networks, three-tier architectures may be more appropriate, in which an aggregation tier is inserted in the middle of the edge tier and the core tier as shown in Figure 1. All servers attach to DCNs through edge tier switches. There are typically 20 to 40 servers per rack, each singly connected to two aggregation switches for redundancy, and these aggregation switches are further connected to the core tier switches. The aggregation tier switches provide service module integration functions, such as server load balancing, firewalling and Secure Socket Layer (SSL) offloading. The aggregation layer defines the Layer 2 domain size and manages the Layer 2 domain by a spanning tree protocol (STP) such as Rapid Per-VLAN Spanning Tree (R-PVST) or Multiple STP (MSTP). The aggregation layer also provides a primary and secondary router "default gateway" address for all servers across the entire access layer by using default gate redundancy protocols such as Hot-Standby Routing Protocol (HSRP), Virtual Redundant Routing Protocol (VRRP), or Gateway Load Balancing Protocol (GLBP). At the top of the three-tier hierarchy, core tier switches provide the high-speed packet switching backbone for all flows going into and out of DCN. The core tier switches connect to multiple aggregation modules and provide a resilient Layer 3 routing fabric with no single point of failure. Typically, edge tier switches have a number of GigE ports connected to service servers as well as a number of 10 GigE uplinks to

TABLE I
CHALLENGES TO TREE-BASED DCN ARCHITECTURE AND REQUIREMENTS FOR OPTIMAL DCN ARCHITECTURES

	Challenges to Tree-based DCN Architecture	Requirements for Optimal DCN Architecture
Scalability	Scale up : replacing or upgrading existing hardware with advanced higher-capacity modules.	Scale out : adding more individual components to increase the capacity.
Static Network Assignment	Each application is directly mapped to specific physical switches and routers to cover the servers dedicated to the application, which ossifies the server assignment.	Any server could be assigned to any service dynamically according to the fluctuating demand requirements.
Server-to-Server Capacity and Bandwidth Bottleneck	Poor server-to-server capacity: limited by the server oversubscription ratio; server oversubscription ratio increases rapidly when moving up a higher layer in the tree-based DCN hierarchy; bandwidth bottleneck at the aggregation and core switches.	Intensive internal communications between any pair of servers should be supported; any host can potentially communicate with other arbitrary hosts at the full bandwidth of its network interface.
Resource Fragmentation and Agility	Resource is fragmented due to the hierarchical nature of the typical tree-based DCN and large oversubscription at higher level of hierarchy; static network assignment prevents idle servers from being assigned for overloaded services, thus resulting in under-utilization of resources.	The server can be placed anywhere, and can be assigned for any service; the server pools can be dynamically expanded or shrunken; agility improves server utilization, risk management and cost savings.
Reliability & Utilization	The typical tree-based data centers suffer from poor reliability and utilization; multiple paths are not effectively deployed due to the STP used within the layer 2 domain; switches and links in aggregation and core layer run at most 50% of utilization.	A dynamic DCN model is required to deploy network resources when and where they are needed to improve service reliability and maximize server utilization.
Load Balance	Links in the core layer have higher utilization than those in the edge layer.	Traffic needs to be distributed evenly across network paths.
Fault Tolerance	A high-level switch is a potential single-point failure spot for its sub-tree branch in the tree-based DCN architecture; performance degrades ungracefully when hardware fails.	Failure detection should be rapid and efficient; system performance should degrade gracefully when hardware fails; recovery time should be minimal.
Cost	High-end switches incur prohibitive costs to expand the cluster size.	The optimal DCN should deliver scalable bandwidth at moderate cost.
Power Consumption	Power consumption is high and inefficient; overprovisioning of server resources is a huge drain on energy and cooling systems.	Power consumption should be proportional to network utilization.
Performance Isolation	Traffic affects each other from different services if they share the same network sub-tree.	Traffic of different services should not be affected by each other in the optimal DCNs.
Network Capacity	High bandwidth services are constrained by a large server oversubscription ratio.	High network capacity should be provided to better support bandwidth hungry services.

one or more aggregation switches that aggregate and transfer packets between edge switches. The aggregation and core tier switches are typically equipped with 10 GigE ports to provide significant switching capacity for aggregated traffic between edges. Servers are usually isolated into several server groups by virtual LAN (VLAN) partitioning in the Layer 2 domain.

B. Challenges to Typical DCN Architecture and Requirements for Optimal DCN Architectures

Several works have observed the problems of typical tree-based DCN architecture. Table I summarizes the challenges to typical tree-based DCN architecture and requirements for optimal data centers.

- Scalability

Modern DCNs must be scalable to accommodate a large number of servers and allow for incremental expansion. The conventional tree-based DCN architecture increases capacity by scaling up, in other words, by replacing or upgrading existing hardware with advanced high-end ones. However, scaling up is unscalable and expensive, and therefore, newly proposed DCN architectures should be scaled out instead of scaled up by adding more individual components to increase the capacity instead of replacing or upgrading existing hardware with newer, higher capacity and expensive models.

- Static Network Assignment

In tree-based DCNs, each application is mapped to one VLAN, constructed by specific physical switches and routers to cover the servers dedicated to the application. The direct physical mapping application to switches and routers ossifies the server assignment. In the optimal DCN, any server could be assigned to any service dynamically according to the fluctuating demand requirements.

- Server-to-Server Capacity and Bandwidth Bottleneck

Server-to-server capacity is limited by the server oversubscription ratio, which can be calculated by simply dividing the total server connection bandwidth by the total aggregated uplink bandwidth at the access layer switch. The typical oversubscription ratios are 2.5:1 up to 8:1 in large server clusters, and the ratio increases rapidly when moving up in the typical tree-based network architecture hierarchy. Therefore, the bandwidth of a typical tree-based DCN is bottlenecked at the aggregation switches and the core switches. The optimal DCN architecture should support intensive internal communications between any pair of servers in the data center,

regardless of server locations. The server oversubscription ratio should be near to 1:1 as much as possible, indicating that any host can potentially communicate with other arbitrary hosts at the full bandwidth of its network interface.

- Resource Fragmentation and Agility

As a consequence of the hierarchical nature of the typical DCN architecture and the large server oversubscription ratio at the core and aggregation layer, resources are fragmented and isolated, thus limiting the ability to dynamically reassign servers among the applications running in the data center. Owing to limited server-to-server capacity, designers tend to cluster servers near each other in the hierarchy because the distance in the hierarchy affects the performance and cost of the communication. If an application or service grows and requires more servers, resource fragmentation constrains it from using idle servers of other applications. Thus, without agility, each application and service must be pre-assigned with enough servers to meet service demands which are difficult to predict. The resource fragmentation results in under-utilization of resources, and severely limits the entire data center's performance. In the optimal DCN architecture, the server can be placed anywhere and can be assigned to any service, and thus the server pool can be dynamically expanded or shrunk to meet the fluctuating demands of individual services from a large shared server pool. Agility improves server utilization, risk management and cost savings.

- Reliability and Utilization

The typical tree-based DCNs suffer from poor reliability and utilization. Multiple paths are not effectively utilized in typical tree based DCN architecture, since only a single path is used by STP within a layer 2 domain even though multiple paths exist between switches. The typical tree-based DCN topology offers two paths at most. In the aggregation and core layer of tree-based hierarchy, the basic resilience model is 1:1, which leads to at most 50% of maximum utilization of each switch and link. This over-provisioning model used by a typical tree-based DCN architecture for years is rapidly becoming unsustainable. A dynamic DCN model is required to deploy network resources when and where they are needed to improve service reliability and maximize server utilization.

- Load Balance

DCN link loads have been examined by analyzing Simple Network Management Protocol (SNMP) logs collected from 19 DCNs in [50]. Links in the core layer are more heavily utilized than those in the edge layer. Traffic needs to be distributed evenly across network paths in the optimal DCN.

- Fault Tolerance

Fault tolerance is essential for DCNs since failures are common in DCNs. DCNs must be fault tolerant to various types of server failures, server rack failures, link failures and switch failures. A high-level switch may be a single-point failure spot in the tree-based DCN architecture. For instance, a core switch failure may tear down millions of users for several hours. Adding redundant switches may alleviate the problem, but does not solve the problem due to low connectivity of the inherent nature of tree-based architecture. In the optimal DCN architecture, system performance should degrade gracefully when a hardware fails; failure detection should be rapid and efficient, and the recovery time should be minimal.

TABLE II
DATA CENTER NETWORK TOPOLOGY CATEGORIES

Switch-centric Topology	Conventional DCN [10] Two-Table Lookup [12] Monsoon [13] VL2 [14] PortLand [15] Energy Proportional DCN [16] SPAIN [17]
Server-centric Topology	Modular Data Center [18]–[23] DCell [24] BCube [25] MDCube [26] FiConn [27], [28] DPillar [29] CamCube [30], [31]

- Cost

Al-Fares *et al.* [12] estimated the cost of a typical tree-based DCN as a function of the total number of end hosts with different oversubscription ratios. They found that existing techniques for delivering high levels of bandwidth in large clusters incur a significant cost. The high-end switches employed by traditional tree-based DCNs incur prohibitive costs to expand the cluster size. The optimal DCNs should deliver scalable bandwidth at moderate cost.

- Power Consumption

Power consumption is not taken into consideration in the typical tree-based DCN architecture design, and hence power consumption is not efficient. The over-provisioning of server resources in the typical tree-based data center hierarchical architecture is a huge drain on energy and cooling systems. Research in [44] showed that an energy efficient server consumes about half of its full power even when it is idle. It has been well established in the research literature that the average server utilization in data centers is often below 30 percent and servers operate most of the time at between 10 and 50 percent of their maximum utilization levels [44], thus resulting in a poor return on investment (ROI). The power consumption of optimal datacenters should be proportional to utilization.

- Performance Isolation

Normally, multiple services may be hosted in one DCN. In typical tree-based DCNs, a traffic flow may be affected by the traffic flow from other services if they share the same network sub-trees. Traffic of any service should not be affected by traffic of any other service in the optimal DCNs.

- Network Capacity

The large server oversubscription ratio constrains the typical tree-based DCN from providing high bandwidth services. The optimal DCN architecture should be able to provide high network capacity to better support bandwidth hungry services.

III. DCN ARCHITECTURE DESIGN OVERVIEW

As DCNs are growing in size, many problems have been observed, such as scalability, resource fragmentation, reliability, utilization, and fault tolerance. In order to address these

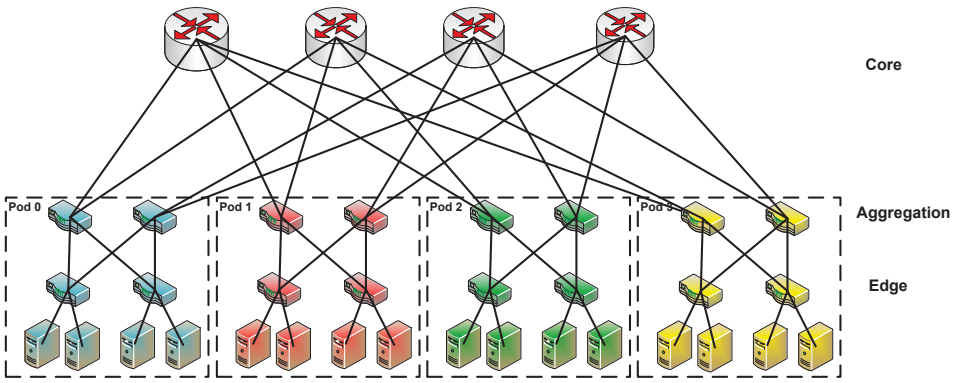


Fig. 2. Fat-tree data center network topology

issues, several DCN architectures have been proposed recently and some of them are reviewed and compared in this section. DCN architectures can be classified into two categories as shown in Table II. One is the switch-centric topology, in which switches provide interconnection and routing intelligence. The other is the server-centric topology, in which servers, with multiple Network Interface Card (NIC) ports, also participate in interconnection and routing.

A. Two-Table Lookup

Al-Fares *et al.* [12] adopted a special instance of a Clos topology [51] called fat-tree [52], by interconnecting commodity off-the-shelf (COTS) switches to build scalable DCNs. This is referred to as the Two-Table Lookup DCN in this survey due to the two-level routing tables used for traffic routing and diffusion. A k-ary fat-tree topology with $k = 4$, shown in Fig. 2, is built with $5k^2/4$ k-port switches that interconnect $k^3/4$ servers. In a k-ary fat-tree topology, there are k pods. Each pod consists of $k/2$ edge switches and $k/2$ aggregation switches, and the edge switches and aggregation switches form a Clos topology by connecting each edge switch to each aggregation switch as shown in Fig. 2. The edge and aggregation switches form a complete bipartite graph in each pod. In each edge switch, $k/2$ ports connect directly to servers and the other $k/2$ ports connect to $k/2$ of the k-ports in the aggregation switches. Thus, each pod is connected to $k^2/4$ servers, and there are $k^3/4$ servers in total. There are $(k/2)^2$ k-port core switches. Each core switch has one port connected to each of k pods, and each pod is connected to all core switches, thus forming a second bipartite graph. A simple, fine-grained method of traffic diffusion between pods, which takes advantage of the fat-tree structure [52], was also proposed.

1) Addressing Scheme: Al-Fares *et al.* [12] assumed that all the IP addresses are allocated in the network within the same private network block. The pod switches are given addresses in the form of $net.pod.switch.1$, where net denotes the private network block, pod denotes the pod number (in $[0, k-1]$), and $switch$ denotes the position of that switch in the pod (in $[0, k-1]$, starting from left to right, bottom to top). The core switch addresses are in the form of $net.k.j.i$, where j and i denote that switch's coordinates in the $(k/2)^2$ core switch grid (each in $[1, (k/2)]$, starting from top-left). The hosts have addresses in the form of $net.pod.switch.ID$, where ID is the host's position in that subnet (in $[2, k/2+1]$, starting from left to right). Therefore, each lower-level switch is responsible for a /24 subnet of $k/2$ hosts (for $k < 256$).

2) Two-Level Lookup Tables Routing Algorithm: To provide the even-distribution mechanism, two-level prefix lookup routing tables were proposed. Entries in the primary table are in the form of 1^m0^{32-m} prefix masks, while entries in the secondary tables are in the form of 0^m1^{32-m} right-handed suffix masks, where m is the mask number. Pod switches act as traffic filter, and pod switches in each pod have prefixes matched to the subnets in that pod. In each edge switch, one matching prefix exists in the primary table for each host in that subnet. In each aggregation switch, a terminating /24 prefix pointing to each subnet in that pod in the form of (net.pod.switch.0/24, port) is inserted in its primary table. For outgoing inter-pod traffic, the pod switches have a default of /0 prefix in the primary table pointing to the secondary table, and in the secondary table a terminating /8 suffix with matching host IDs. Thus, traffic can be evenly distributed upward among the outgoing links to the core switches. In the core switches, a terminating /16 prefix (net.pod.0.0/16, port) is assigned for each network ID in the primary table. Hence, the subsequent packets to the same destination will follow the same path without packet reordering.

B. Monsoon

Greenberg *et al.* [13] proposed Monsoon data center architecture, which creates a huge, flexible mesh switching domain by using programmable commodity layer-2 switches to support any server/any service with low cost. It uses Valiant Load Balancing (VLB) [53] on a mesh of commodity Ethernet switches to realize a hot-spot free intermediate fabric that supports arbitrary traffic patterns in an oblivious manner. Monsoon leverages the programmability of the switches and servers so that VLB can be implemented using only the data-plane features available on commodity switches. It disaggregates the function of load balancing into a group of regular servers, such that load balancing servers can be distributed among racks in the data center leading to greater agility and less fragmentation.

1) Monsoon Architecture: The border routers (BR) of the Monsoon network connect the data center to the Internet, and are connected to a set of access routers (AR) through a layer-3 Equal Cost Multi-Path (ECMP) configuration. The aggregation

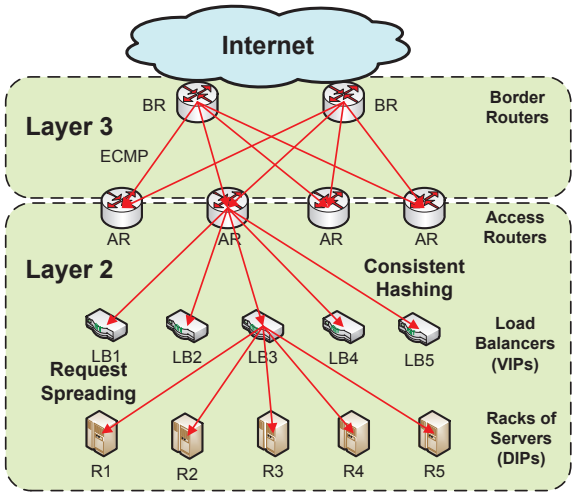


Fig. 3. Overview of the Monsoon architecture.

routers use consistent hashing [54], which is introduced in 1997 as a way of distributing requests among a set of Web servers, to spread the requests going to each application's public virtual IP (VIP) equally over the set of servers acting as load balancers (LB) for that application. Finally, the load balancers spread the requests using an application-specific load distribution function over the pool of servers, identified by their direct IPs (DIPs), which implement the application functionality. Fig. 3 shows an overview of the Monsoon architecture. All servers are connected by a layer 2 network with no oversubscribed links.

2) Monsoon Routing: The sending server encapsulates its frames in a MAC header addressed to the destination's top-of-rack switch, so that switches only need to store forwarding entries for other switches and their own directly connected servers. Each server needs to have the list of MAC addresses for the servers responsible for handling that IP address, and the MAC address of the top-of-rack switch where each of those servers is connected. It also needs to know a list of switch MAC addresses from which it will randomly pick a switch to "bounce" the frame off of. Servers obtain these pieces of information from a Directory Service maintained by Monsoon.

