DataCenter Topologies

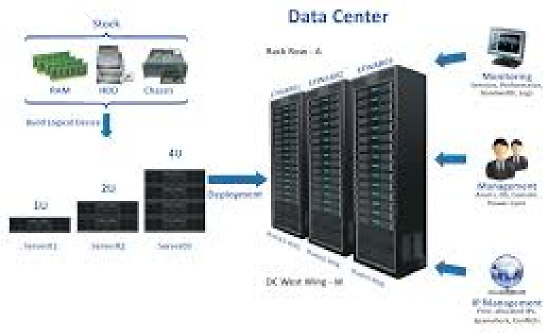
De silvestris Luca, 486652

April 4, 2019

Abstract

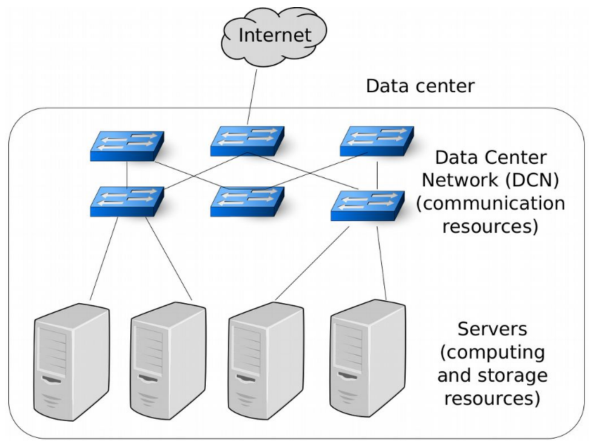
Revision of the definition of a datacenter and of the various topologies that are used starting from the generic definition of a DCN and from the traditional three-layer topology up to the Dcell architecture.

1 Datacenter and DCN



A datacenter is a set of physical infrastructures necessary to support a cloud computing service 24 hours a day, every day of the year. The entire infrastructure is located in a room or building. Because IT operations are critical to business continuity, the data center includes redundant or backup components and infrastructures for power, data communication connections, environmental controls (such as air conditioning, fire suppression) and miscellaneous safety devices. A large data center is an industrial-scale operation that uses the same amount of electricity as a small city like a real city center. A datacenter is usually composed of several resources divided into: processing and storage resources (hard disks, SSDs, servers ...); Communication resource (switch, router, traffic balancing, firewall ...) and infrastructure resources (racks, cooling systems and energy supply). The Telecommunications Industry Association's data center telecommunications infrastructure standard specifies the minimum requirements for the data center and computer room telecommunications infrastructure, including single-tenant business data centers and multi-tenant hosting data centers. The topologies proposed in this document must be applicable to any data center of any size. As seen in the datacenter there is a pool of resources (computational, storage, network) but how do these resources relate to each other? These are interconnected via a communication network. Data Center Network (DCN) plays a key role in a data center as it connects to data center resources together. DCNs need to be scalable and efficient to connect dozens or even hundreds of servers to handle the growing demands of cloud computing. Today's data centers are bound by the interconnection network. The various datacenter topologies will be discussed in detail in the following sections starting from the more traditional ones up to modern and superior solutions for performance and scalability.

2 Traditional DCN and Problems

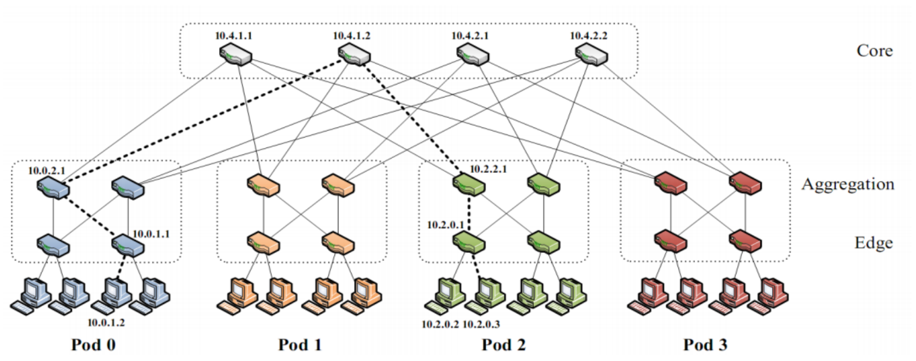

Internet
Data center
Data Center Network (DCN) (communication resources)
Servers (computing and storage resources)

