FCell: Towards the Tradeoffs in Designing Data Center Network Architectures

Dawei Li and Jie Wu
Department of Computer and Information Sciences
Temple University, Philadelphia, USA
{dawei.li, jiewu}@temple.edu

Abstract*—We propose a novel Data Center Network (DCN) architecture, named FCell, which is a tradeoff design in three aspects. First, FCell reflects a tradeoff between DCN power consumption and network performances, which mainly include end-to-end delays and bisection bandwidth. We propose a unified path length definition to characterize end-to-end delays in general DCNs. Comparisons with existing DCN architectures reveal that FCell consumes a moderate amount of power, and achieves generally low end-to-end delays and a satisfiable bisection bandwidth. Second, FCell reflects a tradeoff between switch-centric and server-centric designs. Two basic routing schemes are proposed to show that FCell can place routing intelligence on both servers and switches; thus, FCell can be regarded as a *dual-centric* architecture, which enjoys both the fast switching capability of switches and the high programmability of servers. Third, FCell reflects a tradeoff between scalability and flexibility. Scalability of FCell comes from its regularity; FCell also supports flexible growth of network size with minimal modifications on its original architecture. Through simulations, we evaluate the performances of the two routing schemes in different traffic conditions in FCell, and verify that our unified path length definition is a useful metric to characterize end-to-end delays in general DCNs.

Index Terms*—*Data center network (DCN), power consumption, end-to-end delay, bisection bandwidth, dual-centric design.

I. Introduction

Data centers have become important infrastructures to support various cloud computing services. The Data Center Network (DCN) has significant influences on the quality of the services that the data center can provide to the applications that it hosts.

Performance vs. Power. Two important performance metrics for a DCN architecture are end-to-end delays in the DCN and the bisection bandwidth. End-to-end delays translate directly to applications' response times in various situations. Bisection bandwidth provides key information on the potential throughput that the network can provide and the fault-tolerance capabilities. the DCN power consumption has also become an important issue [1]. To provide low end-to-end delays and high bisection bandwidth, large numbers of networking devices are usually used in DCNs. For example, in Fat-Trees [2], three levels of switches are used, resulting in a high network power consumption. To achieve a low DCN power consumption, other architectures use significantly fewer networking devices. For example, in DPillar [3], SWCube, SWKautz [4], DCell [5], BCN [6], and FiConn [7], the number of switches used is largely reduced, though a small number of extra NIC ports (typically less than 4) are required on servers. The DCN power consumption of these architectures is generally less than that of Fat-Trees. However, these architectures heavily rely on servers for packet forwarding. Since servers usually have much greater processing delays than switches, especially when servers' packet forwarding scheme is software-based, the end-to-end delays in these architectures are much greater; besides, these architectures also have a much lower bisection bandwidth. Can we achieve high performances and low power consumption at the same time?

Switch-centric vs. Server-centric. Existing DCN architectures have been classified into two categories: switch-centric and server-centric architectures [8]. In switch-centric designs [1], [2], [9], routing intelligence is placed on switches; servers are equipped with one NIC port, and are not involved in forwarding packets for other servers. In server-centric designs [3]–[7], [10], switches are only used as cross-bars, and routing intelligence is placed on servers; servers are usually equipped with multiple NIC ports, and act as both computing and packet forwarding nodes. Switch-centric architectures enjoy the fast switching capability of switches, but switches are less programmable than servers. Server-centric architectures enjoy the high programmability of servers, but servers usually have larger processing delays than do switches. Can we combine the advantages of both categories?

Scalability vs. Flexibility. Scalability requires that the networking devices, typically the switches, rely on a small amount of information, which does not increase significantly with the network size, to make efficient routing decisions. Flexibility means that expanding the network in a fine-grained fashion should not destroy the current architecture or replace the networking devices. Since modern data centers usually have large network sizes, scalability is an important requirement. Also, data centers require flexible growth of network size after initial deployment, due to the rapidly increasing needs. Regular architectures are generally highly scalable, but do not support flexible growth of the network size due to their rigid topologies. Some regular architectures are able to increase the network size, but have certain limitations. For example, FiConn supports coarse-grained growth; because adding one level to the architecture will make the network size increase by tens, or even hundreds of times, which does not reflect practical needs. Recent works have proposed random networks, such as Jellyfish [11], Scafida [12], and Small-World Data Center [13], to provide arbitrary-grained flexibility; however, due to their irregularity, networking devices in these architectures rely on a large amount of information for efficient routing, making them unable to scale to a large network size. Can we design both scalable and flexible DCN architectures?

In this paper, we consider the tradeoffs (in all of the above-mentioned three aspects) in designing DCN architectures. Our main contributions are as follows.

- First, we propose a unified path length definition, and consequently, a unified diameter definition, to characterize the end-to-end delays in a general DCN. Also, a DCN power consumption model is presented to characterize the power efficiency of general DCNs.
- Second, we propose a novel DCN architecture, named FCell. FCell reflects a tradeoff between switch-centric and server-centric designs, and can be regarded as a *dual-centric* architecture, where routing intelligence can be placed on both switches and servers. FCell also reflects a tradeoff between scalability and flexibility. Scalability of FCell comes from its high degree of regularity; meanwhile, FCell is a flexible architecture that supports two fundamental ways to expand its network size in a fine-grained fashion.
- Third, based on our unified path length, diameter definitions and DCN power consumption model for general DCNs, we conduct quantitative comparisons between FCell and several typical existing DCN architectures. Results show that FCell reflects a tradeoff between network performances and DCN power consumption.
- Fourth, we conduct simulations to evaluate the performances of the two basic routing schemes under various traffic conditions in FCell. Also, we verify that our unified path length definition, and thus the unified diameter definition, are useful metrics to characterize end-to-end delays in general DCNs.

The rest of the paper is organized as follows. Section II presents the unified path length, DCN diameter definitions, and DCN power consumption model. We describe our novel DCN architecture, FCell in Section III, where we also illustrate that FCell is a dual-centric architecture, and that FCell is scalable and flexible. We review related existing works in Section IV. Quantitative comparisons on the network performances and the DCN power consumption between FCell and several typical existing architectures are provided in Section V. Supporting simulations are conducted in Section VI. Conclusions are made in Section VII.