Inside the DCN, traffic is routed by address resolution using the Monsoon directory service. Ethernet frames are encapsulated at the source. Packets at the source will go through a randomly selected intermediate node and the target's top-of-rack switch to the destination. The Monsoon routing is based on source routing, and therefore, the outermost header is the selected intermediate node address as the destination, the middle header is the target's top-of-rack switch address as the destination, and the innermost header is the ultimate destination's MAC address. The sending server's top-of-rack switch forwards the frame towards the VLB intermediate node, which upon receiving the frame removes the outer header and forwards the frame to the top-of-rack switch of the destination. The target's top-of-rack switch will remove the middle routing address in the packet header and forward a normal Ethernet frame with a single header towards the destination server.

External traffic enters and exits the data center through Border Routers, which are connected to a set of Access Routers through a layer-3 ECMP routing configuration. The Access Router is configured to send all the external traffic through an Ingress Server, which is bundled with each Access Router since today's routers do not support the Monsoon load spreading primitive or the packet encapsulation for VLB. The Ingress Servers process all the external traffic redirected by the Access Router and implement the Monsoon functionality and acts as a gateway to the data center. Each Ingress Server has two network interfaces; one is directly connected to an Access Router and the other is connected to DCN via a top-of-rack switch. For packets from the Internet, the Ingress Server receives packets from the Access Router, resolves internal IPs using the Monsoon directory service, and forwards traffic inside the data center using encapsulated Ethernet frames like any other server. The directory service maps the IP address of the layer 2 domains default gateway to the MAC address of the ingress servers, and so packets headed to the Internet flow out through them to the access routers.

C. VL2

Virtual Layer Two (VL2) [14] employs flat addressing to allow service instances to be placed anywhere in the network. In addition, it uses Valiant Load Balancing (VLB) to spread traffic uniformly across network paths, and end-system based address resolution to scale out to huge data centers, without introducing complexity to the network control plane.

1) VL2 Scale-out Topology: VL2 is a three-tier architecture, which shares many features with the conventional DCN architecture. The main difference is that a Clos topology is formed with the core tier and the aggregation tier switches. VL2 shares almost all of the features with the conventional DCN architecture except the rich connectivity between the aggregation switches and the core switches. The large number of paths between every two aggregation switches implies improvement of graceful degradation of bandwidth during link failures. An example of VL2 networks is shown in Fig. 4. A Clos network, formed between aggregation and core switches, provides rich-connection that is quite suitable for VLB.

2) VL2 address resolution: VL2 uses two different IP-address families: location-specific IP addresses (LAs) and application-specific IP addresses (AAs). The network infrastructure operates by using LAs; all switches and interfaces are assigned LAs, and switches run an IP-based link-state routing protocol that disseminates only these LAs. This allows switches to obtain the complete switch-level topology, as well as forward packets encapsulated with LAs along the shortest paths. On the other hand, applications use AAs, which remain unaltered no matter how servers' locations change due to virtual-machine migration or re-provisioning. Each AA is associated with a LA of the top-of-rack (TOR) switch to which the server is connected. The VL2 directory system stores the mapping of AAs to LAs, and this mapping is created when application servers are provisioned to a service and are assigned AA addresses.

3) VL2 routing: Since the network infrastructure operates by using LAs while applications operate by using AAs, the VL2 agent at each server needs to trap packets and encapsulate them with the LA address of the destination edge

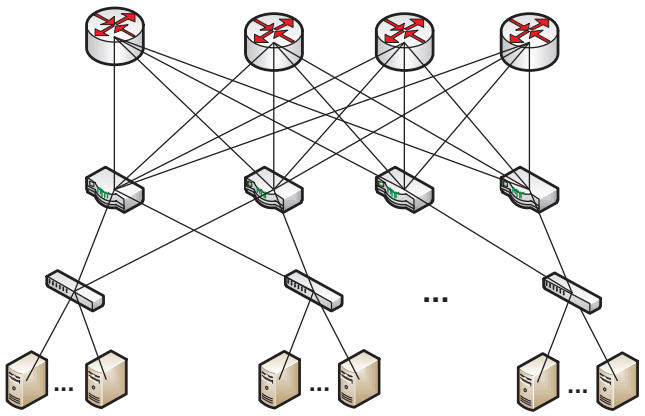


Fig. 4. An example of VL2 network architecture.

switch. Switches run link-state routing protocol and maintain only switch-level topology. Once the packet arrives at the destination edge switch, the switch de-capsulates the packet and delivers it to the destination AA carried in the inner header. VL2 designs directory systems to lookup as well as update AA-to-LA mappings. In order to offer hot-spot-free performance for arbitrary traffic matrices, the VL2 agent at each server implements VLB by sending traffic through one of randomly-chosen intermediate switches. In fact, the same LA address is assigned to all core switches, and the directory system returns this anycast address to agents upon lookup.

D. PortLand

By deploying fat-tree topology, PortLand [15] was designed to deliver scalable, fault-tolerant layer 2 routing, forwarding, and addressing for DCNs. PortLand assigns internal pseudo MAC (PMAC) addresses to all end hosts to encode their positions in the topology. PMAC enables efficient, provably loop-free forwarding with a small number of switch states.

1) PortLand Topology: PortLand is constructed by using fat-tree topology. The edge and aggregation switches in one pod form a complete bipartite graph, and the core switches and the pods form a second Clos topology by connecting each pod with all core switches.

2) Positional PMAC and Fabric Manager: The basis for efficient forwarding and routing as well as virtual machine (VM) migration in PortLand is hierarchical PMAC addressing. Each end host is assigned a unique PMAC address, which encodes the location of the end host in the topology. A PMAC address follows the format *pod:position:port:vmid*, where *pod*(16 bits) represents the pod number of the edge switch, *position*(8 bits) is its position number in the pod, *port*(8 bits) is the switch-local view of the port number the host is connected to, and *vmid*(16 bits) is used to multiplex multiple virtual machines on the same physical machine. The edge switches perform mapping between the actual MAC (AMAC) and PMAC to maintain the unmodified MAC addresses at the end hosts. The pod number and position number within each pod are obtained by edge switches through Location Discovery Protocol (LDP). When an edge switch detects a new source MAC address, it creates an entry in a local PMAC table mapping AMAC to PMAC and simultaneously communicates this mapping to the fabric manager, which is a user process running on a dedicated machine responsible for assisting with ARP resolution, fault tolerance, and multi-cast. Fabric manager maintains soft state about network configuration information such as topology, and it uses this state to respond to ARP requests. When an end host broadcasts an ARP request, its directly connected edge switch will intercept the ARP request for an IP to MAC address mapping and forward the request to the fabric manager. The fabric manager consults its PMAC table to see if an entry is available for the target IP address. If so, it returns the PMAC address to the edge switch; otherwise, the fabric manager will fall back to broadcast to all end hosts to retrieve the mapping by forwarding the ARP request to any core switch, which in turn distributes it to all pods and finally to all edge switches. The target host will reply with its AMAC, which will be rewritten by the edge switch to the appropriate PMAC before forwarding to both the querying host and the fabric manager.

3) Routing in PortLand: PortLand switches use their positions in the global topology to perform more efficient forwarding and routing. PortLand utilizes LDP to obtain the pod numbers and switch numbers for edge switches, aggregation switches and core switches. PortLand switches periodically send a Location Discovery Message (LDM) out their ports to exchange the information with other switches and to monitor the liveness of other switches. LDM contains the information of the switch identifier, pod number, position number, tree level and a switch port facing downward or upward in the multi-rooted tree. The key insight behind LDP is that edge switches receive LDMs only from the ports connected to aggregation switches. If one switch only hears LDM messages from ports smaller than half of its total ports, then it determines that the switch is an edge switch. Aggregation switches set their level once they learn that some of their ports are connected to edge switches. Finally, core switches learn their levels once they confirm that all ports are connected to aggregation switches. Aggregation switches assist in assigning a unique position number for edge switches in each pod. LDP leverages the fabric manager to assign unique pod numbers to all switches in the same pod. A detailed location discovery procedure in PortLand networks has been described in [15].

Core switches learn the pod number of directly-connected aggregation switches. When forwarding a packet, the core switch inspects the pod number in the PMAC destination address to determine the appropriate output port. Aggregation switches learn the position number of all directly connected edge switches. Aggregation switches determine whether a packet is destined for a host in the same pod or not; if so, the packet will be forwarded to an output port corresponding to the position entry in the PMAC. Otherwise, the packet may be forwarded along any of the aggregation switch's links to the core layer. The forwarding protocol in PortLand is provably loop-free. The packet is always forwarded up to either an aggregation or core switch, and then down toward their ultimate destination. Transient loops and broadcast storms are avoided by ensuring that once a packet begins to travel down, it is not possible for it to travel back up the topology.

LDP monitors the switch and link status in PortLand networks. Upon not having received LDM for some configurable period of time, a switch assumes a link failure. The detecting

switch informs the fabric manager about the failures and the fabric manager maintains a logical fault matrix with per-link connectivity information for the entire topology and updates it with new information. Finally, the fabric manager informs all affected switches of the failure, upon which they individually recalculate their forwarding tables based on the new version of the topology.

E. SPAIN

SPAIN ("Smart Path Assignment In Networks") [17] was designed to improve bisection bandwidth by providing multi-path forwarding using low-cost, commodity off-the-shelf Ethernet (COTS) switches over arbitrary topologies. Bisection bandwidth can be thought of as the worst-case network segmentation. A bisection of a network is a network partitioned into two equal parts. The sum of the link capacities connecting these two partitions is the bandwidth between these two parts. The bisection bandwidth of a network is referred to the minimum of such bandwidth among all possible bisections. SPAIN merges a set of pre-computed paths, which exploit the redundancy in the physical wiring, into a set of trees by mapping each tree to a separate VLAN. SPAIN is composed of three key components: path computation, path setup, and path selection. The first two are run off-line, and the last step is run online at the end hosts. The fault tolerance of SPAIN is based on providing multiple paths between any pair of hosts, and on end-host link and switch failure detection and recovery.

1) Path Computation: The goal of the path computation is to compute a set of loop-free paths connecting pairs of end hosts through the given network topology. The centralized configuration mechanism of SPAIN maintains the actual network topology, and configures the switches with the appropriate VLANs. SPAIN leverages on Link-Layer Discovery Protocol (LLDP) to programmatically determine the topology of the entire L2 network. The VLAN assignment is performed by using Simple Network Management Protocol (SNMP) in SPAIN. The set of calculated paths includes the shortest path for each source-destination pair first, and then the set grows to meet the desired path diversity between any source-destination pair. Finally, link-disjoint paths are added to the path set by incrementing the edge switches of the path to improve the usability of a path set.

2) Path Setup: SPAIN maps the pre-computed path set onto a minimal set of VLANs in the Path Setup step. It has been proved that this VLAN Minimization problem, assigning paths to the minimal number of VLANs, is NP hard. A greedy VLAN-packing heuristic algorithm was employed to process the paths computed in Path Computation serially and construct the minimal number of VLANs. Owing to the serial processing of each path in the path set, the greedy VLAN-packing heuristic algorithm does not scale well. In order to improve the scalability of the greedy algorithm, a parallel per-destination VLAN computation algorithm was proposed. It computes the set of subgraphs for each destination, and then, these subgraphs of different destinations are merged to reduce the overall number of VLANs required.

3) Path Selection: The high bisection bandwidth of SPAIN is achieved by spreading traffic across multiple VLANs.

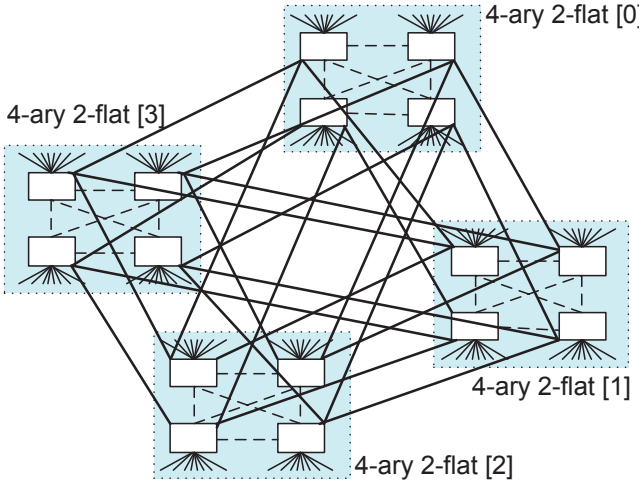


Fig. 5. An example of a (8,4,3) FBFLY topology. It is composed of 4 (8,4,2) FBFLY structure.

SPAIN requires end-host software modifications to make a decision on VLAN selection to efficiently spread load over the network and trigger re-selection of the VLAN for a flow. When a flow starts, the end host finds the set of usable VLANs to reach the edge switch to which the destination is connected. Then, the end host randomly selects a VLAN for this new flow.

F. Energy Proportional DCN

Power consumption has spurred great interest in improving data center energy efficiency. A high-performance energy proportional DCN architecture [16] was proposed by using the flattened butterfly (FBFLY) [55] topology. Abts *et al*. [16] proved that a flattened butterfly topology is inherently more power efficient.

1) Flattened Butterfly DCN Structure: A flattened butterfly is a multi-dimensional direct network, and each dimension is fully connected in an FBFLY. A tuple (c, k, n) can be used to describe a k-ary n-flat with c (concentration) nodes per switch flattened butterfly topology. A (c, k, n) $(n > 2)$ FBFLY network can be built with k $(c, k, n-1)$ FBFLY networks by connecting each switch in each $(c, k, n-1)$ FBFLY network with its peer switches. Fig. 5 shows an example of a $(8, 4, 3)$ FBFLY network topology, which is built with four $(8, 4, 2)$ FBFLY networks by interconnecting each switch in each $(8, 4, 2)$ FBFLY network with its peer in other three groups. In addition, switches are fully connected in each $(8, 4, 2)$ FBFLY network. This $(8, 4, 3)$ FBFLY network houses 128 servers with 16 switches, each with 14 ports.

2) FBFLY Properties: FBFLY network properties can be summarized as follows:

- Scalability: a FBFLY scales exponentially with the number of dimensions. A (c, k, n) flattened butterfly network can house $ck^{(n-1)}$ servers with $k^{(n-1)}$ switches. Each switch is equipped with $c + (k-1)(n-1)$ ports.
- Packaging locality: for the flattened butterfly, the first dimension can use short electrical links. In general, the number of inexpensive electrical cables in a (c, k, n) FBFLY network is $k^{(n-2)}(c * k + k * (k-1)/2)$.

G. DCell

A new concept in the data center design and deployment, called modular data center [18]–[23], was proposed to construct data centers with new building blocks of shipping containers instead of server racks. Each container houses up to a few thousands of servers on multiple racks within a standard 40- or 20-feet shipping container. Shipping container-based MDC is a large pluggable service component with high degree of mobility. It simplifies supply management by hooking up to power, networking, and cooling infrastructure to commence the services, shortens deployment time, increases system and power density, and reduces cooling and manufacturing cost. DCell, BCube and MDCube are recently proposed network architectures for modular data centers.

The $DCell$-based DCN [24] solution has four components: $DCell$ scalable network structure, efficient and distributed routing algorithm that exploits the $DCell$ structure, fault-tolerant routing that addresses various types of failures, and an incremental upgrade scheme that allows for gradual expansion of the DCN size. $DCell$ is a recursively defined, high network capacity structure with mini-switches to interconnect servers, which connect to different levels of $DCells$ switches via multiple links. High-level $DCells$ are constructed recursively by forming a fully-connected graph with many low-level $DCells$. Network traffic in $DCell$ is distributed quite evenly among servers and across links at a server. There is no single point of failure in $DCell$, and $DCell$ addresses various failures at link, server, and server-rack level. Fault tolerance of $DCell$ is attributed to both its rich physical connectivity and the distributed fault-tolerant routing protocol.

1) DCell Construction: $DCell$ employs servers with multiple network ports and mini-switches to construct a recursively defined structure. High-level $DCells$ are constructed recursively by forming a fully-connected graph with many low-level $DCells$. A level-k $DCell_k (k > 0)$ is constructed from $g_k = t_{k-1} + 1$ $DCell_{k-1}$ modules, where t_{k-1} is the total number of servers in one $DCell_{k-1}$. Thus, the total number of servers in a $DCell_k$ is $t_k = t_{k-1} g_k = t_{k-1}(t_{k-1}+1)(k > 0)$. Equipped with n connected servers through one mini-switch, $DCell_0$ ($g_0 = 1$ and $t_0 = n$) is the basic building block to construct higher level $DCells$. To facilitate the $DCell$ construction, each server in a $DCell_k$ is assigned a $(k+1)$-tuple $[a_k, a_{k-1}, \cdots, a_1, a_0]$, where $a_i < g_i (0 < i \leq k)$ indicates at which $DCell_{i-1}$ this server is located, and $a_0 < n$ indicates the index of the server in that $DCell_0$. Each server can be equivalently identified by a unique ID uid_k, taking a value from $[0, t_k)$. The mapping between a unique ID and its $(k+1)$-tuple is a bijection. The ID uid_k can be calculated from the $(k+1)$-tuple using $uid_k = a_0 + \sum_{j=1}^{k} \{a_j \times t_{j-1}\}$, and the $(k+1)$-tuple can also be derived from its unique ID. A server in $DCell_k$ is denoted as $[a_k, uid_{k-1}]$, where a_k is the index of $DCell_{k-1}$ to which this server belongs and uid_{k-1} is the unique ID of the server inside this $DCell_{k-1}$. To construct $DCell_k$ with g_k $DCell_{k-1}$s, server $[i, t_{k-1})$ in the $i^{th} (i \in [0, t_{k-1}))$ $DCell_{k-1}$ will be connected to the i^{th} server in the $j^{th} (j \in [i+1, g_k))$ $DCell_{k-1}$ recursively. A detailed building procedure is described in [24]. Fig. 6 shows an example of $DCell_2$ network structure with $n = 2$. If there are two servers in the basic building structure of $DCell_0$, it will need 3 $DCell_0$ modules to build a $DCell_1$ structure and 7 $DCell_1$ modules to build a $DCell_2$ network.