In the traditional topology that can be found at two or three tiers. In the three-tier DCN there are 20 to 40 servers connected per rack and each server is connected to two access switches that connect to an appropriate aggregation switch in this way the aggregate level switches interconnect multiple access level switches. At the aggregate level then the switches are connected together by the switches of the core switch that are responsible for the connection of the data center to the Internet. In this architecture, each Layer 2 domain typically limited to a few hundred servers to limit broadcast. Major problems faced by the three-tier architecture include, scalability, fault tolerance, energy efficiency, and cross-sectional bandwidth. The three-tier architecture uses enterprise-level network devices at the higher layers of topology that are very expensive and power hungry. Among the various problems encountered in traditional topology we can highlight:

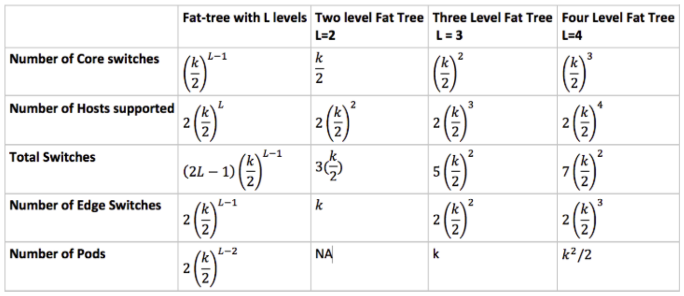
- **Cannot handle the growing demand for cloud computing.**
- **Little robustness**
- **Aggregation switching errors eliminate entire racks**
- **Reduced performance** like: Oversubscription = maxThroughput / (worst Case Throughput)
- **high cost**: 7K for 48-port Gigabit switches 700 K for 128-port 10-port Gigabit switches
- **Single point of failure**

To solve these problems it was decided to adopt a special example of Clos topology by basing the data centers on a **fat-three topology**. In this way it is possible to have a datacenter that offers full bandwidth, is fault tolerant and has a lower maintenance cost

3 Fat-three DCN



The fat-tree DCN architecture manages the problem of overwriting and bandwidth known from the traditional three-layer DCN architecture. The fat-tree DCN architecture actually has an almost identical length in each bisection and each level has the same bandwidth as the aggregation. Scalability has also been improved with this architecture, since the k-port switch supports $k^3/4$ servers but the maximum number of pods is equal to the number of ports in each switch. The fat-tree topology was introduced by Charles E. Leirson in 1985 in which the branches (vertices) near the central level are ”thicker”, which means that they allow more bandwidth than the base axis using more switches. This can also be seen and studied by the simple online fat-tree viewer (an example is http: // liulonnie .Net / ftree-vis /) where you can go to increase and decrease the depth of the tree and the ports for the switches and see the effects. The network elements in the fat-tree topology also follow the hierarchical organization of the network switches in the three levels, unchanged with respect to the traditional DCN: access level, aggregation and core. The number of network switches, however, is much greater than the three-layer DCN and offers a 1: 1 overwrite ratio and full-width bandwidth. The table below shows the number of switches for each level, assuming in fact that a switch has ”k” ports and the number of levels is ”L”, the first column of the following table provides the generalized way to calculate various parameters. The next three columns show various numbers for levels 2,3 and 4. Based on the number of hosts we need to support, it is possible to calculate how many tree levels are needed.



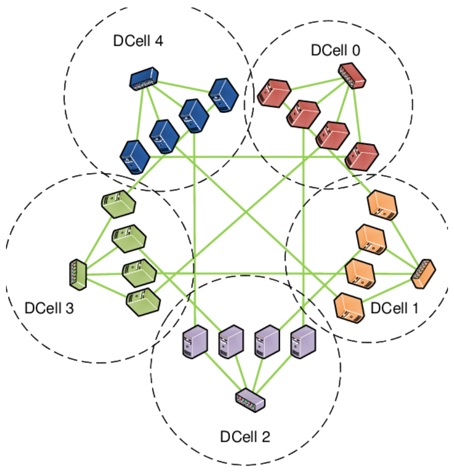
	Fat-tree with L levels	Two level Fat Tree L=2	Three Level Fat Tree L = 3	Four Level Fat Tree L=4
Number of Core switches	$\left(\frac{k}{2}\right)^{L-1}$	$\frac{k}{2}$	$\left(\frac{k}{2}\right)^{2}$	$\left(\frac{k}{2}\right)^{3}$
Number of Hosts supported	$2\left(\frac{k}{2}\right)^{L}$	$2\left(\frac{k}{2}\right)^{2}$	$2\left(\frac{k}{2}\right)^{3}$	$2\left(\frac{k}{2}\right)^{4}$
Total Switches	$(2L-1)\left(\frac{k}{2}\right)^{L-1}$	$3(\frac{k}{2})$	$5\left(\frac{k}{2}\right)^{2}$	$7\left(\frac{k}{2}\right)^{2}$
Number of Edge Switches	$2\left(\frac{k}{2}\right)^{L-1}$	k	$2\left(\frac{k}{2}\right)^{2}$	$2\left(\frac{k}{2}\right)^{3}$
Number of Pods	$2\left(\frac{k}{2}\right)^{L-2}$	NA	k	$k^2/2$