II. Unified Path Length, Diameter Definitions, and DCN Power Consumption Model

To characterize the end-to-end delays between two servers in a DCN, the concept of diameter is usually used, which is defined as the maximum length of the shortest path between any pair of two servers. However, for switch-centric and server-centric architectures, the path lengths are calculated differently. For switch-centric architectures, the length of a path is calculated as the number of links in the path [14], [15]; for server-centric architectures, the length is calculated as the number of servers (excluding the source and the destination) in the path between the two servers, plus 1 [3]–[7], [10]. A diameter of 6 in Fat-Tree means totally different things from a diameter of 6 in BCube. However, a lot of works still compare these two different kinds of diameters [5], [8], [14], [15]. This somewhat confuses the understanding of the end-to-end delay in a general DCN.

In a DCN, the end-to-end delay of a packet from a source server to a destination server consists of the delays on all the devices that the packet traverses. For the ease of presentation, we assume that all the switches and servers are homogeneous. Packets on switches and servers experience three important delays: *processing delay*, *transmission delay*, and *queuing delay*; we denote them as $d_{w,p}$, $d_{w,t}$, $d_{w,q}$ and $d_{v,p}$, $d_{v,t}$, $d_{v,q}$ for switches and servers, respectively. The processing delay is the time required to examine the packet's head and determine where to direct the packet. Queuing delays largely depend on network traffic conditions and routing protocols. Currently, our focus is on the architectures of DCNs; thus, we do not consider the queuing delay explicitly, and just assume that $d_{w,q}$=$d_{v,q}$=0.

Switches can operate in two modes: store-and-forward and cut-through. In store-and-forward mode, a switch needs to receive all the flits of the packet before it forwards the packet to the next device. The total delay on the switch is $d_w = d_{w,p}+d_{w,t}$. The typical value of $d_{w,p}$ is around $2\mu s$ [16]. $d_{w,t} = S_{packet}/r_{bit}$, where S_{packet} is the size of the packet and r_{bit} is the data transmission rate. S_{packet} varies between 64 bytes and 1514 bytes. Given data transmission rate r_{bit} = 1Gbps, $d_{w,t}$ varies from about $0.5\mu s$ to about $10\mu s$. In cut-through mode, a switch starts forwarding the packet when it receives the first flit of the packet. Thus, the transmission delay is negligible, and the total delay is around d_w=$d_{w,p}$=$2\mu s$.

The packet forwarding scheme on a server can be implemented in either software or hardware. In software-based forwarding, the processing delay on a server, $d_{v,p}$ is much higher than that on a switch, with a typical value of about $10\mu s$ [16]. Depending on CPU load and NIC configuration, this value varies significantly. In hardware-based forwarding, $d_{v,p}$ can be reduced to be close to the processing delay on a switch [17]. We do not delve into the detailed implementation of the packet forwarding schemes on servers. The overall delay on a server is d_v=$d_{v,p}$+$d_{v,t}$, where $d_{v,t}$ can be calculated in the same way as $d_{w,t}$. Based on the typical values, d_v is generally 1 to several times of d_w.

Network links have *propagation delay*, d_l, which can be calculated by dividing the length of the link (L_{link}) by the speed of the signal in the transmission medium: $d_l = L_{link}/(\eta c)$, where η is a constant around 0.7 and c is the speed of light in vacuum. Since the length of links in a data center is usually less than 10 meters, the propagation delay on a link is usually less than $10/(0.7\times3\times10^8)s = 0.048\mu s$. Compared with the typical delays on switches and servers, the propagation delay is negligible.

Unified Path Length and Diameter Definitions. In general DCNs, both switches and servers may be used for packet forwarding. Denote the numbers of switches and servers in a path, P from a source server to a destination server by

$n_{P,w}$, and $n_{P,v}$ (excluding the source and the destination), respectively. We define the *path length* of P as follows:

$$d_P = n_{P,w}d_w + (n_{P,v} + 1)d_v, \tag{1}$$

where 1 is added to $n_{P,v}$, because the delay on the source server should be included as part of the end-to-end delay. The above path length definition applies to all general DCNs. If we assume that $d_w = d_v = 1$, the above path length definition is consistent with the path lengths in a switch-centric architecture. If we assume that d_v=1 and that d_w is negligible, the above path length definition is consistent with the path lengths in a server-centric architecture. Under this unified path length definition, we define the *diameter of a general DCN* as the maximum path length (based on (1)) of the shortest paths between all pairs of servers in the DCN:

$$d = \max_{P \in \{\mathcal{P}\}} d_P, \tag{2}$$

where $\mathcal{P}$ is the set of shortest paths between all pairs of servers in the DCN.

DCN Power Consumption Model. We consider the power consumption of all DCN devices. A switch's power consumption, p_w is part of the DCN power consumption. For a server in a switch-centric architecture, only the NIC's power consumption, p_{nic} belongs to the DCN power consumption. In a DCN where the server can be used for packet forwarding for other servers, the power consumption of the server's packet forwarding engine, either software-based or hardware-based, should also be included as the DCN power consumption. We denote p_{fwd} as the power consumption of the server's packet forwarding engine (either the CPU core's power consumption for software-based forwarding [18] or the additional hardware's power consumption for hardware-based forwarding [17]), and denote the extent to which a server is involved in packet forwarding by α. The overall *DCN power consumption* can be calculated as follows: p_{dcn}=$N_w p_w$+$n_{nic}N_v p_{nic}$+$\alpha N_v p_{fwd}$, where N_w and N_v are the numbers of switches and servers in the DCN, respectively, and n_{nic} is the average number of NIC ports used on a server. Since different DCNs can hold different numbers of servers, we define the *DCN power consumption per server* as the power efficiency metric of a general DCN:

$$p_V = p_{dcn}/N_v = p_w N_w/N_v + n_{nic}p_{nic} + \alpha p_{fwd}. \tag{3}$$

For switch-centric architectures, $\alpha = 0$. For DCNs where servers are involved for packet forwarding for other servers, α depends on various factors; for simple and fair comparison, we can choose $\alpha = 1$. A practical value of p_w for a switch with 48 1Gbps ports is about 150 Watts [19]; a practical value of p_{nic} for 1Gbps NIC port is 2 Watts [20]. As reported in [18], when software-based forwarding is used, the CPU cores can be in reserved or shared models, which correspond to different p_{fwd} values, varying around 5Watts if NIC ports are 10Gbps. The value for p_{fwd} will be lower if NIC ports are 1Gbps. In hardware-based forwarding, p_{fwd} may also have quite different values [17].