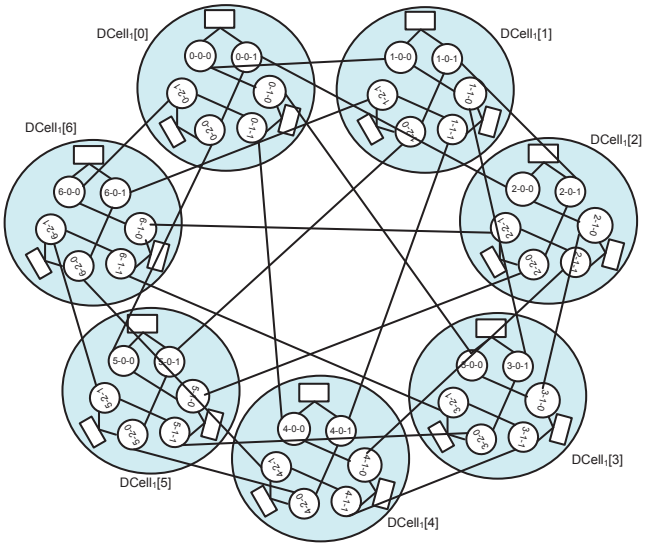


Fig. 6. A $DCell_2$ network structure is composed of 7 $DCell_1$ networks, and each $DCell_1$ is composed of 3 $DCell_0$s.

2) Routing in a DCell: DCellRouting and *DCell* Fault-tolerant Routing (DFR) are the main routing protocols in $DCell$. Both are designed to exploit the *DCell* structure to perform routing. *DCellRouting* is the basic routing protocol in a *DCell* without any failure while DFR is designed to effectively handle various failures due to hardware, software, and power problems.

- *DCellRouting*

DCellRouting is a simple and efficient single-path routing algorithm by exploiting the recursive structure of $DCell$. The design of $DCellRouting$ follows a divide-and-conquer approach. In a $DCell_k$ network, each server is assigned a $(k+1)$-tuple, and so, it is easy to examine the same prefix $[a_k, a_{k-1}, \cdots, a_l]$ that the source server and the destination server belongs to. At level-l, $DCell_{l-1}$ modules form a complete graph, and therefore, the path of *DCellRouting* can be divided into two sub-paths, one within the source $DCell_{l-1}$ and the other within the destination $DCell_{l-1}$, and the sub-path between the source $DCell_{l-1}$ and the destination $DCell_{l-1}$. The upper bound of the path length in $DCellRouting$ is $2^{k+1} - 1$ and the diameter of a $DCell_k$ network is at most $2^{k+1} - 1$. $DCellRouting$ is not a shortest-path routing algorithm, but the performance of $DCellRouting$ is quite close to that of shortest-path routing.

- $DCell$ Fault-Tolerant Routing (DFR)

DFR is a distributed, fault-tolerant routing protocol by employing *DCellRouting* and *DCellBroadcast* as building blocks. A server in the $DCell$ network will broadcast the packet to all its $(k+1)$ neighbors when it broadcasts a packet in a $DCell_k$ by using $DCellBroadcast$. *DFR* deploys local-reroute, local link-state, and jump-up to address link, server and rack failures, respectively. Local-reroute is used to make local decisions to reroute packets to bypass failed links in *DCellRouting*. Local-reroute is efficient in handling link-

failures, but it is not loop free and cannot completely address node failures alone since local-reroute is purely based on $DCell$ topology and does not utilize any kind of link or node states. Local link state can be updated at each node by using $DCellBroadcast$. In a *DCell*, each node knows the status of all the outgoing/incoming links by using *DCellBroadcast* to broadcast the status of each node's $(k+1)$ links periodically or when it detects a link failure. If a whole rack fails, local-reroute and local link-state may result in re-route endlessly within the failed rack, and so jump-up is introduced to address rack failure. With jump-up, the failed server rack can be bypassed.

3) DCell Properties: $DCell$ network properties can be summarized as follows:

- Scalability: the number of servers in a $DCell$ scales doubly exponentially as the node degree increases. The total number of servers in a $DCell_k$ can be expressed as $t_k = g_k g_{k-1} ... g_1 g_0 t_0 = t_0 \prod_{i=0}^{k} g_i$ and $(n+\frac{1}{2})^{2^k} - \frac{1}{2} < t_k < (n+1)^{2^k} - 1$ $(k > 0)$, where n is the total number of servers in a $DCell_0$.
- Rich physical connectivity: each server in a $DCell_k$ network is connected to $k+1$ links.
- Fault Tolerance: there is no single point of failure in *DCell*, and *DCell* addresses link, server, and server-rack failures. Fault tolerance is attributed to the rich physical connectivity and the distributed fault-tolerant routing protocol.
- High network capacity: the bisection width of a $DCell_k$ is larger than $\frac{t_k}{4log_n t_k}$ for $k > 0$. Bisection width denotes the minimal number of links to be removed to partition a network into two parts of equal size. A large bisection width implies high network capacity and a more resilient structure against failures. The bisection of $DCell$ is larger than $\frac{1}{2t_k log_n t_k}$ times that of the bisection of its embedding directed complete graph.
- No severe bottleneck links: under all-to-all traffic pattern, the number of flows in a level-i link is less than $t_k 2^{k-i}$. The difference among the number of flows at different levels is bounded by a small constant, and the lowest level links carry most of the traffic instead of the highest-level links.

H. BCube

The $BCube$ network architecture [25] is another server-centric DCN structure. Similar to $DCell$ networks, $BCube$ networks are built with mini-switches and severs equipped with multiple network ports. The $BCube$ network structure not only provides high one-to-one bandwidth, but also accelerates one-to-several and all-to-all traffic by constructing edge-disjoint complete graphs and multiple edge-disjoint server spanning trees. Furthermore, $BCube$ improves fault-tolerance and load balance.

1) BCube Construction: BCube is a recursively defined structure, composing of servers with multiple network ports, and mini-switches that connect to a constant number of servers. A $BCube_0$ is the basic building block, by simply connecting n servers to an n-port mini-switch. A $BCube_k$ is constructed from n $BCube_{k-1}$ modules with n^k n-port switches for $k > 0$. In order to construct a $BCube_k$, the level-k port of the i^{th} server $(i \in [0, n^k - 1])$ in the j^{th} $BCube_{k-1}$ $(j \in [0, n-1]$) is connected to the j^{th} port of the i^{th} level-k switch. A detailed building procedure of the $BCube$ structure is described in [25]. Fig. 7 shows an example of the $BCube_2$ network structure with $n = 2$. In this example, the basic building block $BCube_0$ is constructed with one 2-port mini-switch connecting 2 servers, $BCube_1$ is built with 2 2-port mini-switches and 2 $BCube_0$s, and $BCube_2$ is built with 4 2-port mini-switches and 2 $BCube_1$s.

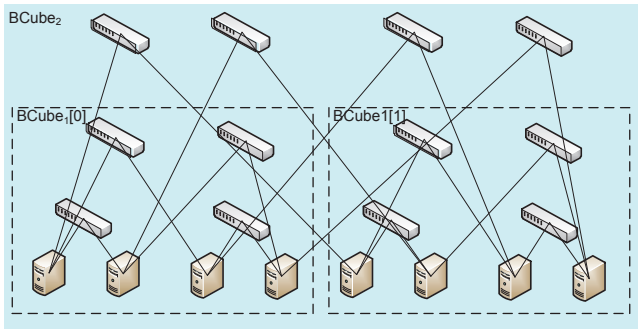


Fig. 7. A $BCube_2$ network structure with $n = 2$.

2) BCube Source Routing (BSR): BSR, a source routing protocol, was proposed to fully utilize the high capacity of $BCube$, and to realize automatic load-balancing by leveraging $BCube$ topological properties. As a source routing protocol, the source server controls the routing path of a packet flow. The source server probes multiple parallel paths for a new coming flow. The probe packets will be processed by the intermediate servers by filling the needed information, and will be returned by the destination server. When the source receives the responses, it will start the path selection procedure based on some kind of metrics, e.g., the path with the maximum available bandwidth and least end-to-end delay. This path selection procedure will be performed periodically to adapt to network failures and network dynamics. The intermediate server will send a path failure message back to the source when it detects a next hop failure for a packet. Upon receiving a path failure message, the source will switch the flow to one of the available paths obtained from the previous probing; otherwise, it will probe the network immediately. In BSR, path switching could occur occasionally, but only one path is used at a given time to avoid the packet out-of-order problem.

3) BCube Properties: $BCube$ network properties can be summarized as follows:

- Low-diameter network: the diameter, which is the longest shortest path among all the server pairs, of a $BCube_k$ is $k+1$. Owing to its low diameter, $BCube$ provides high network capacity for all-to-all traffic.
- Fault-tolerance and load balancing: the number of parallel paths between two servers is upper bounded by $k+1$ in a $BCube_k$. There are n edge-disjoint parallel paths between any two switches in a $BCube_k$. There are $k+1$ edge-disjoint parallel paths between a switch and a server in a $BCube_k$ network. These parallel paths greatly improve fault-tolerance and load balancing. By taking

advantage of these parallel paths, BSR distributes traffic evenly and handles server and/or switch failures without link-state distribution. With BSR, the capacity of $BCube$ decreases gracefully as the server and/or switch failure increases.

- Speedup one-to-several and one-to-all traffic: the time to deliver a file to $r(r \leq (k+1))$ servers or all can be sped up by a factor of r or $k+1$ in a $BCube_k$ than that of the tree and fat-tree structure by constructing r or $k+1$ edge-disjoint server spanning trees in a $BCube_k$.
- No bottleneck links: $BCube$ does not have performance bottlenecks since all links are used equally.
- Aggregate Bottleneck Throughput (ABT) for all-to-all traffic: ABT, defined as the number of flows times the throughput of the bottleneck flow, of $BCube$ increases linearly with the increase of the number of servers. ABT reflects the network capacity under the all-to-all traffic pattern. ABT for a $BCube$ network under the all-to-all traffic model is $\frac{n}{n-1}(N-1)$, where n is the number of ports of a switch and N is the number of servers.
- Supporting bandwidth-intensive applications: $BCube$ supports various bandwidth-intensive applications by speeding-up one-to-one, one-to-several, and one-to-all traffic patterns and by providing high network-capacity for all-to-all traffic.

I. MDCube

$BCube$ was proposed to build high capacity modular DCN structures by using servers with multiple network ports and COTS mini-switches. However, the fundamental barrier to $BCube$ is to scale out to millions of servers by adding more ports to servers and deploying more COTS switches. $MDCube$ [26], built recursively with $BCube$ containers, deploys optical fibers to interconnect multiple $BCube$ containers by using the high speed (10Gb/s) interfaces of COTS switches. Therefore, there are two kinds of links in $MDCube$: one is the normal link connecting servers to switches, and the other is the inter-container high-speed link between switches in different containers. The high-speed interfaces of the $BCube$ containers form a complete graph in $MDCube$. In order to be scalable with the increased number of containers, the complete graph topology is be extended to a generalized cube by introducing dimensions in $MDCube$.

1) MDCube Construction: in order to build a $(D+1)$ dimensional $MDCube$ with $M = \prod_{d=0}^{D} m_d$ $BCube$ containers, where m_d is the number of containers on dimension d. A $BCube$ container, identified by a $(D+1)$-tuple $cID = c_D c_{D-1} \cdots c_0, (c_d \in [0, m_d - 1], d \in [0, D])$, can house $\sum_{d=0}^{D}(m_d - 1)$ switches, with $(m_d - 1)$ switches on dimension d. Each switch, identified by its container ID and its switch ID in its $BCube$ container $\{cid, bwid\}, cid \in [0, M-1], bwid \in [0, \sum_{d=0}^{D}(m_d - 1) - 1]$, contributes its high-speed interface port for $MDCube$ interconnection. $MDCube$ is built recursively with $BCube$ containers. On dimension d, the switch $j - 1 + \sum_{x=0}^{d-1}(m_x - 1)(j \in [i+1, m_d - 1])$ in the container $[c_D \cdots c_{d+1} i c_{d-1} \cdots c_0](i \in [0, m_d - 2])$ will be connected to the switch $i + \sum_{x=0}^{d-1}(m_x - 1)$ in the container $[c_D \cdots c_{d+1} j c_{d-1} \cdots c_0]$. A detailed $MDCube$ building procedure is described in [26].

2) MDCube Routing: by exploring the hierarchy and multi-dimensional structure properties of *MDCube*, a single-path routing, *MDCubeRouting*, is designed for *MDCube*, which works well for the balanced all-to-all communication pattern. However, it may use bandwidth inefficiently to deal with bursty traffic patterns because it tends to only choose the shortest path at the container level. Moreover, it is not fault tolerant when the selected inter-container link breaks. Therefore, a load-balancing and fault tolerant routing algorithm is designed. The detour routing is employed to achieve high throughput by balancing the load among containers, and the fault tolerant routing is deployed by handling inter- and inner-container routing failures.

- MDCubeRouting

MDCubeRouting is designed to correct the tuples of the container ID one by one to reach the destination container, and the order of such correction is controlled by a permutation. The inner-container routing is controlled by $BCube$ routing.

- *MDCubeDetourRouting*

$MDCubeDetourRouting$ balances the load among containers by initiating the routing by a random, container-level jump to a neighboring container, then following with $MDCubeRouting$ from this randomly selected node to the destination to avoid unnecessary overlap at that container.

- Fault Tolerant Routing

Fault tolerant $MDCube$ routing is designed by leveraging the $MDCube$ structure: 1) the source node balances the load on candidate parallel paths at the container level implemented by a loosely controlled source routing protocol; 2) inner-container routing is maintained by $BCube$. For inter-container switch or link failures, a route error message will be generated and sent back to the source node to trigger rerouting at the container level using other available parallel paths.

3) MDCube Properties: $MDCube$ network properties can be summarized as follows:

- Scalability: 1-d $MDCube$ can support up to $n(k+1)+1$ $BCube_k$ containers and 2-d $MDCube$ can support up to $(\frac{n(k+1)}{2}+1)^2$ $BCube_k$ containers. Each $BCube_k$ container can house n^{k+1} servers, and therefore, with 48-port switches, 1-d $MDCube$ can support up to 0.22 million servers and 2-d $MDCube$ can support up to 5.5 million servers.
- $MDCube$ diameter: the diameter of an $MDCube$ network is at most $h = 4k+3+D(2k+3)$, and so the shortest path in $MDCubeRouting$ between any two servers is bounded by $h = 4k + 3 + D(2k + 3)$.
- Parallel paths: the number of parallel paths between any two servers in an $MDCube$ built from $BCube_k$ structures is $k+1$.
- Traffic distribution with $MDCubeRouting$: the number of flows carried on a $BCube$ normal link with $MDCubeRouting$ is around $(2k+1+D(\frac{k^2}{k+1}+1))\frac{N-1}{k+1}$, and the number of flows carried on a high-speed link on dimension i is $\frac{tN}{m_i}$, where t and N are the numbers of servers in a $BCube$ and an $MDCube$, respectively. The bottleneck link is determined by the ratio of the capacity of high-speed links over that of normal links, which can be estimated as $r = \frac{t(k+1)}{m_i(2k+1+D(\frac{k^2}{k+1}+1))}$. The ABT of

$MDCube$ is constrained by normal links and equals to $N(\frac{2k+1}{k+1} + D\frac{k^2+k+1}{(k+1)^2})^{-1}$.

- Traffic distribution with $MDCubeDetourRouting$: the number of flows carried on a $BCube$ normal link is nearly $\frac{(3k+2)t}{k+1}$, and the number of flows carried on a high-speed link is $\frac{nt}{k+1}$. The ratio of the capacity of high-speed links over that of normal speed links between two containers can be estimated as $r = \frac{n}{3k+2}$. ABT of two containers is constrained by normal links and equals to $2t\frac{2t+2}{3k+2}$.

J. FiConn

$FiConn$ [27], [28], a new server-interconnection structure, was proposed to form a scalable and highly effective structure with commodity servers having two built-in Ethernet ports, and low-end commodity switches, based on the observation that the commodity servers used in today's data centers usually come with two built-in Ethernet ports, one for network connection and the other for backup. $FiConn$ uses traffic-aware routing that exploits the available link capacities based on traffic dynamics and balances the usage of different links to improve the overall network throughput.

1) FiConn Architecture: $FiConn$ defines a recursive network structure in levels. A level-k $FiConn_k$ is constructed by many level-$(k-1)$ $FiConn_{k-1}$ modules. Each server s can be identified by a $(k+1)$-tuple, $[a_k, \cdots, a_1, a_0]$, where a_0 identifies s in its $FiConn_0$, and a_l $(1 \leq l \leq k)$ identifies the $FiConn_l$ comprising s in its $FiConn_l$. There are $u_k = a_0 + \sum_{l=1}^{k}(a_l * N_{l-1})$ servers in a $FiConn_k$, where N_l is the total number of servers in a $FiConn_l$.

When constructing a higher-level $FiConn$, the lower-level $FiConn$ structures use half of their available backup ports for interconnections and form a mesh over $FiConn_{k-1}$ modules. $FiConn_0$ is the basic construction unit, which is composed of n servers and an n-port commodity switch connecting the n servers. If there are totally b servers with available backup ports in a $FiConn_{k-1}$, the number of $FiConn_{k-1}$ in a $FiConn_k$, g_k is equal to $b/2+1$. In each $FiConn_{k-1}$, $b/2$ servers out of the b servers with available backup ports are selected to connect the other $b/2$ $FiConn_{k-1}$s using their backup ports, each for one $FiConn_{k-1}$. A detailed $FiConn$ building procedure was described in [27].

2) Traffic-Aware Routing in FiConn: A greedy approach to set up the traffic-aware path hop by hop on each intermediate server was proposed in $FiConn$. Each server seeks to balance the traffic volume between its two outgoing links. Specifically, the source server always selects the outgoing link with higher available bandwidth to forward the traffic. For a level-l $(l > 0)$ intermediate server, if the outgoing link using TOR is its level-l link and the available bandwidth of its level-0 link is higher, its level-l link is bypassed via randomly selecting a third $FiConn_{l-1}$ in the $FiConn_l$ to relay the traffic; otherwise, the traffic is routed by TOR.

3) FiConn Properties: $FiConn$ network properties can be summarized as follows:

- Scalability: the number of servers in $FiConn$, N, grows double-exponentially with $FiConn$ levels. If we denote the total number of servers in a $FiConn_k$ as N_k, $N_k \geq 2^{k+2} * (n/4)^{2^k}$ (for $n > 4$), where n is the number of servers in $FiConn_0$.
- Degree: the average server node degree in $FiConn_k$ is $2 - 1/2^k$.
- Diameter: the diameter of $FiConn$ is $O(\log N)$, which is small and can thus support applications with real-time requirements. The upper bound of the diameter of $FiConn_k$ is $2^{k+1} - 1$.
- Level-l links: the number of level-l links in $FiConn_k$, denoted by L_l, is $L_l = \begin{cases} 4 * L_{l+1}, & \text{if } l = 0 \\ 2 * L_{l+1}, & \text{if } 0 < l < k \end{cases}$
- Fault Tolerance: the bisection width of $FiConn$ is $O(N = logN)$, implying that $FiConn$ may well tolerate port/link faults. The lower bound of the bisection width of $FiConn_k$ is $N_k/(4 * 2^k)$, where N_k is the total number of servers in $FiConn_k$.
- Cost: the number of used switches are much smaller in $FiConn$. To construct a DCN of N servers with n-port switches, the number of switches needed in Fat-Tree is $5N/n$, while the number in $FiConn$ is N/n.
- In traffic-aware routing, the maximum length of routing path between any two servers in $FiConn_k$ is $2 * 3^k - 1$.

K. DPillar

DPillar [29] uses low-end off-the-shelf typical dual-port commodity PC servers and plug-and-play commodity Ethernet switches to develop a scalable data center interconnection architecture. One of the most important feature of DPillar is that DCN can scale out without any physical upgrading of the existing servers.

1) DPillar Architecture: A DPillar network is built with low-cost dual-port commodity servers and n-port Ethernet switches, which are arranged into k server columns $[H_0, H_1, \cdots, H_{k-1}]$ and k switch columns $[S_0, S_1, \cdots, S_{k-1}]$. These server columns and switch columns are placed alternately along a cycle: the server column H_i is neighboring with switch columns S_i and $S_{(i+k-1)\%k}$, and the switch column S_i is neighboring with server columns H_i and $H_{(i+k-1)\%k}$, where the symbol % denotes the modulo operation. Each server in DPillar can be identified as a unique $(k+1)$-symbol label $(C, v^{k-1}, \cdots, v^0)$, where $C(0 \leq C \leq k-1)$ is the index of server column and v^i is an integer between 0 and $(n/2-1)$. Those servers with the same label symbols except v^C $(v^{k-1}, \cdots, v^C_*, \cdots, v^0)$ in server columns H_C and $H_{(C+1)\%k}$ are connected to the same n-port switch in switch column S_C. A detailed DPillar building procedure is described in [29]. A DPillar network can be represented uniquely by a tuple (n, k), where n is the number of ports of the switch, and k is the number of server columns.

2) Routing in DPillar Network: The packet routing and forwarding process in DPillar can be divided into two phases: helix phase and ring phase. In the helix phase, the packet is sent from the source server to an intermediate server whose label is the same as the destination server's label. In the ring phase, the packet is forwarded to the destination from this intermediate server.

Failures, including server failures and switch failures, in DCNs are very common, and therefore DPillar includes a fault-tolerant routing scheme to bypass a wide range of failures

TABLE III
COMPARISON OF DIFFERENT NETWORK STRUCTURES

Structure	Degree	Diameter	BiW	BoD
Ring [57]	2	$\frac{N}{2}$	2	$\frac{N^2}{8}$
2D Torus [57]	4	$\sqrt{N}-1$	$2\sqrt{N}$	$\frac{N\sqrt{N}}{8}$
Full Mesh [57]	N-1	1	$\frac{N^2}{4}$	1
Tree [57]	-	$2\log_{d-1} N$	1	$N^2(\frac{d-1}{d^2})$
Fat-tree [52]	-	$2\log_2 N$	$\frac{N}{2}$	N
Hypercube [56]	$\log_2 N$	$\log_2 N$	$\frac{N}{2}$	$\frac{N}{2}$
Butterfly$^+$ [56]	4	$2l$	$\frac{N}{l+1}$	$O(Nl)$
De Bruijn [58]	d	$\log_d N$	$\frac{2dN}{\log_d N}$	$O(N\log_d N)$
DCell [24]	$k+1$	$< 2\log_n N - 1$	$\frac{N}{4\log_n N}$	$< N\log_n N$
BCube [25]	$k+1$	$k+1$	$\frac{N}{2}$	$\frac{(n-1)}{n}N$
FiConn [27], [28]	2	$< 2^{k+1}-1$	$> \frac{N}{(4*2^k)}$	$N2^k$
DPillar [29]	2	$k+\lfloor k/2 \rfloor$	$(n/2)^k$	$3k\frac{N-1}{2}$

+For Butterfly, $N = (l+1)\times 2^l$

in DPillar. The $Hello$ protocol is employed to discover the connection failures. Server A assumes that server B is not directly reachable if it does not hear its $Hello$ message for a certain period of time. It is relatively straightforward to bypass a failure in the ring phase by changing the forwarding direction from the clockwise direction to the counter-clockwise direction, or vice versa. In order to avoid over forwarding loops, a packet can change its forwarding direction once; otherwise, it will be dropped. In the helix phase, a server can tunnel the packet to bypass failed servers. It always tries to bypass the failed server by sending the packet to a reachable sever in the clockwise neighboring column first. If it cannot send the packet to any servers in this clockwise neighboring column, the server will forward the packet to a server in the counter-clockwise neighboring column to bypass the failure.

3) DPillar Properties: DPillar network properties can be summarized as follows:

- Scalability: A (n,k) DPillar network can accommodate $k(n/2)^k$ servers.
- Cost efficiency: A (n,k) DPillar network is built with $k(n/2)^{k-1}$ switches. The average cost of connecting one server in the DPillar network is $2(U_s/n+U_c)$, where U_s is the unit price of an n-port switch and U_c is the unit price of an Ethernet cable.
- Bisection width: The bisection width of a (n,k) DPillar network is close to $(n/2)^k$.
- Traffic distribution: in all-to-all communications, a server-to-switch link carries at most $3k(N-1)/2$ flows.
- Longest path: the longest path in a (n,k) DPillar network by using helix and ring two-phase routing is $k+\lfloor k/2 \rfloor$.

L. Comparison and Discussion

In recent years, the architecture design of DCNs has been heavily investigated, and several structures have been proposed. Switch-oriented structures, such as tree, Clos network [51], Butterfly [56], Flattened Butterfly [55] and fat-tree [52], cannot support one-to-many/one-to-all traffic well, and are limited by the bottleneck links. Existing server-centric structures either cannot provide high network capacity (e.g., 2-D and 3-D meshes, Torus [57], Ring [57]) or use a large number of server ports and wires (e.g., Hypercube and de Bruijn [58]).

Table III shows the comparison results in terms of node degree, network diameter, bisection width (BiW), and bottleneck degree (BoD). The smaller the node degree, the fewer the links, and the lower the deployment cost. Network diameter is defined as the longest shortest-path among all server pairs, and a low diameter value typically results in efficient routing. Bisection width denotes the minimal number of links to be removed to partition a network into two parts of equal size. A large bisection width implies high network capacity and a more resilient structure against failures. The metric bottleneck degree is defined as the maximum number of flows over a single link under an all-to-all traffic mode. BoD indicates how the network traffic is balanced over all the links. The smaller the BoD, the more balanced of the traffic over all the links. A total of N servers with n switch ports, and k construction levels are assumed for comparison.

The ring structure has a node degree of 2, which is similar to FiConn and DPillar. However, it has a large value of diameter $(N/2)$ and small value of bisection bandwidth (2). 2D Torus only utilizes local links and has a constant degree of 4. However, it has large diameter $(\sqrt{N}-1)$ and BoD (in proportion to $N\sqrt{N}$). Full Mesh has the smallest diameter (1), smallest BoD (1), and large bisection bandwidth $(N^2/4)$, but it has large node degree $(N-1)$, which means high cost of link deployment and complexity of wiring. It is obvious that all of these three structures are not suitable for DCNs even for a very small one with hundreds of servers.

In a tree structure constructed with switches having the same degree d, the diameter of the tree is $2\log_{d-1} N$, but the bisection bandwidth of the tree is 1 and the bottleneck degree is proportional to N^2. The Fat Tree structure introduces more bandwidth into the switches near the root with a multi-stage network to overcome the bottleneck degree problem in the tree structure. However, the Fat-Tree structure does not scale well as the required number of switches scales as $O(N\log N)$, where N is the number of supported servers, thus incurring high cost for deployment.

With a large bisection width of $\frac{N}{2}$ and a small bottleneck degree of $(\frac{N}{2})$, Hypercube is widely used in high-performance computing. However, the node degree of hypercube is $\log_2 N$, i.e., hypercube does not scale well and is thus not practical for large DCNs. Butterfly has a node degree of 4, and the diameter and bottleneck degree are very good, but unfortunately, Butterfly is not tolerant to any failure because there is only one path between any pair of two servers. Although de Bruijn [58] can achieve near-optimal tradeoff between node degree and network diameter, and between good bisection and bottleneck degree, this structure is not practical for a large data center due to the asymmetric links in de Bruijn, which doubles the wire deployment and maintenance effort. Furthermore, wiring in de Bruijin is un-expandable, i.e., even when the diameter is increased by 1, the whole network has to be re-wired.

The node degree in DCell and BCube is as low as $k+1$, where k is the number of levels to deploy DCell and BCube structures, and the diameter is small in both structures. The

problem of DCell is that the traffic is imbalanced: the number of flows carried on a level-i link is less than $N2^{k-i}$. Therefore, the level-0 links carry much higher traffic than the other links. Another problem of DCell is the wiring problem: the high-level links in DCell may travel a relatively long distance. The parallel paths between any two servers in DCell have different lengths, thus making it difficult to speedup one-to-many applications. There are $k+1$ parallel paths between any two servers in a BCube$_k$ with the same path length, and thus one-to-many/one-to-all traffic can be sped up. Another advantage of BCube is that there is no performance bottleneck in all-to-all traffic model since the traffic is distributed equally along all the links. BCube requires more mini-switches and wires than DCell does.

The node degree in FiConn and DPillar is as low as 2, which is smaller than that of DCell and BCube. Hence, the wiring cost in FiConn and DPillar is less than that of DCell and BCube since each server uses only two ports. The diameter of FiConn is $O(\log N)$ and the bisection width is $O(N/\log N)$. The high bisection bandwidth implies multiple available paths between a pair of servers, thus providing fault tolerance of FiConn to port or link failures. Routing in FiConn balances the use of different levels of FiConn links. However, FiConn has lower aggregate network capacity, which results from the less number of links as the tradeoff of easy wiring. The network throughput can be improved by traffic-aware routing in FiConn by utilizing the available link capacities. The network structure of DPillar is totally symmetric and it offers rich connections between servers. DPillar has balanced network capacity and does not incur network bottleneck. In all-to-all communications, a server-to-switch link carries at most $3k(N-1)/2$ flows.

IV. Congestion Notification in Data Centers

Owing to the inherent merits of Ethernet, such as low cost, ubiquitous connectivity, and ease of management, Ethernet has become the primary network protocol for computer-to-computer communications in a DCN. However, Ethernet was designed for best-effort transmissions that may drop packets when the network or switches are busy. In order to address issues raised within a data center, such as increasing demand for higher bandwidth and performance, and low latency interconnect for high performance cluster computing, new Ethernet protocols are being developed by two separate standards bodies, the Internet Engineering Task Force (IETF) [59] and the IEEE Data Center Bridging Task Group of IEEE 802.1 Working Group [60].

Traditionally, transport protocols, such as TCP, are responsible for reliable transmissions in IP networks. Thus, one area of Ethernet extensions is to provide congestion notification to enhance transport reliability without penalizing the performance of transport protocols. Therefore, congestion notification is a Layer 2 traffic management protocol that monitors the queuing status at the Ethernet switches and pushes congestion to the edge of the network at where the transmission rate limiters shape the traffic to avoid frame losses. Reliable transport protocols, such as TCP that has congestion control mechanism, can also benefit from congestion notification protocols since it can react to congestions in a timelier manner. With the increase of Ethernet link rate to 10 Gbps in data centers, congestion notification protocols are also able to enhance the bandwidth usage more effectively.

Project IEEE 802.1Qau is concerned with specifications of an Ethernet Layer or Layer 2 congestion notification mechanism for DCNs deliberated in the IEEE Data Center Bridging Task Group. Several congestion notification algorithms have been proposed, e.g., BCN [32], FECN [34], enhanced FECN (E-FECN) [35], and QCN [36]. The system models of these congestion notification algorithms as shown in Fig. 8; BCN, FECN, E-FECN and QCN are all queue-based congestion notification algorithms. They all assume that congestion detector is integrated into the switch, where congestion happens. The switches detect the congestion and generate the feedback congestion message to the source based on the calculated congestion measure, and the rate regulators at the sources will adjust the rate of individual flows according to congestion feedback messages received from switches. In this section, we will summarize and compare the congestion notification algorithms proposed for Ethernet extensions in data centers.

A. Backward Congestion Notification (BCN)

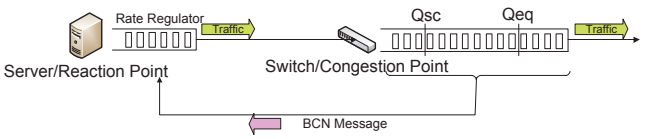
The BCN mechanism as shown in Fig. 8(a) for DCNs was introduced by Bergasamo *et al.* at Cisco [32]. BCN is also known as Ethernet Congestion Manager (ECM) [33]. BCN works in three phases: Congestion Detection, Signaling, and Source Reaction.

1) Congestion Detection: Two thresholds Q_{eq} (equilibrium queue length) and Q_{sc} (severe congestion queue length) are used to indicate tolerable congestion levels at the switch. The switch samples the incoming packets with probability P_m. When a packet is sampled, the congestion measure e_i is calculated as $e_i = -(Q_{off}(t) + w * Q_\delta(t))$, where $Q_{off} = q(t) - Q_{eq}$, $Q_\delta = Q_a - Q_d$, $q(t)$ denotes the instantaneous queue-size, Q_a and Q_d denote the number of arrived and departed packets between two consecutive sampling times, respectively, and w is a non-negative constant weight.

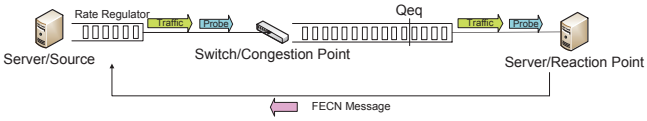
2) Backward Signaling: The arriving packets at the switch are sampled with probability P_m and for each sampled packet the feedback BCN message, which uses the 802.1Q tag format [32], is generated as follows:

- if the packet does not contain rate regulator tag
  - if $q(t) < Q_{eq}$, no BCN message is sent.
  - if $Q_{eq} < q(t) < Q_{sc}$, normal BCN message is sent.
  - if $q(t) > Q_{sc}$, BCN STOP message is sent.
- if the packet contains rate regulator tag
  - if $q(t) < Q_{eq}$ and the CPID field in the 802.1Q tag, which is the ID for the congestion point, matches with this switch's ID, a positive BCN message is sent.
  - if $Q_{eq} < q(t) < Q_{sc}$, normal BCN message is sent.
  - if $q(t) > Q_{sc}$, BCN STOP message is sent.

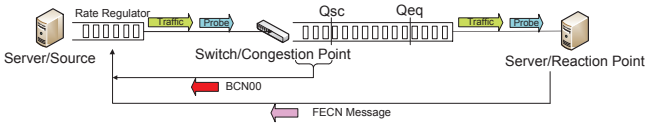
3) Source Reaction: When a STOP message is received at the source, the rate regulator stops sending packets for a random period, and recovers with a rate of C/K, where C is the capacity of the bottleneck link, and K is a constant depending on the number of flows in the network. When a BCN normal message is received at the source, the rate



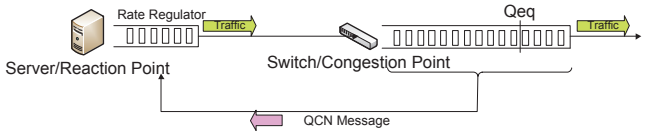
(a) BCN system model



(b) FECN system model



(c) E-FECN system model



(d) QCN system model

Fig. 8. System model overview of congestion notification algorithms for data center networks.

regulator adjusts its rate by using a modified Additive Increase and Multiplicative Decrease (AIMD) algorithm as follows:

$$r_i = \begin{cases} r_i + G_i e_i R_u & (\text{if } e_i > 0) \\ r_i(1 + G_d e_i) & (\text{if } e_i < 0) \end{cases}$$

where R_u represents the increase rate unit parameter, and G_i and G_d denote the additive increase and the multiplicative decrease gain parameters, respectively.

B. Forward Explicit Congestion Notification (FECN) and Enhanced FECN (E-FECN)

As shown in Fig. 8(b), FECN is a close-loop explicit rate feedback control mechanism. All flows are initialized with the full rate. The sources periodically probe the congestion condition along the path to the destination. The rate field in the probe message is modified along the forward path by the switches if the available bandwidth at each switch in the forward path is smaller than the value of the rate field in the probe message. When the sources receive the probe messages returned back from the destination, the rate regulator adjusts the sending rate as indicated in the received message. All flows are treated fairly in FECN since the same rate is advertised by the switch. FECN uses rate based load sensor to detect congestion. At the congestion point, the switch periodically measures the average arrival rate A_i and the instantaneous queue length q_i during the interval, where i is the index of the measurement interval. The effective load can be measured as $\rho_i = \frac{A_i}{f(q_i) \times C}$, where C is the link capacity, and $f(q_i)$ is the hyperbolic queue control function which is defined below to ensure a constant queue length:

$$f(q_i) = \begin{cases} \frac{aQ_{eq}}{(a-1)q_i + Q_{eq}} & \text{if } q_i \le Q_{eq}, \\ max(c, \frac{bQ_{eq}}{(b-1)q_i + Q_{eq}}) & \text{otherwise,} \end{cases}$$

where a, b, c are constants. For $q < Q_{eq}$, $f(q) > 1$. Then, the queue control function attempts to aggressively increase the advertised rate. For $q > Q_{eq}$, generally $f(q) < 1$. Then, the queue control function attempts to aggressively decrease the advertised rate.