Algorithm 1 FCellBuild(n)

```
1: Label the jth server in the ith cluster in an FCell($n$) as
   $a_{i,j}$, where $0 \le i \le n^2/2$, and $0 \le j \le n^2/2 - 1$.
2: for $i = 0$ to $n^2/2 - 1$ do
3:     for $j = i$ to $n^2/2 - 1$ do
4:         Connect server $a_{i,j}$ with server $a_{j+1,i}$.
```

III. FCell: A Novel DCN Architecture

The motivation of our work is to design high performance architectures with low DCN power consumption. An intuitive remedy for switch-centric architectures, such as Fat-Tree, is to reduce the levels of switches. However, this makes the DCN unable to scale to a practically large size. Thus, we consider using interconnections among servers to scale the network. The detailed construction of FCell is presented in the following.

A. FCell Construction

An FCell built from servers with 2 NIC ports and switches with n ports is denoted by FCell(n). An FCell(n) consists of $n^2/2+1$ clusters. In each cluster, there are two levels of switches: $n/2$ level 2 switches and n level 1 switches. Every level 2 switch is connected to every level 1 switch. In other words, the set of $n/2$ level 2 switches and the set of n level 1 switches form a complete bipartite graph. Then, there are $n/2$ ports remaining on each of the level 1 switches; we use these ports to connect $n/2$ servers. A level 1 switch is also called a Top of Rack (ToR) switch. As a result, the switches and servers in one cluster form a simple instance of the folded Clos [9] topology. The numbers of switches and servers in each cluster are $3n/2$ and $n^2/2$, respectively.

Servers in all of the $n^2/2+1$ clusters are interconnected in a similar way to that of DCell [5]. Simply put, each of the $n^2/2$ servers in a cluster is directly connected to another server in each of the other $n^2/2$ clusters. Thus, if we regard each cluster as a single node, the $n^2/2$+1 clusters will form a complete graph. We denote a server by $a_{i,j}$, which represents the jth server in the ith cluster, $\forall 0 \le i \le n^2/2, 0 \le j \le n^2/2$–1. Algorithm 1 shows the detailed interconnections for building FCell(n) from the $n^2/2+1$ clusters. Fig. 1 shows the interconnections of an FCell(4), where each switch has 4 ports.

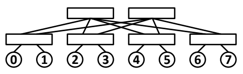
B. FCell Basic Properties

Property 1. *In an FCell(n), the number of switches is $N_w = 3n(n^2+2)/4$, and the number of servers is $N_v = n^2(n^2+2)/4$.*

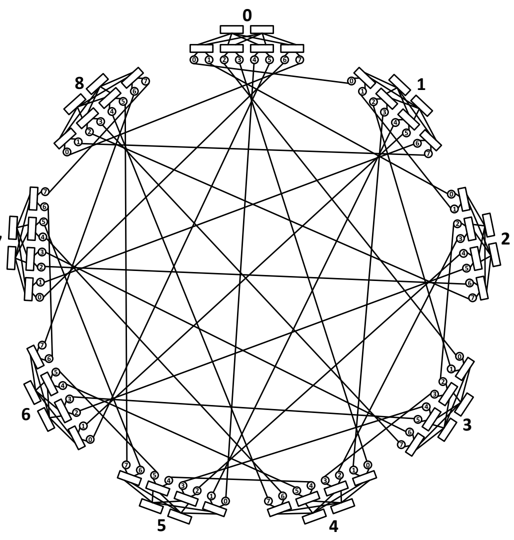
Proof. There are $n^2/2 + 1$ clusters, each with $3n/2$ switches and $n^2/2$ servers. □

Property 2. *The diameter of an FCell(n) is $d = 6d_w + 3d_v$.*

Proof. The diameter is defined as the maximum length of the shortest path between two servers. Obviously, the longest shortest path in an FCell is between two servers that are not in the same cluster. We consider two servers, $a_{i,j}$ and $a_{k,l}$, which are not in the same cluster, i.e., $i \neq k$. Without loss of generality, we assume that $0 \le i < k \le n^2/2$. According to the interconnection rules of FCell, the server $a_{i,k-1}$ in the ith cluster, and the server $a_{k,i}$ in the kth cluster are directly



(a) The interconnections in one cluster.



(b) Final interconnections of FCell(4).

Fig. 1. An example of FCell(4). We consistently use rectangles to represent switches and circles to respresent servers, if not otherwise specified.

connected. The shortest path from server $a_{i,j}$ to server $a_{k,l}$ consists of at most three segments: 1) the shortest path from server $a_{i,j}$ to server $a_{i,k-1}$ in the ith cluster, 2) the path from server $a_{i,k-1}$ to server $a_{k,i}$, and 3) the shortest path from server $a_{k,i}$ to server $a_{k,l}$ in the kth cluster. Though $a_{i,j}$ and $a_{i,k-1}$ may be the same server, or may connect to the same level 1 switch, in the worst case, the shortest path from $a_{i,j}$ to $a_{i,k-1}$ includes 3 switches. Similarly, the shortest path from $a_{k,i}$ to $a_{k,l}$ includes 3 switches in the worst case. Thus, the shortest path from $a_{i,j}$ and $a_{k,l}$ includes at most 6 switches (at most 3 in each cluster), and at most 2 servers (excluding the source and destination). According to (1) and (2), the diameter of an FCell(n) is $d = 6d_w + 3d_v$. □

We assume that all the links in a DCN have a unit bandwidth, 1. Then, bisection bandwidth of a DCN is the minimal number of links to be removed to partition the DCN into two parts of "equal" sizes that differ by at most 1. We conjecture that FCell has the following property.

Property 3. *The bisection bandwidth of an FCell(n) is $B \approx N_v/4$.*

Proof. The bisection bandwidth of a complete graph with N nodes, when N is even, is $N/2 \times N/2 = N^2/4$. The reason is quite straightforward. The cut partitions the N nodes into two equal sets, each consisting of $N/2$ nodes. Since each node in one set has a link to every node in the other set, the total number of links in the cut is $N/2 \times N/2 = N^2/4$, which is the bisection bandwidth. When N is odd, the bisection bandwidth is $(N+1)/2 \times (N-1)/2 = (N^2-1)/4$. As has been mentioned, for the FCell architecture, we can regard each of the $(n^2/2+1)$ clusters as a single node; then, these nodes form a complete graph. Besides, within each cluster, the architecture has a much greater bisection bandwidth, just like Fat-Tree and folded-Clos. For a cut to have a minimum number of links, it should try to avoid cutting through the clusters. Thus, the cut should cut between clusters. As a result, the bisection bandwidth of FCell is approximately equal to the bisection bandwidth of a complete graph with $(n^2/2+1)$ nodes: $B \approx 1/4(n^2/2+1)^2 \approx N_v/4$. □

Property 4. *The DCN power consumption per server of an FCell(n) is $p_V = 3p_w/n + 2p_{nic} + p_{fwd}$.*

Proof. The switch-number to server-number ratio in an FCell(n) is $N_w/N_v = 3/n$; in an FCell, all servers are equipped with 2 NIC ports, and servers may be involved in forwarding packets for other servers. □

We will show that FCell is a tradeoff design between network performances and DCN power consumption by comparing with various existing architectures in Section V. In the following two subsections, we illustrate FCell's two other tradeoffs: between switch-centric and server-centric designs, and between scalability and flexibility.

C. FCell Routing Schemes

We present two basic routing schemes: shortest path routing and detour routing, to show that FCell reflects a tradeoff between switch-centric and server-centric designs. Notice that, the two basic routing schemes are also useful for practical routing protocol design.

1) Shortest Path Routing: We denote the source and destination servers by $a_{i,j}$ and $a_{k,l}$ ($0 \le i,k \le n^2/2$ and $0 \le j,l \le n^2/2-1$), respectively. Then, the ith and the kth clusters are called source and destination clusters, respectively.

If $a_{i,j}$ and $a_{k,l}$ are in the same cluster, i.e., $i=k$, $a_{i,j}$ sends the packet to its level 1 (ToR) switch, which checks whether the destination is in the local rack. If the destination is in the local rack, the level 1 switch forwards the packet to the destination. Otherwise, it forwards the packet to a randomly chosen level 2 switch; the level 2 switch checks which rack the destination is in, and forwards the packet to the corresponding level 1 switch, i.e., the $\lfloor l/(n/2) \rfloor$th level 1 switch, which forwards the packet to the destination directly.

If $a_{i,j}$ and $a_{k,l}$ are not in the same cluster, based on servers' interconnection rules in FCell, the source server can determine the two servers (one in the source cluster, denoted by a_{i,r_1} and one in the destination cluster, denoted by a_{k,r_2}) that are directly connected. Then, $a_{i,j}$ forwards the packet to a_{i,r_1} within the source cluster. After that, a_{i,r_1} sends the packet to a_{k,r_2} directly, since they are directly connected. Finally, a_{k,r_2} forwards the packet to $a_{k,l}$ within the destination cluster.

We can see that, in all cases, the source server can determine all the server(s) in the path (including the destination) before sending the packet. Since the servers have high programmability and the decision logic is quite simple, we place the task of determining all the servers in the path on the source server. The source server initializes a *server stack (srv_stk)* that pushes the servers from the last one to the first one in the path, and indicates whether they are the true destination of the packet. For example, for the case where $a_{i,j}$ and $a_{k,l}$ are not in the same cluster, and $i < k$, $j \ne k-1$, $i \ne l$, all the servers in the shortest path are $a_{i,k-1}$, $a_{k,i}$, and $a_{k,l}$ (including the destination). The source server labels $a_{k,l}$ as the *true* destination and labels $a_{i,k-1}$ and $a_{k,i}$ as *fake* destinations. Then, it pushes $a_{k,l}$, $a_{k,i}$ and $a_{i,k-1}$ into *srv_stk* one by one. When sending the packet to a ToR switch, the source server uses the next server of the packet as the *temporary* destination, based on which, switches make decisions within the local cluster.

When another server in the DCN receives the packet, it pops the *srv_stk* of the packet. If the popped value is a true destination, the server consumes the packet. If the popped value is a fake destination, it checks whether the next server in the path of the packet is in the local cluster. If yes, it sends the packet to its ToR switch, using the next server as the temporary destination. Otherwise, it means that the next server is the server that directly connects with this server; then, it sends the packet to the next server directly.

When a switch receives the packet, only the destination (either fake or true) set by the previous sending server is visible to the switch. We call this destination a *temporary* destination. The switch makes forwarding decisions based on this temporary destination. Specifically, when a level 1 switch receives the packet, it sends the packet to the temporary destination directly if the temporary destination is in the local rack; otherwise it sends the packet to a randomly chosen level 2 switch in the local cluster. When a level 2 switch receives the packet, it sends the packet to the level 1 switch, on which the temporary destination of the packet resides.

Notice that, instead of randomly choosing, a level 1 switch can smartly choose a level 2 switch, if related information is available, and if it has the intelligence to do so. Thus, level 1 switches can help with load-balancing, traffic-aware, fault-tolerant or even multi-path routing within the local cluster.

2) Detour Routing: The problem with the shortest path routing is that, if servers in two clusters have intensive communications, the link that directly connects the two servers in each of the two clusters will become congested. Thus, the queuing delay will be increased significantly and the achievable throughput is limited by the capacity of the bottleneck link, even when other parts of the network have no traffic load. To solve this problem, a detour routing scheme can be applied. Instead of determining the shortest path from the source to the destination directly, the source server can choose to detour the packet to a randomly chosen intermediate cluster before the packet arrives at the destination cluster; we call the intermediate cluster the *relay* cluster.