The bandwidth allocation can be calculated as

$$r_{i+1} = \frac{r_i}{\rho_i} = \frac{Cf(q_i)}{\frac{A_i}{r_i}}$$

where $Cf(q_i)$ could be thought of as the available effective bandwidth, and $N = \frac{A_i}{r_i}$ is the effective number of flows. If $r_{i+1} < r$ where r is the rate value in the rate discovery probes, then the rate value in the probe message will be marked with r_{i+1}.

As shown in Fig. 8(c), almost all E-FECN operations assume the same operations as those in FECN, except that, the switches are allowed to feed back to the source directly under severe congestion ($q(t) > Q_{sc}$) in E-FECN. Under severe congestion, the switch sends a specific BCN00 message, which causes the rate regulator at the source to reduce to a low initial rate upon receiving this BCN00 feedback message.

C. Quantized Congestion Notification (QCN)

The QCN algorithm is composed of two parts as shown in Fig. 8(d): switch or congestion point (CP) dynamics and rate limiter (RL) or reaction point (RP) dynamics. At CP, the switch buffer attached to an oversubscribed link samples incoming packets and feeds back the congestion severity level back to the source of the sampled packet. While at RP, RL associated with a source decreases its sending rate based on congestion feedback message received from CP, and increases its rate voluntarily to recover lost bandwidth and probe for extra available bandwidth.

1) Congestion Point Algorithm: The goal of CP is to maintain the buffer occupancy at a desired operating point, Q_{eq}. CP samples the incoming packet with a probability depending on the severity of congestion measured by F_b, and computes the severity of congestion measurement F_b. Fig. 9(a) shows the sampling probability as a function of $|F_b|$. F_b is calculated as $F_b = -(Q_{off} + w * Q_\delta)$, where Q_{off} and Q_δ are defined similarly as those in the BCN algorithm, and w is a non-negative constant, taken to be 2 for the baseline implementation. F_b captures a combination of queue-size excess Q_{off} and rate excess Q_δ. Indeed, Q_δ is the derivative of the queue-size, and equals the input rate less the output rate. Thus, when F_b is negative, either the buffers or the link or both are oversubscribed and a congestion message containing the value of F_b, quantized to 6 bits, is sent back to the source of the sampled packet; otherwise, no feedback message is sent.

2) Reaction Point Algorithm: The RP algorithm adjusts the sending rate by decreasing the sending rate based on the congestion feedback message received from CP, and by increasing the sending rate voluntarily to recover lost bandwidth and

probe for extra available bandwidth.

Rate decrease: when a feedback message is received, the current rate (CR) and target rate (TR) are updated as follows:

$$\begin{aligned} TR &= CR \\ CR &= CR(1 - G_d|F_b|) \end{aligned}$$

where the constant G_d is chosen to ensure that the sending rate cannot decrease by more than 50%, and thus $G_d * |F_{bmax}| = \frac{1}{2}$, where F_{bmax} denotes the maximum of F_b.

Rate increase: Two modules, Byte Counter (BC) and Rate Increase Timer, are introduced at RP for rate increases. Byte Counter is a counter at RP for counting the number of bytes transmitted by RL. As a clock at RP employed for timing rate increase, Rate Increase Timer is introduced to allow fast bandwidth recovery when the sending rate is very low while tremendous bandwidth becomes available.

Rate increases at RP occur in Fast Recover (FR) and Active Increase (AI) phases. The byte counter is reset every time a rate decrease is applied and enters the FR state. At the FR state, BC counts data bytes transmitted by RL and increases the BC cycle by 1 when the $BC\_THRESHOLD$ bytes are transmitted. After each cycle, RL increases its rate to recover some of the bandwidth it lost at the previous rate decrease episode. Thus, the goal of RP in FR is to rapidly recover the rate it lost at the last rate decrease episode. After the $FR\_THRESHOLD$ cycles (where $FR\_THRESHOLD$ is a constant chosen to be 5 cycles in the baseline implementation), BC enters the AI state to probe for extra bandwidth on the path. In the AI phase, RP will transmit $BC\_THRESHOLD/2$ bytes data in each BC cycle.

The Rate Increase Timer functions similarly as BC. It is reset when a feedback message arrives and enters the FR state. In the FR state, this timer completes one cycle of T ms. After $FR\_THRESHOLD$ cycles, it enters the AI state where each cycle is set to $T/2$ ms long.

The BC and Rate Increase Timer jointly determine rate increases at RL. After a feedback message is received, they each operate independently and execute their respective cycles of FR and AI. The QCN control mechanism is summarized in Fig. 9(b). BC and Timer determine the state of RL and the sending rate is updated as follows:

1) RL is in FR if both the Byte Counter and the Timer are in FR. In this case, when either the Byte Counter or the Rate Increase Timer completes a cycle, TR remains unchanged while CR is updated as $CR = (CR + TR)/2$.

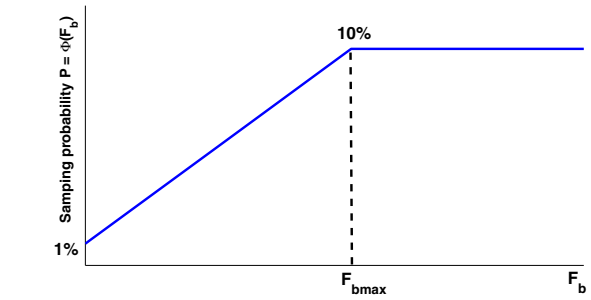
2) RL is in AI if either BC or the Timer is in AI. In this case, when either BC or Timer completes a cycle, TR and CR are updated as:

$$\begin{aligned} TR &= TR + R_{AI} \\ CR &= \tfrac{1}{2}(CR + TR) \end{aligned}$$

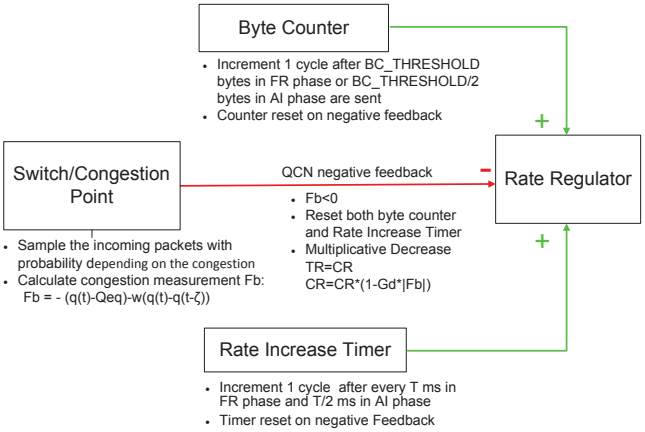
where R_{AI} is a constant chosen to be 5 Mbps in the baseline implementation.

3) RL is in the Hyper-Active Increase (HAI) phase if both BC and Rate Increase Timer are in AI. In this case, TR and CR are updated as:

$$\begin{aligned} TR &= TR + R_{HAI} * (min(BC\_cycle, Timer\_cycle) \\ &\quad - FR\_THRESHOLD) \\ CR &= \tfrac{1}{2}(CR + TR) \end{aligned}$$



(a) Sampling probability in QCN CP as a function of $|F_b|$



(b) QCN control mechanism

Fig. 9. QCN overview

where the constant R_{HAI} is set to 50 Mbps in the baseline implementation. So, the increment of TR in the HAI phase occurs in multiples of 50 Mbps. It is very important to note that RL goes to the HAI state only after at least $5BC\_THRESHOLD$ packets have been sent and $5T$ ms have passed since the last congestion feedback message was received. This doubly ensures that aggressive rate increases occur only after RL provides the network adequate opportunity for sending rate decrease signals should there be congestion. This is vital to ensure stability of the QCN algorithms while optimization can be performed to improve its responsiveness, for the sake of stability and simplicity.

In order to improve the fairness of multiple flows sharing the link capacity of QCN, an enhanced QCN congestion notification algorithm, called fair QCN (FQCN) [37], was proposed. The performance of FQCN was evaluated in [37] in terms of fairness and convergence. As compared to QCN, fairness is improved greatly and the queue length at the bottleneck link converges to the equilibrium queue length very fast.

D. Comparison and Discussion

Some characteristics of BCN, FECN, E-FECN, and QCN congestion notification algorithms are summarized in Table IV. All of them are concerned with provisioning congestion notification in DCNs. We will discuss and compare the pros and cons of these congestion notification algorithms in the following aspects.

TABLE IV
A COMPARISON OF BCN, FECN, E-FECN, AND QCN

Parameters	BCN [32], [33]	FECN [34]	E-FECN [35]	QCN [36]
Fairness	Unfair (better with drift)	Perfect	Perfect	Unfair
Feedback Control	Backward	Forward	Forward with beacon	Backward
Overhead	High (Unpredictable)	Low (predictable), 20bytes	Medium	Medium (unpredictable)
Rate of Convergence to Stability	Slow	Fast	Fast	Medium
Congestion Regulation	Fast	Slow	Medium	Fast
Throughput Oscillation	Large	Small	Small	Medium
Load Sensor	Queue based	Rate based	Rate + Queue based	Queue based
Link Disconnection	Support	N/A	Support	Support
Fast Start	Support	N/A	Support	Support
Number of Rate Regulators	Variable	Fix (= number of source flows)	Variable	Variable
Reactive and Proactive Signalling	Reactive	Proactive	Reactive & Proactive	Reactive

1) Fairness: Research work has shown that BCN achieves only proportional fairness but not max-min fairness [61]. With max-min fairness, the minimum data rate that a data flow achieves is maximized first; the second lowest data rate that a data flow achieves is maximized, and so on. Proportional fairness is a compromise between max-min fairness and maximum throughput scheduling algorithm. With proportional fairness, data rates are assigned inversely proportional to its anticipated resource consumption. The fairness in FECN/E-FECN is ensured by the congestion detection algorithm at the switch, which advertises the same rate to all the flows passing through the switch. Similar to BCN, the feedback message is only sent to the source of the sampled packet in QCN, and therefore QCN only achieves proportional fairness rather than max-min fairness.

2) Feedback Control: From the system models shown in Fig. 8, it is obvious that BCN and QCN use backward feedback control, FECN employs forward feedback, and the forward feedback control in E-FECN works together with BCN00 messages to inform the source with congestion at the switch.

3) Overhead: The overhead of BCN is high and unpredictable. The overhead of QCN is also unpredictable, but smaller than that of BCN since there is only negative QCN message to reduce the sending rate, and the sampling probability is proportional to the congestion indicator. The overhead of FECN is low and predictable because the FECN message is sent periodically with a small payload of about 20 bytes. The overhead of E-FECN is larger than that of FECN due to the BCN00 signal involved in the E-FECN algorithm.

4) Rate of Convergence to Fair State: BCN is slow in convergence to the fair state because AIMD-like algorithms can achieve fairness only in the long term sense. FECN and E-FECN can reach the perfect fair state within a few round trip times because all sources get the same feedback.

5) Congestion Regulation: The source rate in BCN and QCN can be reduced more quickly than that in FECN because the message in BCN and QCN is sent directly from the congestion point while the probe message in FECN has to take a round trip before it returns back to the source. The congestion adjustment speed is improved in E-FECN by using the BCN00 message under severe congestion.

6) Throughput Oscillation: BCN incurs large oscillations in throughput. FECN and E-FECN do not incur large oscillations in source throughput. The throughput oscillation is improved in QCN with the rate increase determined jointly by Byte Counter and Rate Increase Timer at the source.

7) Load Sensor: BCN and QCN send the queue dynamics back to the sources, while FECN uses a rate based load sensor to detect congestion. In addition to the rate-based sensors, queue monitor is also employed for severe congestion notification in E-FECN.

8) Link Disconnection: If a link is broken, BCN, E-FECN, and QCN can employ the reactive feedback message to inform the source in order to decrease or stop the packet transmission, while in FECN, there is no reactive feedback message and the probe might not return back to the source which will keep the sending rate, thus causing packet loss. To solve this problem, a probe timeout is introduced in FECN to let the sources lower their sending rates during the link disconnection.

9) Fast Start: In BCN and QCN, the sources are initialized with the full rate, and ramp down if a negative feedback is received from a switch. In FECN, the sources start at a low rate and move to the equilibrium rates as successive probes return.

10) Number of Rate Regulators: FECN requires as many regulators as the number of concurrent flows. Owing to the introduced BCN00 message under severe congestion, the number of rate regulators in E-FECN varies. In BCN and QCN, the feedback message is only sent to the source of the sampled packet, and therefore the number of rate regulators in BCN and QCN also varies.

11) Reactive and Proactive Signaling: BCN and QCN use reactive signaling, FECN deploys proactive signaling, and E-FECN employs both signalling methods. Since proactive probes are only sent periodically, at least one periodic interval is needed to respond to the sudden overload, which may cause the network to be severely congested. Reactive signaling with feedback can help reduce congestion with sudden change of the link capacity or traffic patterns. However, the problem of

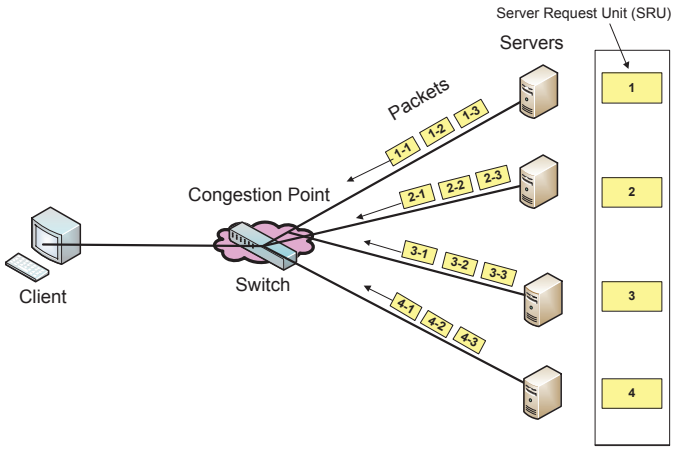


Fig. 10. A simple and representative TCP Incast network setting with one client requesting data from multiple servers through synchronized reads.

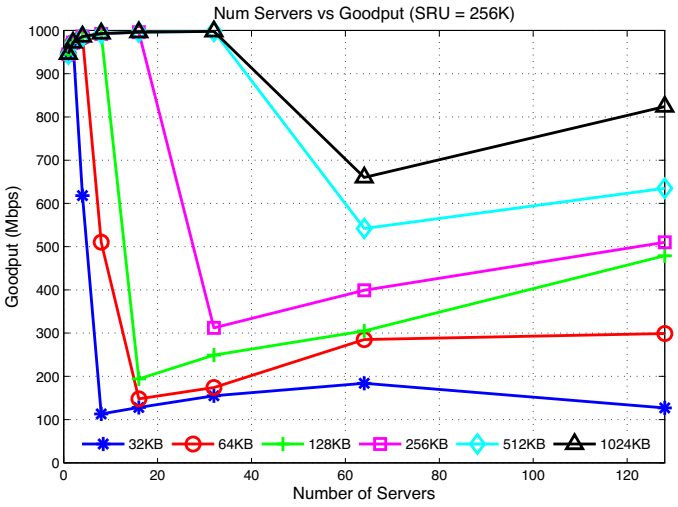


Fig. 11. TCP throughput collapse for a synchronized read application performed on the network as shown in Fig. 10 with different buffer sizes.

reactive signalling is that the rates of some flows are reduced too much that they may not recover.

V. TCP Incast

TCP Incast, where TCP throughput drastically reduces when multiple sending servers communicate with a single receiver separated by one or more Ethernet switches or routers in high bandwidth (1-10 Gbps), low latency (round trip time of tens to hundreds of microseconds) networks using TCP, potentially arises in many datacenter applications, e.g., in cluster storage [1], when storage servers concurrently respond to requests for a data block, in web search, when many workers respond near simultaneously to search queries, and in batch processing jobs like MapReduce [3], in which intermediate key-value pairs from many Mappers are transferred to appropriate Reducers during the "shuffle" stage. In this section, we will present a brief overview of TCP Incast network setting pattern and earlier proposed solutions to mitigate TCP Incast.

A. Simplified TCP Incast network settings

A simple and basic representative network setting in which TCP Incast can occur is shown in Fig. 10. Data is stripped over a number of servers, and stored as a Server Request Unit (SRU) on each server. In order to access one particular data block, a client needs to perform synchronized readings: sending request packets to all of the storage servers containing a fragment of data block for this particular block. The client will not generate data block requests until it has received all the data for the current block. Upon receiving the requests, the servers transmit the data to the receiver through one Ethernet switch almost concurrently. Small Ethernet buffers may be exhausted by the concurrent flood of traffic, thus resulting in packet loss and TCP timeouts. Therefore, TCP Incast may be observed during synchronized readings for data blocks across an increasing number of servers. Fig. 11 illustrates this performance drop in a cluster-based storage network environment when a client requests data from several servers with different buffer size. It shows that increasing the buffer size can delay the onset of TCP Incast.

Studies in [38]–[40] revealed that TCP timeout is the primary cause of TCP Incast. The synchronized reading or writing data flows can overload small switch buffers, thus causing packet drops and leading to TCP timeouts which last hundreds of milliseconds. Thus, TCP throughput is reduced drastically. Phanishayee *et.al* [38] analyzed the TCP traces obtained from TCP Incast simulations, suggesting that TCP retransmission timeouts are the main root cause of TCP Incast. As reported in [38], packet losses occurred during one block request result in a link idle duration of 200 ms, which is the TCP retransmission timeout (RTO) time. References [39], [40] further confirmed this root cause analysis on TCP Incast by investigating the effects of reducing TCP RTO parameter from the default 200 ms to sub-milliseconds to alleviate the TCP Incast problem. The inciting of TCP Incast can be summaried by the following conditions:

- high-bandwidth, low-latency networks connected by Ethernet switches with small buffers;
- clients issuing barrier-synchronized requests in parallel;
- servers responding with a fragment of data block per request.