After choosing the relay cluster, also based on servers' interconnection rules in FCell, the source server can determine the *first relay server* (in the relay cluster), which has a direct connection with a server in the source cluster, and the *second relay server* (in the relay cluster), which has a direct connection with a server in the destination cluster. Then, the *detour path* consists of the shortest path from the source server to the first relay server, the shortest path from the first relay server to the second relay server, and the shortest path from the second relay server to the destination server.

D. *FCell Scalability and Flexibility*

FCell has good scalability due to its high degree of regularity. As can be seen in the basic routing schemes, switches in FCell only need local information for packet forwarding. Servers only need basic configuration parameters of FCell for packet forwarding. In other words, they both need a small amount of information to make efficient routing decisions. Thus, FCell is highly scalable.

Unlike various rigid regular architectures, FCell supports flexibility quite well, i.e., it allows fine-grained incremental growth of its network size. We call the FCell(n) constructed previously in this paper a *complete* FCell(n). FCell supports two fundamental ways for expanding the network.

The first way is to expand a complete FCell. In this case, we require that the level 2 switches have a number of ports reserved for future expansion. Using one reserved port on each of the level 2 switches, we are able to add one level 1 switch with $n/2$ servers to each cluster, by connecting the added level 1 switch with each of the $n/2$ level 2 switches in the cluster. We call the cluster with $n/2$ added servers an *expanded cluster*. After this, each of the $n^2/2+1$ expanded clusters in the FCell(n) will have $n/2(n/2{+}1)$ servers, among which, $n/2$ added servers are not directly connected to other servers. Thus, we are allowed to add $n/2$ expanded clusters into the current architecture. Adding the first expanded cluster, the $(n^2/2)$th server of the ith expanded cluster is connected to the ith server in the newly added expanded cluster, which becomes the $(n^2/2+1)$th expanded cluster, $\forall 0 \le i \le n^2/2$. Adding the second expanded cluster, the $(n^2/2+1)$th server of the ith expanded cluster is connected to the ith server in the newly added $(n^2/2+2)$th expanded cluster, $\forall 0{\le}i{\le}n^2/2+1$. Continuing this process until adding the $(n/2)$th expanded cluster, the $(n^2/2+n/2-1)$th server of the ith cluster is connected to the i server in the newly added $(n^2/2+n/2)$th expanded cluster, $\forall 0{\le}i{\le}n^2/2{+}n/2{-}1$.

Notice that, the original interconnections among switches and servers are never modified. The original architecture consists of $N_v^{original}{=}n^2(n^2+2)/4$ servers. After expanding, the architecture consists of $N_v^{expanded}{=}(n^2/2{+}n/2)(n^2/2{+}n/2{+}1)$ servers. The increase of the number of servers for $n=24$ is from 83,232 to 90,300, i.e., an increase of 8.49%. The increase for $n=48$ is from 1,328,256 to 1,384,152, i.e., an increase of 4.21%. In this way, FCell supports a fine-grained incremental growth of network size, without modifying its original interconnections.

We have used one reserved port on each of the level 2 switches. If there are k ports reserved for future expansion on each level 2 switch, the expanded architecture can reach $(n^2/2 + nk/2)(n^2/2 + nk/2 + 1)$ servers. For $k = n$, it indicates a size of approximately 4 times of the original size; we argue that this meets typical requirements of network size growth. We have to admit that having reserved ports on level 2 switches is a drawback. However, only one third of the switches need to have reserved ports; this cuts down the extra initial investment for future expansion.

FCell supports another way of expanding its network size.

TABLE I
COMPARISON OF VARIOUS DCN ARCHITECTURES

	$N_v(n{=}24)$	$N_v(n{=}48)$	N_w/N_v	d	B	p_V
FDCL$(n,3)$	3,456	27,648	$5/n$	$5d_w{+}d_v$	$N_v/2$	$5p_w/n+p_{nic}$
FDCL$(n,4)$	41,472	663,552	$7/n$	$7d_w{+}d_v$	$N_v/2$	$7p_w/n+p_{nic}$
FBFLY$(4,7,3)$	49,125	—	$8/24$	$8d_w{+}d_v$	$N_v/3$	$8p_w/24+p_{nic}$
FBFLY$(8,6,6)$	—	1,572,864	$8/48$	$7d_w{+}d_v$	$N_v/3$	$8p_w/48+p_{nic}$
FCell(n)	83,232	1,328,256	$3/n$	$6d_w{+}3d_v$	$N_v/4$	$3p_w/n+2p_{nic}+p_{fwd}$
BCube$(n,3)$	331,776	5,308,416	$4/n$	$4d_w{+}4d_v$	$N_v/2$	$4p_w/n+4p_{nic}+p_{fwd}$
SWCube$(r,4)$	28,812	685,464	$2/n$	$5d_w{+}5d_v$	$(N_v/8)\times r/(r-1)$	$2p_w/n+2p_{nic}+p_{fwd}$
DPillar$(n,4)$	82,944	1,327,104	$2/n$	$6d_w{+}6d_v$	$N_v/4$	$2p_w/n+2p_{nic}+p_{fwd}$
DCell$(n,2)$	360,600	5,534,256	$1/n$	$4d_w{+}7d_v$	$>N_v/(4\log_n N_v)$	$p_w/n+3p_{nic}+p_{fwd}$
FiConn$(n,2)$	24,648	361,200	$1/n$	$4d_w{+}7d_v$	$>N_v/16$	$p_w/n+7p_{nic}/4+3p_{fwd}/4$

Instead of using $n^2/2{+}1$ clusters, we can use $m{+}1<n^2/2{+}1$ clusters to build an *incomplete* FCell, by connecting the first m $(<n^2/2)$ servers in all of the $m+1$ clusters. The method for adding clusters to an incomplete FCell is similar to that of expanding a complete FCell. This provides the possibility of flexibly adding servers, without reserving ports on the level 2 switches. Of course, the two ways to expand the network can be combined. When expanding an incomplete FCell makes the FCell complete, we can further expand the complete FCell.

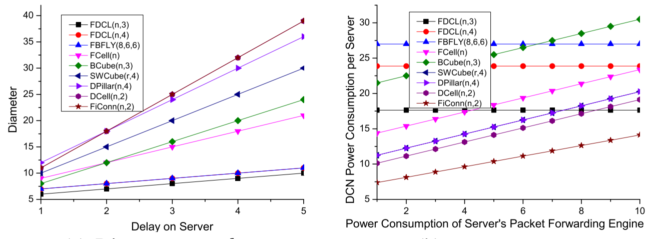
Since the original interconnections among switches and servers are never modified, after expanding the network, very limited information needs to be updated for switches and servers to make efficient routing decisions. Therefore, scalability of FCell is well maintained.

IV. RELATED EXISTING WORKS

Existing DCN architectures have been classified as switch-centric architectures and server-centric architectures.

Typical switch-centric architectures include folded-Clos [9], Fat-Tree [2], Flattened Butterfly [1], and HyperX [9]. We denote a folded-Clos DCN architecture with l levels of n-port switches with by FDCL(n,l). The switch-number to server-number ratio in an FDCL(n,l) is $N_w/N_v{=}(2l{-}1)/n$. Fat-Tree is actually a folded-Clos with 3 levels, i.e., FDCL$(n,3)$. In a Flattened Butterfly (FBFLY), switches form a generalized hypercube [21]. Then, each switch is connected to a set of c servers. An FBFLY with k dimensions and r switches along each dimension is denoted by FBFLY(r,k,c). The switch-number to server-number ratio is $N_w/N_v{=}1/c$. If the numbers of switches in each dimension are different in an FBFLY, it becomes the HyperX architecture.

Typical server-centric architectures include BCube [10], SWCube [4], DPillar [3], DCell [5], and FiConn [7]. In a BCube(n,k) the switch-number to server-number ratio is $(k{+}1)/n$ and its diameter is $(k{+}1)(d_w{+}d_v)$. It uses $k{+}1$ NIC ports on all the servers; its DCN power consumption per server is $p_V=(k{+}1)p_w/n{+}(k{+}1)p_{nic}{+}p_{fwd}$. The diameter of SWCube$(r,k)$ is $d=(k{+}1)(d_w{+}d_v)$. The diameter of DPillar(n,k) is $d=(k{+}\lfloor k/2\rfloor)(d_w{+}d_v)$. The switch-number to server-number ratios of SWCube and DPillar are both $2/n$, and they both use 2 NIC ports on all the servers; thus, their DCN power consumption per server values are both $p_V{=}2p_w/n{+}2p_{nic}{+}p_{fwd}$. For DCell and FiConn, their switch-number to server-number ratios are both $1/n$. DCell(n,k) uses $k{+}1$ NIC ports on each server; in FiConn(n,k), the average number of NICs used on a server is $2{-}1/2^k$. The diameters of DCell$(n,2)$ and FiConn$(n,2)$ are both $d=4d_w{+}7d_v$.



(a) Diameters vs. d_v. (b) p_V vs. p_{fwd}.

Fig. 2. Comparison of various architectures ($n=48$). Notice that, some of the lines are overlapped.

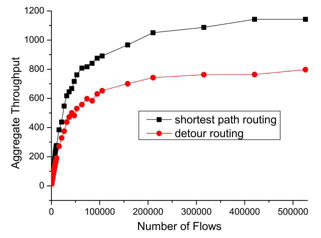
In DCell$(n,2)$, $p_V=p_w/n{+}3p_{nic}{+}p_{fwd}$. In FiConn$(n,2)$, $p_V{=}p_w/n{+}7p_{nic}/4{+}3p_{fwd}/4$.

V. COMPARISONS OF VARIOUS DCN ARCHITECTURES

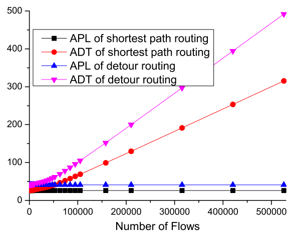
We compare various DCN architectures, constructed by the same homogenous servers and switches, with comparable numbers of servers. For architectures using 24-port and 48-port switches, basic quantitative comparisons are presented in Table I. Typical data centers have tens of thousands, or hundreds of thousands of servers, and the world's largest data centers can achieve one or two million. The numbers of servers in the table meet the needs of practical data centers.

Switch-centric architectures usually have a small diameter and a large bisection bandwidth. However their switch-number to server-number ratio is usually large, resulting in a large DCN power consumption. BCube also has a large bisection bandwidth; but it needs to use 4 levels of switches to reach a comparable DCN, and consequently 4 NIC ports on all the servers; this results in a large DCN power consumption. Other server-centric architectures, such as SWCube, DPillar, DCell and FiConn, use much fewer switches, though a small number of extra NIC ports are required on servers; their power consumption is lower than switch-centric architectures and BCube. However, they rely heavily on servers for packet forwarding; even the maximum shortest paths contain a considerable number of servers (usually $\geq$5 for them to scale to a comparable network size), which results in large end-to-end delays; besides, their bisection bandwidths are much lower.

We regard the delay on a switch, d_w as 1, and vary the delay on a server, d_v from 1 to 5. Fig. 2(a) shows the diameters of various DCN architectures. FCell has a lower diameter than all server-centric architectures when $d_v>2$, which reflects most practical situations. For switches with $n{=}48$ 1Gbps ports and 1Gbps NIC ports, we set $p_w=150$ and $p_{nic}=2$. We vary p_{fwd} from 1 to 10. Fig. 2(b) shows the DCN power consumption per server of various architectures. When $p_{fwd}\leq 4$, which also reflects most practical situations, FCell consumes less power

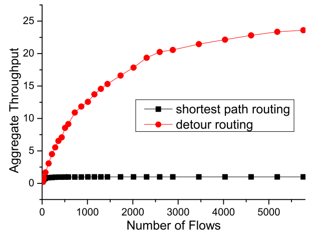


(a) Aggregate throughput.