B. Current Approaches to Mitigate TCP Incast

Incast has not been thoroughly studied, but several methods have been proposed to avoid TCP Incast.

1) Limiting the Number of Servers or Limiting Data Transfer Rate: Current systems attempt to avoid TCP Incast onset by limiting the number of servers involved in data block transfer, or by limiting the data transfer rate. These solutions, however, are typically specific to one configuration, and thus are not robust to configuration changes in DCNs.

2) Reducing Losses with Larger Switch Buffers: Increasing the Ethernet switch buffer size can delay the onset of TCP Incast. Phanishayee *et al.* [38] stated that doubling the Ethernet switch buffer size doubles the number of concurrent sending servers before experiencing TCP Incast. However, switches and routers with large buffers are expensive, and even large buffers may be exhausted quickly with even higher speed links.

3) Reducing Idle Link Time by Increasing SRU Size: Increasing the SRU size can also delay the onset of TCP Incast. However, most applications ask for data in small chunks, typically in the range of 1-256 KB. In addition, a larger SRU size can increase lock content due to overlapping writes, thus causing poor write performance in file system applications.

4) TCP Variants and TCP Slow Start: Phanishayee *et al.* [38] evaluated several TCP variants, including TCP Reno, New Reno, and SACK. None of them can help solve the TCP Incast problem. They also found that eliminating TCP slow start did not help either.

5) Ethernet Flow Control: Ethernet flow control [62] is effective for a single switch topology, but breaks down in multi-switch or multi-layered switching topologies due to head-of-line blocking.

6) Congestion Control: The performance of QCN with respect to the TCP incast problem during data access from clustered servers in datacenters has been investigated in [37]. QCN can effectively control link rates very rapidly in a datacenter environment. However, it performs poorly when TCP Incast is observed. As compared to QCN, FQCN significantly enhances TCP throughput due to the fairness improvement.

7) Application Level Solution: Application level solutions, such as global request scheduling, are possible, but they require potentially complex modifications to many TCP applications.

8) Reducing TCP minimum RTO: Reducing the minimum value of the retransmission timeout (RTO) from the default 200 ms to 200 μs significantly alleviates the problem as claimed in [39]. However, as the authors pointed out, most systems lack the high-resolution timers required for such low RTO values.

9) Fine-Grained TCP Retransmission: Fine-grained timers to facilitate sub-millisecond RTO timer values, adding randomness to RTO, and disabling TCP delayed ACKs were proposed recently [40]. High resolution timers are introduced to support fine-grained RTT measurements with the granularity of hundreds of microseconds. The results in [40] show that enabling microsecond RTO values in TCP is effective in avoiding TCP incast collapse. It also shows that by adding an adaptive randomized RTO component to the scheduled timeout, the throughput does not experience collapse even with a large number of concurrent sources due to the flow retransmission de-synchronization. The TCP delayed ACK mechanism attempts to reduce the amount of ACK traffic; the results in [40] show that coarse-grained delayed ACKs should be avoided when possible in DCNs, and most high-performance applications in the datacenter favor quick response over additional ACK-processing overhead. The results in [39], however, contradict this point; the low resolution RTO timer of 1 ms with delayed ACKs is actually optimal. The authors further showed that with delayed ACKs turned off, the congestion window exhibits larger fluctuations and a more severe throughput collapse.

10) Small Buffers: Packet buffers in switches are expensive, and a recent work [63] also explored the possibility of using small buffers in equipment to still achieve near full link utilization.

VI. GREEN DCN

The high operational cost and the mismatch between data center utilization and power consumption have spurred interest in improving data center energy efficiency. The energy saving opportunities for data centers [64] have been studied to gain a solid understanding of data center power consumption. IT infrastructure, including servers, storage, and communication equipments, as well as power distribution infrastructure and cooling infrastructure are the three main contributors to power consumption of data centers. Based on a benchmark study of data center power consumption [64], the total power consumption of IT equipments accounts for approximate 40%-60% power delivered to a data center, and the rest is utilized by power distribution, cooling, and lighting that support IT infrastructure. Figure 12 shows the power usage of a typical data center [65]. In order to quantify the energy efficiency of data centers, several energy efficiency metrics have been proposed. The most commonly used metric to indicate the energy efficiency of a data center is power usage effectiveness (PUE), which is defined as the ratio of the total energy consumption of data center to the total energy consumption of IT equipment. The PUE metric measures the total power consumption overhead caused by the data center facility support equipment, including the cooling systems, power delivery, and other facility infrastructure like lighting. Working on improving energy efficiency of DCNs, Google publishes quarterly the PUE results from data centers with an IT load of at least 5MW and time-in-operation of at least 6 months [66]. The latest trailing twelve-month, energy-weighted average PUE result obtained in the first quarter of 2011 is 1.16, which exceeds the EPA's goal for state-of-the-art data center efficiency.

Energy savings in DCNs can come from more energy efficient hardware, servers and networking equipment. In contrast, tremendous efforts have been made to address power efficiency in DCNs with power management in server and storage clusters. Improved DCN architecture can also save energy. Moreover, some smart power delivery and cooling technologies have been investigated and verified to be effective solutions to save energy. With servers becoming more energy proportional, the network power cannot be ignored. In addition, "green" energy supply is becoming popular for datacenters as demonstrated by many small- and medium-size datacenters, which are powered partially or completely by solar and/or wind energy. In this section, energy-saving approaches for DCNs are reviewed and discussed.

A. Energy Efficient Server

According to the research by Greenberg *et al.* [11], 45 percent of the total energy consumption goes to servers in DCNs. Two reasons can be identified to cause this high fraction of server energy consumption: low server utilization and a lack of power-proportionality. At low levels of workload, servers are highly energy-inefficient. As shown in [45], [67], the power consumption of current commodity servers can be approximated by a model with a large always-present constant power and a dynamic power linear to server performance. The amount of dynamic power is small, at most

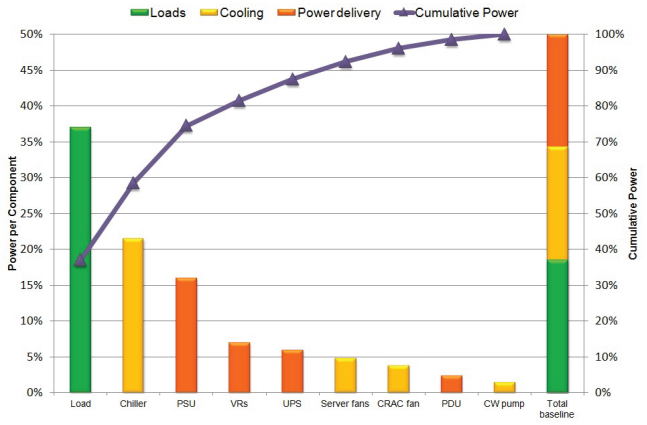


Fig. 12. A typical data center power usage (adapted from [65]).

25% of the total dissipated power. At the idle state, the power consumed is over 50% of its peak power for an energy-efficient server [44] and often over 80% for a commodity server [45]. High power waste at low workload has prompted a fundamental redesign of each computer system component to exemplify the energy-proportional concept, especially on processors since the processor is the most power consuming component before (processor contributes to over 55% of the total server power in 2005 [44]). Many research works have been done to explore processor designs to reduce CPU power with Dynamic Voltage/Frequency Scale (DVFS) [68], [69]. DVFS can be deployed to save energy at the cost of slower program execution by reducing the voltage and frequency. The effectiveness of DVFS in saving energy with moderately intense web workloads was examined in [43], and the results show that DVFS can save from 23% to 36% of the CPU energy while keeping server responsiveness within reasonable limits. Unfortunately, processors no longer dominate power consumption in modern servers. Processors currently contribute around 25% of the total system power consumption [44]. According to a study shown in [45], the chipset is the dominant constant power consumption in modern commodity servers.

Several techniques can be used to reduce power consumption of memory and disk subsystems. A novel, system-level power management technique, power shifting, for increasing performance under constrained power budgets was proposed to re-budget the available power between processor and memory to maintain a server budget [70]. Power shifting is a threshold-based throttling scheme to limit the number of operations performed by each subsystem during an interval of time, but power budget violations and unnecessary performance degradation may be caused by improper interval length. Bruno *et al.* [71] proposed and evaluated four techniques, called Knapsack, LRUGreedy, LRU-Smooth, and LRU-Ordered, to dynamically limit memory power consumption by adjusting the power states of the memory devices, as a function of the load on the memory subsystem. They further proposed energy and performance aware version of these techniques to trade off between energy consumption and performance. Bruno *et al.* [72] proposed Mini-rank, an adaptive DRAM architecture, to limit power consumption of DRAM by breaking a conventional DRAM rank into multiple smaller mini-ranks with a small bridge chip. Dynamic Rotations per Minute (DRPM) [73] was proposed as a low-level hardware-based technique to dynamically modulate disk speed to save power in disk drives since the lower the disk drive spins the less power it consumes.

In order to design an energy-proportional computer system, each system component needs to consume energy in proportion to utilization. However, many components incur fixed power overheads when active like clock power on synchronous memory busses, and thus designing an energy-proportional computer system still remains a research challenge. Several system level power management schemes have been proposed [74], [75] to reduce power consumption by putting idle servers to sleep. Given the state of the art of energy efficiency of today's hardware, energy proportional systems can be approximated with off-the-shelf non-energy-proportional hardware at the ensemble layer through dynamic virtual machine consolidation [74]. It is also believed that new alternative energy-efficient hardware designs will help design energy proportional systems at both the single server and ensemble layer. However, this method works at the coarse time scale (minutes) and cannot address the performance isolation concerns of dynamic consolidation. An energy-conservation approach, called PowerNap [75], was proposed to attune to server utilization patterns. With PowerNap, the entire system transits rapidly between a high-performance active state and a minimal-power nap state in response to instantaneous load. Rather than requiring fine-grained power-performance states and complex load proportional operation from each system component, PowerNap minimizes idle power and transition time. However, a transition time of under $10ms$ is required for significant power savings; unfortunately, sleep times on current servers are two orders of magnitude larger.

B. Server/Storage Clusters

Besides power proportional servers, tremendous efforts have been made to achieve power proportional server/storage clusters. Chen *et al.* [76] proposed to reduce power usage in the data center by adopting dynamic server provisioning techniques, which are effective techniques to dynamically turn on a minimum number of servers required to satisfy application specific quality of service and load dispatching that distributes current load among the running machines. This work specially deals with long-lived connection-intensive Internet services. Several recent works utilized machine learning to dynamically provisioning virtual machines while maintaining quality of service [77], [78]. However, virtual machine migration takes several minutes and virtual machines introduce performance overheads. A power-proportional cluster [79], consisting of a power-aware cluster manager and a set of heterogeneous machines, was designed to make use of currently available energy-efficient hardware, mechanisms for transitioning in and out of low-power sleep states, and dynamic provisioning and scheduling to minimize power consumption. This work specially deals with short lived request-response type of workloads. Based on queuing theory and service differentiation, an energy proportional model was proposed for server clusters in [80], which can achieve theoretically guaranteed service performance with accurate, controllable and predictable quantitative control over power consumption. The authors also

analyzed the effect of transition overhead to reduce the impact of performance degradation.

Significant opportunities of power savings exist at the application layer, which has the most information on performance degradation and energy tradeoff. As verified in [81], consolidation of applications in cloud computing environments can present a significant opportunity for energy optimization. Their study reveals the energy performance trade-offs for consolidation and shows that optimal operating points exist and the application consolidation problem can be modeled as a modified bin-packing problem. Kansal *et al.* [82] enabled generic application-layer energy optimization, which guides the design choices using energy profiles of various resource components of an application.

Several energy saving techniques can be applied to energy proportional storage systems. The first set of techniques uses intelligent data placement and/or data migration to save energy. Hibernator [83], a disk array energy management system, uses several techniques to reduce power consumption while maintaining performance goals, including disk drives that rotate at different speeds and migration of data to an appropriate-speed disk drive. Sample-Replicate-Consolidate Mapping (SRCMap) [84], a storage virtualization solution for energy-proportional storage, was proposed by consolidating the cumulative workload on a minimal subset of physical volumes proportional to the I/O workload intensity. Some other techniques benefit from inactive low energy modes in disks. A log-structured file system solution was proposed in [85] to reduce disk array energy consumption by powering off a fraction of disks without incurring unacceptable performance penalties because a stripping system could perfectly predict disks for write-access. Based on the observation of significant diurnal variation of data center traffic, Sierra [86], a power-proportional, distributed storage system, was proposed to turn off a fraction of storage servers during trough traffic period. Sierra utilizes a set of techniques including power-aware layout, predictive gear scheduling, and a replicated short-term store, to maintain the consistency, fault-tolerance of the data, as well as good system performance. A power-proportional distributed file system, called Rabbit, was proposed in [87] to provide ideal power-proportionality for large-scale cluster-based storage and data-intensive computing systems by using a new cluster-based storage data layout. Rabbit can maintain near ideal power proportionality even with node failures.

C. Green Network Equipment

A power measurement study of a variety of networking gear, e.g., switches, routers, wireless access points, was performed in [88] to quantify power saving schemes. A typical networking router power consumption can be divided among four main components: chassis, switching fabric, line cards, and ports. The chassis alone consumes 25% to 50% of the total router power in a typical configuration [89]. Furthermore, power characteristics of current routers are not energy-proportional; even worse, they consume around 90% of their maximum power consumption [89]. Prompted by the poor energy characteristics of modern routers, active research is being conducted in reducing power consumption of networking equipment, and researchers have suggested a few techniques to save energy [90], [91], e.g., sleeping and rate-adaption. With a low power "sleep" mode, network equipment can stay in two states, a sleeping state and an active state. Network equipment transits into the low-power "sleep" mode when no transmission is needed, and returns back to the active mode when transmission is requested. However, the transition time overhead putting device in and out the sleep mode may reduce energy efficiency significantly [92], and so Reviriego *et al.* [93] explored the use of burst transmission to improve energy efficiency. Another technique to save power is to adapt the transmission rate of network operation to the offered workload [91], based on the fact that the lower the line speed is, the less power the devices consume. Speed negotiation is required in the rate-adaption scheme for two transmission ends. Speed negotiation requires from a few hundred milliseconds to a few seconds; this is excessive for many applications. The effectiveness of power management schemes to reduce energy consumption of networks based on sleeping and rate adaption was evaluated in [94]. It was shown that sleeping or rate-adaptation can offer substantial savings.

Several schemes have been proposed to reduce power consumption of network switches in [95], including Time Window Prediction (TWP), Power Save Mode (PSM), and Lightweight Alternative. The theme of these schemes is to trade off some performance, latency and packet-loss, to reduce power consumption. TWP and PSM schemes concentrate on intelligently putting ports to sleep during idle periods. In TWP, switches monitor the number of packets crossing a port in a sliding time window and predict the traffic in the next sliding window. If the number of packets in the current sliding time window is below a pre-defined threshold, the switch powers off the port for some time. Packets that arrive at the port in the low power state are buffered. Adaptive sleep time is used in TWP based on the prediction of the traffic for the next sliding window and latency requirements. The performance of TWP relies on the accuracy of the prediction function. The more accurate the prediction function, the less latency it may cause. PSM is a special case of the TWP where the sleep happens with regularity instead of depending on the traffic flow. PSM is more of a policy-based scheme to power off ports and naturally causes more latency than TWP does. By observing a clear diurnal variation in the traffic patterns, lightweight alternative switches have been proposed. Lightweight alternative switches are low-power integrated switches with lower packet processing speed and line speeds. At low traffic load, only the lightweight alternative switches are powered on to provide the connection. An analytical framework, which can effectively be adopted to dynamically optimize power consumption of a network device while maintaining an expected forwarding performance level, was proposed in [96].

D. Networking

With servers becoming more energy proportional, the network power cannot be ignored. In a recent work, a network-wide power manager, ElasticTree [97], was proposed to dynamically adjust the set of active network elements, links and switches, to satisfy changing data center traffic loads. ElasticTree optimizes a DCN by finding the power-optimal

network subset to route the traffic. It essentially extends the idea of power proportionality into the network domain, as described in [44]. Chabarek *et al.* [89] used mixed integer programming to optimize router power in a wide area network, by choosing the chassis and line card configuration to best meet the expected demand. Mahadeva *et al.* [98] proposed a network wide energy monitoring tool, Urja, which can be integrated into network management operations to collect configuration and traffic information from live network switches and accurately predict their power consumption. By analyzing real network traces, several techniques, including disabling unused ports, port rate adaptation, maximizing active ports on a line card and using fewer switches, were proposed to near-proportional power consumption with non power proportional devices.

In [99], three schemes, namely, link state adaption (LSA), network traffic consolidation (NTC), and server load consolidation (SLC), were proposed to reduce the energy consumed by networking equipment in any network with a single administrative control domain. In LSA, the power control uses traffic information on each link to adapt its state accordingly. Typically, each link can operate at the disabled state when there is no traffic on the link, or operate at a different link rate based on the traffic load. In NTC, traffic is routed and consolidated on fewer links and switches, while some of the non-utilized links and switches can be disabled. This approach reduces energy consumption significantly by removing all redundancy in the networks. The energy consumed is the minimum required to support the offered network load, but it comes at a great cost to reliability as there are no redundant paths in the topology. The SLC scheme migrates jobs, and so fewer number of servers are being used. In the SLC scheme, server resources such as CPU and memory should be adequate to handle the assigned jobs. The energy savings of all of these three schemes come at the cost of availability and reliability. To mitigate the performance degradation, one constraint is applied to ensure that each link's utilization never exceeds a certain threshold before adapting its rate. The authors suggested to incorporate service level awareness by adding constraints to ensure a minimum performance.

The energy-saving problem in DCNs was solved from a routing perspective in [100]. They established the model of energy-aware routing, in which the objective is to find a route for a given traffic matrix that minimizes the total number of switches. They proved that the proposed energy-aware routing model is NP-hard, and designed a heuristic algorithm to solve the energy-aware routing problem. Vasić *et al.* [101] proposed Energy-Proportional Networks (EPNs), which use the minimum amount of energy to carry the required traffic. In EPN, three sets of routing tables, alwayson, on-demand and failover, are calculated. It uses an energy-aware traffic engineering algorithm to activate/de-activate network elements to achieve the goal of energy-proportionality.

E. Network Architecture

Intel Research proposed and evaluated the proxy architecture which uses a minimal set of servers to support different forms of idle-time behavior for saving energy [102]. Abts *et al.* [16] proposed a new high-performance DCN architecture, called energy proportional datacenter networks and showed that a flattened butterfly DCN topology is inherently more power efficient than the other commonly proposed topology for high-performance DCNs. GreenCloud [103] enables comprehensive online monitoring, live virtual machine migration, and virtual machine placement optimization to reduce DCN power consumption while guaranteeing the performance goals.

F. Smart Power Delivery and Cooling

Some smart power delivery and cooling technologies have been investigated and verified to be effective in saving energy. Early works on power delivery and cooling techniques for DCNs focused on computational fluid dynamic models to analyze and design server racks and data center configurations to optimize the delivery of cold air to minimize cooling costs [104], [105]. Distributing the workload onto those servers that are more efficient to cool than others was proposed and verified to reduce power consumption of data centers [106], [107]. Parolini *et al.* [108] proposed a coordinated cooling and load management strategy to reduce data center energy consumption, and formulated the energy management problem as a constrained Markov decision process, which can be solved by linear programming to generate the optimal strategies for power management. Vasic *et al.* [109] proposed a thermodynamic model of a data center and designed a novel model-based temperature control strategy, by combining air flow control for long time-scale decisions and thermal-aware scheduling for short time-scale workload fluctuations. Load distribution in a thermal-aware manner incurs the development of fast and accurate estimation techniques for temperature distribution of a data center [110], [111]. Several proposals have been made to reduce power consumption in data centers by keeping as many servers as necessarily active and putting the rest into the low-power sleep mode. However, these proposals may cause hot spots in data centers, thus increasing the cooling power and degrading response time due to sleep-to-active transition delays. In order to solve these problems, PowerTrade [112], a novel joint optimization of idle power and cooling power, was proposed to reduce the total power of data centers. Furthermore, Ahmad and Vijaykumar [112] proposed a two-tier scheme, SurgeGuard, to address response time degradation by using over-provisioning at coarse time granularity to absorb common and smooth load increases, and to provide a fine-grain replenishment of the over-provisioned reserves at fine time granularity to handle abrupt loading surges.

G. Green Energy Supply

Besides the strategies discussed above to reduce the power consumption in data centers, "green" energy has been partially or completely initiated for data center power supply [113], [114]. Solar and wind energy are the most promising clean energy technologies as they do not create big issues such as environmental problems caused by hydroelectric energy and the waste storage problem of nuclear energy. How to maximize the usage of green energy across multiple datacenters have been discussed in [115]–[118]. In [115], a framework for request distribution policies based on time zones, variable

electricity prices, and green energy has been investigated for multi-datacenter Internet services. The evaluation results demonstrated that datacenters powered with "green" energy can reduce "brown" energy consumption, which is produced by carbon-intensive means such as coal-fired power plants, with a small cost increase. Another request distribution policy based on capping the "brown" energy consumption was proposed in [116]. Stewart and Shen [117] discussed renewable energy management across multiple datacenters for cloud-computing Internet services. They showed some promising preliminary results for request-level power/energy profiling, with which the energy consumption type of the request was considered in the workload distribution. Thus, more energy-hungry requests should be forwarded to datacenters with excess green energy. Similar observations have been reported in [118] that geographical load balancing can encourage the use of "green" energy and reduce the use of "brown" energy. "Green" energy management has also been explored within a single datacenter. Goiri *et al*. [119] proposed GreenSlot, a parallel batch job scheduler for a single DCN partially powered by solar energy, to maximize the usage of green energy in a datacenter based on the availability prediction of the solar energy.

VII. Conclusion

In this survey, we have reviewed most recent research activities in DCNs, with a specific emphasis on the topics of network architecture design, congestion notification, TCP Incast, and power consumption. We have introduced the tree-based hierarchical topology of conventional DCNs, detailed the major problems of tree-based hierarchical topology, and presented requirements for new DCN architectures. Some of the recently proposed architectures have been discussed and compared. In addition, we have summarized and compared the pros and cons of the major congestion notification algorithms proposed for DCNs. TCP Incast potentially arises in many datacenter applications. We have presented a brief overview of the TCP Incast problem and previously proposed solutions to mitigate TCP Incast. Furthermore, the energy cost of servers and DCNs have spurred great interest in improving data center energy efficiency, aiming to build energy proportional DCNs. We have reviewed the solutions to build energy efficient data centers from various aspects including energy-efficient servers, networking equipment, power management in server and storage clusters, smart power delivery and cooling, energy-efficient network architecture design, and network-wide power management. Readers are referred to the respective references on some other research topics on DCNs, not covered in this survey, such as traffic characteristics [14], [50], [120], [121], TCP protocol specific suitable for high-bandwidth low-latency DCNs [122], routing [9], forwarding and switching algorithms [123]–[126], flow management/scheduling [127], [128], load balancing [129], and data center modeling and simulation [130]–[136].

References

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 29–43, 2003.

[2] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A Distributed Storage System for Structured Data," *ACM Trans. Computer Systems*, vol. 26, no. 2, pp. 1–26, 2008.

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[4] T. Hoff, "Google Architecture," http://highscalability.com/google-architecture, Nov. 2008.

[5] F. Machida, D. S. Kim, J. S. Park, and K. Trivedi, "Toward optimal virtual machine placement and rejuvenation scheduling in a virtualized data center," in *Proc. IEEE International Conference on Software Reliability Engineering Workshops*, Nov. 2008, pp. 1 –3.

[6] M. Mishra and A. Sahoo, "On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach," in *Proc. IEEE International Conference on Cloud Computing*, July 2011, pp. 275 –282.

[7] H. Liu, H. Jin, X. Liao, C. Yu, and C. Xu, "Live virtual machine migration via asynchronous replication and state synchronization," *IEEE Trans. Parallel Distrib. Syst.*,, vol. PP, no. 99, March 2011.

[8] V. Shrivastava, P. Zerfos, K. won Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers," in *Proc. IEEE International Conference on Computer Communications*, Shanghai, China, April 10-15, 2011, pp. 66–70.

[9] K. Chen, C. Hu, X. Zhang, K. Zheng, Y. Chen, and A. Vasilakos, "Survey on routing in data centers: insights and future directions," *IEEE Network*, vol. 25, no. 4, pp. 6 –10, July-August 2011.

[10] Cisco Data Center Infrastructure 2.5 Design Guide, http://www.cisco.com/application/pdf/en/us/guest/netsol/ns107/c649/ccmi\newlinegration\underline\space09186a008073377d.pdf.

[11] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The Cost of a Cloud: Research Problems in Data Center Networks," *Sigcomm Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2009.

[12] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *Proc. ACM SIGCOMM*, Seattle, WA, USA, August 17-22, 2008, pp. 63–74.

[13] A. Greenberg, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Towards a Next Generation Data Center Architecture: Scalability and Commoditization," in *Proc. ACM workshop on Programmable Routers for Extensible Services of Tomorrow*, Seattle, WA, USA, August 22, 2008, pp. 57–62.

[14] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: a Scalable and Flexible Data Center Network," in *Proc. ACM SIGCOMM*, Barcelona, Spain, August 17-21, 2009, pp. 51–62.

[15] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "PortLand: a Scalable Fault-Tolerant Layer 2 Data Center Network Fabric," in *Proc. ACM SIGCOMM*, Barcelona, Spain, August 17-21, 2009, pp. 39–50.

[16] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy Proportional Datacenter Networks," in *Proc. 5th international conference on Emerging Networking Experiments and Technologies*, Saint-Malo, France, June 19-23, 2010, pp. 338–347.

[17] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul, "SPAIN: COTS Data-Center Ethernet for Multipathing over Arbitrary Topologies," in *Proc. 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI '10)*, San Jose, CA, April 2010.

[18] J. Hamilton, "An Architecture for Modular Data Centers?" in *Proc. of the 3rd Biennial Conference on Innovative Data Systems Research (CIDR '07)*, Asilomar, California, USA, January 7-10, 2007, pp. 3–10.

[19] K. V. Vishwanath, A. Greenberg, and D. A. Reed, "Modular Data Centers: How to design Them?" in *Proc. 1st ACM workshop on Large-Scale System and Application Performance, (LSAP '09)*, Garching, Germany, 2009, pp. 3–10.

[20] IBM, http://www-935.ibm.com/services/us/its/pdf/smdc-eb-sfe03001\newline-usen-00-022708.pdf.

[21] R. Systems, http://www.rackable.com/products/icecube.aspx.

[22] V. Systems, http://www.cirrascale.com/forest_specifications.asp.

[23] M. Mitchell Waldrop, "Data Center in a Box," *Scientific American*, Aug. 2007.

[24] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "DCell: a Scalable and Fault-Tolerant Network Structure for Data Centers," in *Proc. ACM SIGCOMM*, Seattle, WA, August 17-22, 2008, pp. 75–86.

[25] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: a High Performance, Server-Centric Network Architecture for Modular Data Centers," in *Proc. ACM SIGCOMM*, Barcelona, Spain, August 17-21, 2009, pp. 63–74.

[26] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "MDCube: a High Performance Network Structure for Modular Data Center Interconnection," in *Proc. 5th international conference on Emerging Networking Experiments and Technologies*, Rome, Italy, Dec. 2009, pp. 25–36.

[27] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, and S. Lu, "FiConn: Using Backup Port for Server Interconnection in Data Centers," in *Proc. IEEE International Conference on Computer Communications*, April 19-25, 2009, pp. 2276 –2285.

[28] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, S. Lu, and J. Wu, "Scalable and Cost-Effective Interconnection of Data-Center Servers Using Dual Server Ports," *IEEE/ACM Trans. Netw.*, vol. PP, no. 99, 2010.

[29] Y. Liao, D. Yin, and L. Gao, "DPillar: Scalable Dual-Port Server Interconnection for Data Center Networks," in *Proc. 19th International Conference on Computer Communications and Networks*, Zurich, Switzerland, Aug. 2-5, 2010.

[30] P. Costa, T. Zahn, A. Rowstron, G. O'Shea, and S. Schubert, "Why Should We Integrate Services, Servers, and Networking in a Data Center?" in *Proc. 1st ACM workshop on Research on Enterprise Networking*, Barcelona, Spain, Aug. 21, 2009, pp. 111–118.

[31] P. Costa, A. Donnelly, G. O'Shea, and A. Rowstron, "CamCube: A Key-based Data Center," Microsoft Research, Tech. Rep. MSR TR-2010-74, 2010.

[32] D. Bergamasco and R. Pan, "Backward Congestion Notification Version 2.0," in *IEEE 802.1 Meeting*, September 2005.

[33] D. Bergamasco, "Ethernet Congestion Manager," in *IEEE 802.1Qau Meeting*, March 13 2007.

[34] J. Jiang, R. Jain and C. So-In, "An Explicit Rate Control Framework for Lossless Etherent Operation," in *Proc. ICC*, Beijing, China, May 19-23, 2008, pp. 5914 – 5918.

[35] C. So-In, R. Jain and J. Jiang, "Enhanced Forward Explicit Congestion Notification (E-FECN) Scheme for Datacenter Ethernet Networks," in *Proc. Symposium on Performance Evaluation of Computer and Telecommunication Systems*, Edinburgh, UK, Jun. 2008, pp. 542 – 546.

[36] M. Alizadeh, B. Atikoglu, A. Kabbani, A. Lakshmikantha, R. Pan, B. Prabhakar, and M. Seaman, "Data Center Transport Mechanisms: Congestion Control Theory and IEEE Standardization," in *Proc. 46th Annual Allerton Conference on Communication, Control, and Computing*, Allerton House, UIUC, Illinois, Sep. 2008, pp. 1270–1277.

[37] Y. Zhang and N. Ansari, "On Mitigating TCP Incast in Data Center Networks," in *Proc. IEEE International Conference on Computer Communications*, Shanghai, China, Apr. 10-15, 2011.

[38] A. Phanishayee, E. Krevat, V. Vasudevan, D. G. Andersen, G. R. Ganger, G. A. Gibson, and S. Seshan, "Measurement and Analysis of TCP Throughput Collapse in Cluster-based Storage Systems," in *Proc. USENIX Conference on File and Storage Technologies*, San Jose, Feb. 2008, pp. 1–14.

[39] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph, "Understanding TCP Incast Throughput Collapse in Datacenter Networks," in *Proc. ACM workshop on Research on Enterprise Networking*, Barcelona, Spain, August 21, 2009, pp. 73–82.

[40] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller, "Safe and Effective Fine-grained TCP Retransmissions for Datacenter Communication," in *Proc. ACM SIGCOMM*, Barcelona, Spain, August 21, 2009, pp. 303–314.

[41] D. Nagle, D. Serenyi, and A. Matthews, "The Panasas ActiveScale Storage Cluster: Delivering Scalable High Bandwidth Storage," in *Proc. conference on Supercomputing*, Washington, 2004, pp. 53–53.

[42] U.S. Environmental Protection Agency, "Environmental Protection Agency, Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431," *ENERGY STAR Program*, Aug. 2007.

[43] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony, "The Case for Power Management in Web Servers," *Power Aware Computing*, pp. 261–289, 2002.

[44] Luiz André Barroso and Urs Hölzle, "The Case for Energy-Proportional Computing," *Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[45] S. Dawson-Haggerty, A. Krioukov, and D. Culler, "Power Optimization - a Reality Check," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-140, Oct. 2009.

[46] A. Bianzino, C. Chaudet, D. Rossi, and J. Rougier, "A Survey of Green Networking Research," *IEEE Commun. Surveys Tutorials*, vol. PP, no. 99, pp. 1 –18, 2010.

[47] Z. Hasan, H. Boostanimehr, and V. K. Bhargava, "Green Cellular Networks: A Survey, Some Research Issues and Challenges," *IEEE Commun. Surveys Tutorials*, vol. 13, no. 4, pp. 524 –540, 2011.

[48] Y. Zhang, P. Chowdhury, M. Tornatore, and B. Mukherjee, "Energy Efficiency in Telecom Optical Networks," *IEEE Commun. Surveys Tutorials*, vol. 12, no. 4, pp. 441 –458, 2010.

[49] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Network Infrastructures," *IEEE Commun. Surveys Tutorials*, vol. 13, no. 2, pp. 223 –244, 2011.

[50] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding Data Center Traffic Characteristics," in *Proc. ACM workshop on Research on Enterprise Networking*, Barcelona, Spain, Aug. 2009, pp. 65–72.

[51] C. Clos, "A Study of Non-Blocking Switching Networks," *The Bell System Technical Journal*, vol. 32, no. 2, pp. 406–424, March 1953.

[52] C. E. Leiserson, "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing," *IEEE Trans. Computer*, vol. 34, no. 10, pp. 892–901, Oct. 1985.

[53] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Efficient and Robust Routing of Highly Variable Traffic," in *Proc. 3rd Workshop on Hot Topics in Networks*, San Diego, CA, USA, November 15-16, 2004.

[54] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web," in *Proc. 29th annual symposium on Theory of Computing*, El Paso, USA, 1997, pp. 654–663.

[55] J. Kim, W. J. Dally, and D. Abts, "Flattened Butterfly: a Cost-Efficient Topology for High-Radix Networks," in *Proc. 34th annual Symposium on Computer Architecture*, May 2007, pp. 126–137.

[56] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures:Arrays, Trees, Hypercubes*. Morgan Kaufmann, 1992.

[57] B. Parhami, *Introduction to Parallel Processing: Algorithms and Architectures*. Kluwer Academic, 2002.

[58] D. Loguinov, J. Casas, and X. Wang, "Graph-Theoretic Analysis of Structured Peer-to-Peer Systems: Routing Distances and Fault Resilience," *IEEE/ACM Transactions on Networking*, vol. 13, no. 5, pp. 1107 – 1120, Oct. 2005.

[59] "Address resolution for massive numbers of hosts in the data center (armd)," http://datatracker.ietf.org/wg/armd/.

[60] "Data Center Bridging Task Group," http://www.ieee802.org/1/pages/dcbridges.html.

[61] J. Jiang and R. Jain, "Analysis of Backward Congestion Notification (BCN) for Ethernet In Datacenter Applications," in *Proc. IEEE International Conference on Computer Communications*, Anchorage, Alaska, USA, May 6-12, 2007, Anchorage, Alaska, USA, pp. 2456 – 2460.

[62] "Ethernet Flow Control," http://en.wikipedia.org/wiki/Ethernet_flow_control.

[63] A. Vishwanath, V. Sivaraman, and M. Thottan, "Perspectives on Router Buffer Sizing: Recent Results and Open Problems," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 2, pp. 34–39, 2009.

[64] N. Rasmussen, "Apc white paper #113, electrical efficiency modeling for data centers," http://www.apcmedia.com/salestools/SADE-5TNRLG_R5_EN.pdf, 2007.

[65] M. Ton, B. Fortenbery, and W. Tschudi, "DC Power for Improved Data Center Efficiency," http://hightech.lbl.gov/documents/DATA_CENTERS/DCDemoFinalReport.pdf, March 2008.

[66] Google Data Centers, http://www.google.com/corporate/datacenter/efficient-computing/measurement.html.

[67] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A Comparison of High-Level Full-System Power Models," in *Proc. 2008 Conference on Power aware Computing and Systems*, San Diego, California, 2008.

[68] T. Pering, T. Burd, and R. Brodersen, "Dynamic Voltage Scaling and the Design of a Low-Power Microprocessor System," in *Proc. Power Driven Microarchitecture Workshop 1998*, June 1998.