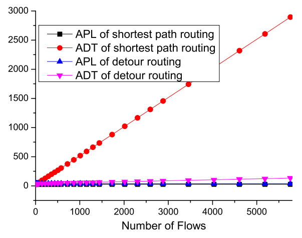


(b) APL and ADT.

Fig. 3. Aggregate throughput, APL and ADT vs. No. of flows (random traffic).



(a) Aggregate throughput.



(b) APL and ADT.

Fig. 4. Aggregate throughput, APL and ADT vs. No. of flows (bursty traffic).

than switch-centric architectures and BCube$(n, 3)$. Also, FCell has a satisfiable bisection bandwidth of $N_v/4$. We can see that FCell reflects a tradeoff design between network performances and DCN power consumption.

VI. Simulations

We develop a proprietary simulator to conduct routing simulations in FCell. Our goals are: 1) to evaluate the performances of the two basic routing schemes under different traffic conditions in FCell, and 2) to verify that our unified path length is a useful metric for end-to-end delays in a general DCN by evaluating the influences of servers' and switches' processing delays on the average path lengths and the average delivery times of the two basic routing schemes in FCell.

We build a basic model for store-and-forward switches. Both switches and servers are assumed to have 1Gbps full duplex ports. We consider single-packet flows and fixed packet size. Thus, we have a fixed transmission delay, which is considered as one unit of time, i.e., $d_{w,t} = d_{v,t} = 1$. This time unit has a typical value around 2μs. The switch's and the server's processing delays, $d_{w,p}$ and $d_{v,p}$ are normalized by this time unit. Queuing delay happens when multiple packets compete for the same output port (either on a switch or on a server) simultaneously. Two routing schemes are simulated: Shortest path Routing (SRouting) and Detour Routing (DRouting).

Two traffic patterns are considered: random and bursty traffic patterns [7]. In random traffic patterns, the source server and the destination server of each packet are randomly generated among all the servers. In bursty traffic patterns, servers in one cluster of FCell have a flow destined at other servers in another cluster. We choose the zeroth cluster and the first cluster as the source and destination clusters, respectively. In both traffic patterns, we can choose different numbers of flows to be generated, to reflect different traffic loads in the network. All of the flows are generated and pushed to the network at the same time. We calculate the aggregate throughput, the Average Path Length (APL) and the Average Delivery Time (ADT) of the two routing schemes. Aggregate throughput is defined as the average amount of data transmitted in one unit of time when all the flows are delivered to their destinations, i.e., the total data amount divided by the maximum delivery time among all flows. For APL, the path lengths are calculated based on our unified definition in (1).

A. Simulation Settings

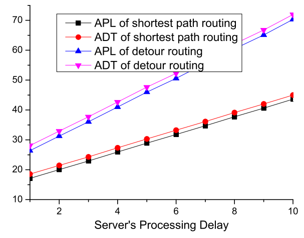
We conduct simulations on a complete FCell(12), which is built from 12-port switches. In an FCell(12), the number of servers in one cluster is $N_v^{cluster} = 12^2/2 = 72$, the total number of servers is N_v=5, 256. To evaluate the performances of the two routing schemes under different traffic conditions, we set the switch's processing delay $d_{w,p} = 1$, and set the server's processing delay $d_{v,p}$=4. For random traffic, we vary the number of flows from 657 to 525,600; for bursty traffic, we vary the number of flows from 9 to 5,760. Step sizes are different in different ranges.

To evaluate the influences of $d_{v,p}$ on APLs and ADTs, we set $d_{w,p} = 1$, and vary $d_{v,p}$ from 1 to 10. To evaluate the influences of $d_{w,p}$, we set $d_{v,p}$=10, and vary $d_{w,p}$ from 1 to 10. We choose the number of flows equal to the number of servers, N_v=5, 256 for random traffic; and choose the number of flows equal to $(N_v^{cluster}/4)^2$=324 for bursty traffic.

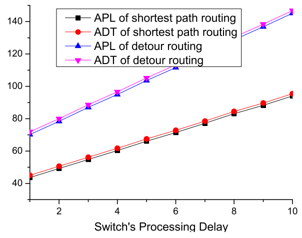
B. Simulation Results

Fig. 3 shows the performances of the two routing schemes under random traffic. As we can see, APLs of SRouting and DRouting remain constant, because APLs do not depend on the number of flows. APL of DRouting is greater than that of SRouting, because DRouting does not choose the shortest path. When the number of flows is small, the aggregate throughput increases almost linearly with the number of flows, and ADTs are very close to APLs; this is because the network has a very light traffic load and the main end-to-end delays come from processing delays and transmission delays, instead of queuing delays. When the number of flows is large, the increase of aggregate throughput becomes slower and slower and ADTs of both SRouting and DRouting increases almost linearly; this is because the network tends to be saturated and queuing delays become an important part of end-to-end delays. Notice that the upper bound of the aggregate throughput is the bisection bandwidth $B \approx N_v/4 = 1,314$. When the number of flows is 525,600, SRouting achieves an aggregate throughput of 1,142.6, which is 87.96% of the ideal maximum throughput. Thus, SRouting has good performances under random traffic. DRouting has a lower aggregate throughput because it has a larger maximum delivery time.

Fig. 4 shows the performances of the two routing schemes under bursty traffic. Though APL of DRouting is greater than that of SRouting, when the number of flows increases, ADT of DRouting is much smaller than that of SRouting. This is because DRouting experiences significantly less queuing delays by avoiding the congested link. When the number of flows increases, the aggregate throughput of SRouting is limited by the capacity of the congested link, 1; while the aggregate throughput of DRouting continues increasing significantly, because DRouting largely avoids the congested link and can use other links' capacities. Notice that, under bursty traffic, the aggregate throughput is also upper bounded

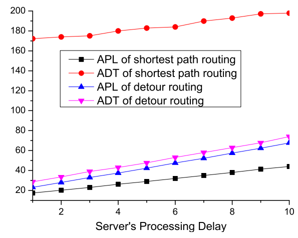


(a) APL and ADT vs. $d_{v,p}$.