[69] G. Semeraro, G. Magklis, R. Balasubramonian, D. Albonesi, S. Dwarkadas, and M. Scott, "Energy-efficient Processor Design Using Multiple Clock Domains with Dynamic Voltage and Frequency Scaling," in *Proc. 8th International Symposium on High-Performance Computer Architecture 2002.*, Feb. 2002, pp. 29 – 40.

[70] W. Felter, K. Rajamani, T. Keller, and C. Rusu, "A Performance-Conserving Approach for Reducing Peak Power Consumption in Server Systems," in *Proc. 19th International Conference on Supercomputing (ICS '05)*, Cambridge, Massachusetts, 2005, pp. 293–302.

[71] B. Diniz, D. Guedes, W. Meira, Jr., and R. Bianchini, "Limiting the Power Consumption of Main Memory," in *Proc. 34th annual Symposium on Computer Architecture*, San Diego, 2007, pp. 290–301.

[72] H. Zheng, J. Lin, Z. Zhang, E. Gorbatov, H. David, and Z. Zhu, "Mini-rank: Adaptive DRAM Architecture for Improving Memory Power Efficiency," in *Proc. 41st annual IEEE/ACM International Symposium on Microarchitecture*, Washington, 2008, pp. 210–221.

[73] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke, "DRPM: Dynamic Speed Control for Power Management in Server Class Disks," in *Proc. 30th annual International Symposium on Computer Architecture*, San Diego, California, Jun. 2003, pp. 169–181.

[74] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu, "Delivering Energy Proportionality with Non Energy-Proportional Systems-Optimizing the Ensemble," in *Proc. First Workshop on Power Aware Computing and Systems*, San Diego, CA, Dec. 7, 2008.

[75] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: Eliminating Server Idle Power," in *Proc. 14th international conference on Architectural support for programming languages and operating systems (ASPLOS '09)*, Washington, DC, USA, 2009, pp. 205–216.

[76] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services," in *Proc. USENIX Symposium on Networked Systems Design and Implementation*, San Francisco, USA, 2008, pp. 337–350.

[77] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang, "Power and Performance Management of Virtualized Computing Environments Via Lookahead Control," in *Proc. International Conference on Autonomic Computing*, Jun. 2008, pp. 3 –12.

[78] P. Bodík, R. Griffith, C. Sutton, A. Fox, M. Jordan, and D. Patterson, "Statistical Machine Learning Makes Automatic Control Practical for Internet Datacenters," in *Proc. conference on Hot topics in cloud computing*, San Diego, California, 2009, San Diego, California.

[79] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler, and R. H. Katz, "NapSAC: Design and Implementation of a Power-Proportional Web Cluster," in *Proc. first ACM SIGCOMM workshop on Green networking*, New Delhi, India, Aug. 30 - Sep.3 2010, pp. 15–22.

[80] X. Zheng and Y. Cai, "Achieving Energy Proportionality In Server Clusters," *International Journal of Computer Networks (IJCN)*, vol. 1, no. 2, pp. 21–35, 2010.

[81] S. Srikantaiah, A. Kansal and F. Zhao, "Energy Aware Consolidation for Cloud Computing," in *Proc. Conference on Power Aware Computing and Systems*, San Diego, CA, Dec. 2008.

[82] A. Kansal and F. Zhao, "Fine-Grained Energy Profiling for Power-Aware Application Design," *SIGMETRICS Performance Evaluation Review*, vol. 36, no. 2, pp. 26–31, 2008.

[83] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes, "Hibernator: Helping Disk Arrays Sleep Through the Winter," in *Proc. twentieth ACM Symposium on Operating Systems Principles (SOSP '05)*, Brighton, UK, 2005, pp. 177–190.

[84] A. Verma, R. Koller, L. Useche, and R. Rangaswami, "SRCMap: Energy Proportional Storage Using Dynamic Consolidation," in *Proc. 8th USENIX conference on File and storage technologies (FAST'10)*, San Jose, CA, 2010.

[85] L. Ganesh, H. Weatherspoon, M. Balakrishnan, and K. Birman, "Optimizing Power Consumption in Large Scale Storage Systems," in *Proc. 11th USENIX workshop on Hot topics in operating systems (HOTOS'07)*, San Diego, CA, 2007, pp. 1–6.

[86] E. Thereska, A. Donnelly, and D. Narayanan, "Sierra: a Power-Proportional, Distributed Storage System," Microsoft Research Ltd., Tech. Rep. MSR-TR-2009-153, Nov. 2009.

[87] H. Amur, J. Cipar, V. Gupta, G. R. Ganger, M. A. Kozuch, and K. Schwan, "Robust and Flexible Power-Proportional Storage," in *ACM Symposium on Cloud Computing*, Indianapolis, Indiana, Jun. 2010.

[88] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A Power Benchmarking Framework for Network Devices," in *Proc. 8th International IFIP-TC 6 Networking Conference*, Aachen, Germany, 2009, pp. 795–808.

[89] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power Awareness in Network Design and Routing," in *Proc. IEEE International Conference on Computer Communications*, Phoenix, USA, April 13-18, 2008, Phoenix, USA, pp. 457–465.

[90] "IEEE P802.3az Energy Efficient Ethernet Task Force," http://grouper.ieee.org/groups/802/3/az/public/index.html.

[91] C. Gunaratne, K. Christensen, B. Nordman, and S. Suen, "Reducing the Energy Consumption of Ethernet with Adaptive Link Rate (ALR)," *IEEE Trans. Computers*, vol. 57, no. 4, pp. 448–461, Apr. 2008.

[92] P. Reviriego, J. Hernandez, D. Larrabeiti, and J. Maestro, "Performance Evaluation of Energy Efficient Ethernet," *IEEE Commun. Lett.*, vol. 13, no. 9, pp. 697 –699, Sep. 2009.

[93] P. Reviriego, J. Maestro, J. Hernándndez, and D. Larrabeiti, "Burst Transmission for Energy-Efficient Ethernet," *IEEE Internet Computing*, vol. 14, no. 4, pp. 50 –57, Jul. 2010.

[94] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing Network Energy Consumption via Sleeping and Rate-Adaptation," in *Proc. USENIX Symposium on Networked Systems Design and Implementation*, San Francisco, USA, 2008, pp. 323–336.

[95] G. Ananthanarayanan and R. H. Katz, "Greening the Switch," in *Proc. Power Aware Computing and Systems Conference*, San Diego, CA, Dec. 2008.

[96] R. Bolla, R. Bruschi, F. Davoli, and A. Ranieri, "Performance Constrained Power Consumption Optimization in Distributed Network Equipment," in *Proc. International Conference on Communications*, Dresden, Germany, Jun.14-18 2009, pp. 1–6.

[97] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee and N. McKeown, "ElasticTree: Saving Energy in Data Center Networks," in *Proc. 7th ACM/USENIX Symposium on Networked Systems Design and Implementation*, San Jose, CA, April 2010, pp. 249–264.

[98] P. Mahadevan, S. Banerjee, and P. Sharma, "Energy Proportionality of an Enterprise Network," in *Proc. ACM SIGCOMM*, New Delhi, India, Aug. 30 - Sep.3 2010.

[99] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "Energy Aware Network Operations," in *Proc. IEEE IEEE International Conference on Computer Communications*, Rio de Janeiro, Brazil, April 19-25, 2009, Rio de Janeiro, Brazil, pp. 1–6.

[100] Y. Shang, D. Li, and M. Xu, "Energy-Aware Routing in Data Center Network," in *Proc. first ACM SIGCOMM workshop on Green networking*, New Delhi, India, Aug. 30 - Sep.3 2010, pp. 1–8.

[101] N. Vasić, D. Novaković, S. Shekhar, P. Bhurat, M. Canini, and D. Kostić, "Responsive, Energy-Proportional Networks," EPFL, Tech. Rep. EPFL-REPORT-149959, 2010.

[102] S. Nedevschi, J. Chandrashekar, J. Liu, B. Nordman, S. Ratnasamy, and N. Taft, "Skilled in the art of being idle: reducing energy waste in networked systems," in *Proc. USENIX symposium on Networked systems design and implementation*, Boston, 2009, pp. 381–394.

[103] L. Liu, H. Wang, X. Liu, X. Jin, W. B. He, Q. B. Wang, and Y. Chen, "GreenCloud: a New Architecture for Green Data Center," in *Proc. 6th International Conference on Autonomic Computing and Communications Industry Session*, Barcelona, Spain, 2009, pp. 29–38.

[104] C. Patel, R. Sharma, C. Bash, and A. Beitelmal, "Thermal Considerations in Cooling Large Scale High Compute Density Data Centers," in *Proc. Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, 2002, pp. 767 – 776.

[105] C. Patel, C. Bash, R. Sharma, M. Beitelmam, and R. Friedrich, "Smart Cooling of Data Centers," in *Proc. InterPack*, July 2003.

[106] C. Bash and G. Forman, "Cool Job Allocation: Measuring the Power Savings of Placing Jobs at Cooling-Efficient Locations in the Data Center," HP Laboratories, Tech. Rep. HPL-2007-62, 2007.

[107] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making scheduling "cool": Temperature-Aware Workload Placement in Data Centers," in *Proc. USENIX Annual Technical Conference*, 2005, pp. 61–75.

[108] L. Parolini, B. Sinopoli, and B. H. Krogh, "Reducing Data Center Energy Consumption via Coordinated Cooling and Load management," in *Proc. Power Aware Computing and Systems*, San Diego, 2008.

[109] N. Vasic, T. Scherer, and W. Schott, "Thermal-Aware Workload Scheduling for Energy Efficient Data Centers," in *Proc. 7th International Conference on Autonomic Computing (ICAC '10)*, Washington, DC, USA, 2010, pp. 169–174.

[110] Q. Tang, T. Mukherjee, S. Gupta, and P. Cayton, "Sensor-Based Fast Thermal Evaluation Model For Energy Efficient High-Performance Datacenters," in *Proc. 4th International Conference on Intelligent Sensing and Information Processing*, Oct. 2006, pp. 203 –208.

[111] C.-J. M. Liang, J. Liu, L. Luo, A. Terzis, and F. Zhao, "RACNet: a High-Fidelity Data Center Sensing Network," in *Proc. 7th ACM Conference on Embedded Networked Sensor Systems*, Berkeley, California, 2009, pp. 15–28.

[112] F. Ahmad and T. N. Vijaykumar, "Joint Optimization of Idle and Cooling Power in Data Centers While Maintaining Response Time," in *Proc. ASPLOS on Architectural Support for Programming Languages and Operating Systems*, Pittsburgh, USA, 2010, pp. 243–256.

[113] "IBM tries running data centers on solar power," http://www.eetimes.com/electronics-news/4230811/IBM-tries-running-datacenters-on-solar-power.

[114] "Apple hopes to turn green with solar power data centre," http://www.guardian.co.uk/environment/2011/nov/23/apple-green-solar-data-centre.

[115] K. Le, R. Bianchini, M. Martonosi, and T. D. Nguyen, "Cost- and Energy-Aware Load Distribution Across Data Centers," in *Proc. Workshop on Power Aware Computing and Systems*, October 10, 2009, Big Sky, MT.

[116] K. Le, O. Bilgir, R. Bianchini, M. Martonosi, and T. D. Nguyen, "Capping the Brown Energy Consumption of Internet Services at Low Cost," in *Proc. International Green Computing Conference*, August 2010.

[117] C. Stewart and K. Shen, "Some Joules Are More Precious Than Others: Managing Renewable Energy in the Datacenter," in *Proc. Workshop on Power Aware Computing and Systems*, October 10, 2009, Big Sky, MT.

[118] Z. Liu, M. Lin, A. Wierman, S. Low, and L. Andrew, "Greening Geographical Load Balancing," in *Proc. ACM SIGMETRICS joint International Conference on Measurement and Modeling of Computer Systems*, San Jose, California, USA, June 7-11, 2011, pp. 233–244.

[119] I. Goiri, K. Le, M.E. Haque, R. Beauchea, T.D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "GreenSlot: Scheduling Energy Consumption in Green Datacenters," in *Proc. 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11)*, Seattle, Washington, 2011, pp. 20:1–20:11.

[120] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The Nature of Data Center Traffic: Measurements & Analysis," in *Proc. ACM SIGCOMM*, Chicago, Illinois, 2009, pp. 202–208.

[121] X. Meng, V. Pappas, and L. Zhang, "Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement," in *Proc. IEEE International Conference on Computer Communications*, San Diego, CA, USA, March 15-19, 2010, pp. 1154–1162.

[122] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," in *Proc. ACM SIGCOMM*, New Delhi, India, Aug. 30 - Sep.3 2010, pp. 63–74.

[123] H. Abu-Libdeh, P. Costa, A. Rowstron, G. O'Shea, and A. Donnelly, "Symbiotic Routing in Future Data Centers," in *Proc. ACM SIGCOMM*, New Delhi, India, Aug. 30 - Sep.3 2010, pp. 51–62.

[124] D. A. Joseph, A. Tavakoli, and I. Stoica, "A Policy-Aware Switching Layer for Data Centers," in *Proc. ACM SIGCOMM*, Seattle, WA, August 17-22, 2008, pp. 51–62.

[125] G. Lu, Y. Shi, C. Guo, and Y. Zhang, "CAFE: a Configurable Packet Forwarding Engine for Data Center Networks," in *Proc. ACM SIGCOMM*, Barcelona, Spain, 2009, pp. 25–30.

[126] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving Datacenter Performance and Robustness with Multipath TCP," in *Proc. ACM SIGCOMM*, August 15-19, 2011, Toronto, Ontario, Canada.

[127] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic Flow Scheduling for Data Center Networks," in *Proc. 7th ACM/USENIX Symposium on Networked Systems Design and Implementation*, San Jose, CA, Apr. 2010.

[128] A. R. Curtis, J. C. Mogul, J. Tourrilhes, and P. Yalagandula, "DevoFlow: Scaling Flow Management for High-Performance Networks," in *Proc. SIGCOMM*, August 15-19, 2011, Toronto, Ontario, Canada.

[129] L. Han, J. Wang, and C. Wang, "A Novel Multipath Load Balancing Algorithm in Fat-Tree Data Center," in *Proc. 1st International Conference on Cloud Computing*, Beijing, China, 2009, pp. 405–412.

[130] S. Lim, B. Sharma, G. Nam, E. K. Kim, C. R. Das, "MDCSim: A Multi-tier Data Center Simulation Platform," in *Proc. Cluster Computing and Workshops CLUSTER*, 2009.

[131] R. N. Calheiros, R. Ranjan, C. De Rose, and R. Buyya, "CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services," Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, Tech. Rep. GRIDS-TR-2009-1, 2009.

[132] D. Meisner and T. F. Wenisch, "Stochastic Queuing Simulation for Data Center Workloads," in *Proc. Workshop on Exascale Evaluation and Research Techniques*, Pittsburg, March 2010.

[133] Z. Tan, K. Asanovi, and D. Patterson, "Datacenter-Scale Network Research on FPGAs," in *Proc. Workshop on Exascale Evaluation and Research Techniques*, Newport Beach, CA, March 2011.

[134] Data Center Specialist Group, "DCSG Simulator Software Project," http://dcsg.bcs.org/welcome-dcsg-simulator.

[135] C. Beckmann, O. Khan, S. Parthasarathy, A. Klimkin, M. Gambhir, B. Slechta, and K. Rangan,, "Multithreaded Simulation to Increase Performance Modeling Throughput on Large Compute Grids," in *Proc. Workshop on Exascale Evaluation and Research Techniques (EXERT)*, Pittsburg, March 2010.

[136] IBM, "Venus - Interconnection Network Simulation," https://researcher.ibm.com/researcher/view_project.php?id=1071.

Yan Zhang (S'10) received the B.E. and M.E. degrees in Electrical Engineering from Shandong University, China, in 2001 and 2004, respectively. She is currently pursuing her Ph.D. degree in Computer Engineering in the Department of Electrical and Computer Engineering at the New Jersey Institute of Technology (NJIT), Newark, New Jersey. Her research interests include congestion control and energy optimization in data center networks, content delivery acceleration over wide area networks, and energy efficient networking.

Nirwan Ansari (S'78-M'83-SM'94-F'09) received the B.S.E.E. (*summa cum laude* with a perfect GPA) from the New Jersey Institute of Technology (NJIT), Newark, in 1982, the M.S.E.E. degree from University of Michigan, Ann Arbor, in 1983, and the Ph.D degree from Purdue University, West Lafayette, IN, in 1988.

He joined NJIT's Department of Electrical and Computer Engineering as Assistant Professor in 1988, became tenured Associate Professor in 1993, and has been Full Professor since 1997. He has also assumed various administrative positions at NJIT. He was Visiting (Chair) Professor at several universities. He authored *Computational Intelligence for Optimization* (Springer, 1997, translated into Chinese in 2000) with E. S. H. Hou, and edited *Neural Networks in Telecommunications* (Springer, 1994) with B. Yuhas. He has also contributed over 400 technical papers, over one third of which were published in widely cited refereed journals/magazines. He has also guest-edited a number of special issues, covering various emerging topics in communications and networking. His current research focuses on various aspects of broadband networks and multimedia communications.

Prof. Ansari has served on the Editorial Board and Advisory Board of eight journals, including as a Senior Technical Editor of the *IEEE Communications Magazine* (2006-2009). He has served the IEEE in various capacities such as Chair of the IEEE North Jersey Communications Society (COMSOC) Chapter, Chair of the IEEE North Jersey Section, Member of the IEEE Region 1 Board of Governors, Chair of the IEEE COMSOC Networking Technical Committee Cluster, Chair of the IEEE COMSOC Technical Committee on Ad Hoc and Sensor Networks, and Chair/Technical Program Committee Chair of several conferences/symposia. Some of his recent recognitions include a 2007 IEEE Leadership Award from the Central Jersey/Princeton Section, NJIT Excellence in Teaching Award in Outstanding Professional Development in 2008, a 2008 IEEE MGA Leadership Award, the 2009 NCE Excellence in Teaching Award, a couple of best paper awards (IC-NIDC 2009 and IEEE GLOBECOM 2010), a 2010 Thomas Alva Edison Patent Award, and designation as an IEEE Communications Society Distinguished Lecturer (2006-2009, two terms).