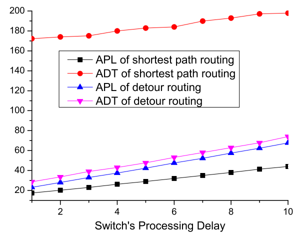


(b) APL and ADT vs. $d_{w,p}$.

Fig. 5. Influences of servers' and switches' processing delays (random traffic).



(a) APL and ADT vs. $d_{v,p}$.



(b) APL and ADT vs. $d_{w,p}$.

Fig. 6. Influences of servers' and switches' processing delays (bursty traffic).

by the sending and receiving rates of servers in the two clusters. If on average, only half of the servers are sending packets at each time unit, it indicates an upper bound on the maximum aggregate throughput of $N_v^{cluster}/2 = 36$. It takes some effort to calculate the true upper bound; we just want to show that this is the reason why the aggregate throughput of DRouting tends to approximate an upper bound around 23.5. We can see that DRouting helps both reducing ADTs and increasing the aggregate throughput. Thus, DRouting has good performances under bursty traffic.

The influences of $d_{v,p}$ and $d_{w,p}$ on APLs and ADTs are shown in Fig. 5 and Fig. 6. Under random traffic, ADTs of both SRouting and DRouting are close to their APLs, and all the values increase linearly with $d_{v,p}$ and $d_{w,p}$. Under bursty traffic, ADT of DRouting is close to its APL, and they increase linearly with $d_{v,p}$ and $d_{w,p}$; however, ADT of SRouting is much greater than its APL. The reason is that, under random traffic, the network congestion is quite low; processing delays and transmission delays account for a majority of ADTs, compared with queuing delays. However, under bursty traffic, if SRouting is adopted, the link that directly connects the source cluster and the destination cluster is heavily congested; the queuing delay accounts for a majority of ADT in SRouting. DRouting largely avoids the congested link; thus, its ADT is still close to APL. The results indicate that, for moderate traffic loads and efficient routing schemes where queuing delays are not significant, APLs characterize ADTs quite well. Thus, our unified path length definition is a useful metric to characterize end-to-end delays in general DCNs.

VII. Conclusion

We consider the tradeoffs in designing DCN architectures. We present a unified path length definition and a DCN power consumption model for general DCNs, to enable fair and meaningful comparisons. We identify a new class of DCNs, that can be regarded as dual-centric. We propose a novel DCN architecture, named FCell, which belongs to this class and serves as a good example of a tradeoff design in three aspects: between performances and power, between switch-centric and server-centric designs, and between scalability and flexibility. Two basic routing schemes are provided for FCell, and their performances under different traffic conditions are evaluated. By simulations, we verify that our unified path length definition is a useful metric to characterize end-to-end delays in general DCNs.

Acknowledgement

This work is supported in part by NSF grants CNS 149860, CNS 1461932, CNS 1460971, CNS 1439672, CNS 1301774, ECCS 1231461, ECCS 1128209, and CNS 1138963.

References

[1] D. Abts, M. Marty, P. Wells, and *et al*, "Energy proportional datacenter networks," in *ISCA*, 2010.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM Conf. on Data Comm.*, 2008.

[3] Y. Liao, D. Yin, and L. Gao, "Dpillar: Scalable dual-port server interconnection for data center networks," in *ICCCN*, 2010.

[4] D. Li and J. Wu, "On the design and analysis of data center network architectures for interconnecting dual-port servers," in *IEEE INFOCOM*, 2014.

[5] C. Guo, H. Wu, K. Tan, and *et al*, "Dcell: a scalable and fault-tolerant network structure for data centers," in *ACM SIGCOMM Conf. on Data Comm.*, 2008.

[6] D. Guo, T. Chen, D. Li, and *et al*, "Expandable and cost-effective network structures for data centers using dual-port servers," *IEEE Trans. on Computers*, vol. 62, no. 7, 2013.

[7] D. Li, C. Guo, H. Wu, and *et al*, "Ficonn: Using backup port for server interconnection in data centers," in *IEEE INFOCOM*, 2009.

[8] Y. Zhang and N. Ansari, "On architecture design, congestion notification, tcp incast and power consumption in data centers," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 1, 2013.

[9] J. H. Ahn, N. Binkert, A. Davis, and *et al*, "Hyperx: Topology, routing, and packaging of efficient large-scale networks," in *ACM/IEEE SC*, 2009.

[10] C. Guo, G. Lu, D. Li, and *et al*, "Bcube: a high performance, server-centric network architecture for modular data centers," in *ACM SIGCOMM Conf. on Data Comm.*

[11] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in *USENIX NSDI*, 2012.

[12] L. Gyarmati and T. A. Trinh, "Scafida: A scale-free network inspired data center architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 5, Oct. 2010.

[13] J.-Y. Shin, B. Wong, and E. G. Sirer, "Small-world datacenters," in *ACM SOCC*, 2011.

[14] Y. Liu, J. Muppala, M. Veeraraghavan, and *et al*, *Data Center Networks: Topologies, Architectures and Fault-Tolerance Characteristics*. Springer Briefs in Computer Science, 2013.

[15] L. Gyarmati and T. A. Trinh, "How can architecture help to reduce energy consumption in data center networking?" in *e-Energy*, 2010.

[16] A. Greenberg and D. A. Maltz, "What goes into a data center - sigmetrics 2009 tutorial." [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=81782

[17] G. Lu, C. Guo, Y. Li, and *et al*, "Serverswitch: A programmable and high performance platform for data center networks," in *USENIX NSDI*, 2011.

[18] L. Popa, S. Ratnasamy, G. Iannaccone, and *et al*, "A cost comparison of datacenter network architectures," in *Co-NEXT*, 2010.

[19] "Cisco nexus 2000 series fabric extenders data sheet." [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-2000-series-fabric-extenders/data_sheet_c78-507093.html

[20] "Intel gigabit et, et2, and ef multi-port server adapters." [Online]. Available: http://www.intel.com/content/dam/doc/product-brief/gigabit-et-et2-ef-multi-port-server-adapters-brief.pdf

[21] L. Bhuyan and D. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," *IEEE Trans. on Computers*, vol. C-33, no. 4, 1984.