As seen in the internal levels, we have a lot of redundancy due to the large number of connections between the switches. Now let’s look at addressing and routing in a fat-tree: Layer-2 addressing has the great advantage of being able to have a single LAN but there is a great disadvantage of very large switch tables and above all the large bradcast traffic; Layer-3 addressing instead has one subnet for VLAN but has the disadvantage of having a high number of routers and switchers and to perform multipath routing and have more efficiency you should have more expensive switchers and given the large number of switches present spending may not be sustainable.
Another important point when the topology of the fat-tree is the fault tolerance of the data center and above all between the levels. In this scheme, each network switch maintains the BFD (Bidirectional Forwarding Detection) session with each of its neighbors to determine when a nearby link or switch does not work. There are various types of failures that can occur:

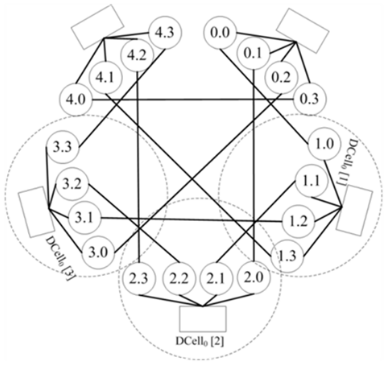
- **Failure between upper layer and core switches**: in Outgoing traffic, local routing table marks the affected link as unavailable and chooses another core switch; on Incoming inter-pod traffic, core switch broadcasts a tag to upper switches directly connected signifying its inability to carry traffic to that entire pod, then upper switches avoid that core switch when assigning flows destined to that pod.

- **Failure between lower and upper layer switches**:on Outgoing inter- and intra pod traffic from lower-layer, the local flow classifier sets the cost to infinity, does not assign it any new flows and chooses another upper layer switch; on Intra-pod traffic using upper layer switch as intermediary.

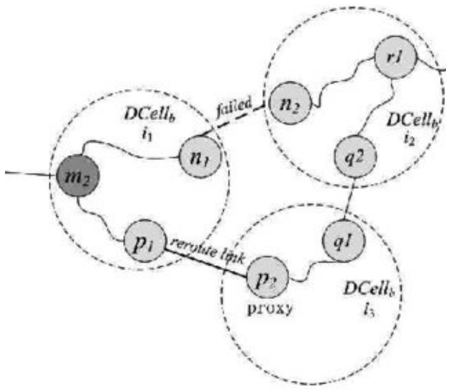
4 DCell



The DCell-based DCN solution has four components that work in concert to address the three challenges. They are the DCell scalable network structure, efficient and distributed routing algorithm that exploits the DCell structure, fault-tolerant routing that addresses various types of failures such as link/server/rack failures, and an incremental upgrade scheme that allows for gradual expansion of the DCN size.DCell uses servers equipped with multiple network ports and mini-switches to construct its recursively defined architecture. In DCell, a server is connected to several other servers and a mini-switch via communication links, which are assumed to be bidirectional. A high-level DCell is constructed from low-level DCells.Following the figure we note that each server is connected to a single mini-switch and each server is connected to multiple servers.The strength of this DCN is the recursion in fact DCell follows a recursively constructed hierarchy of cells. A cell0 is the base unit and the default block of the DC topology arranged on multiple levels, where a higher level cell contains more cells than the lower level. The cell0 is a constituent element of the DCell topology, which contains a server and a product network switch. The network switch is used only to connect the server within a cell0. A cell1 contains k = n + 1 cell0 cells, and similarly a cell2 contains k * n + 1 dcell1.



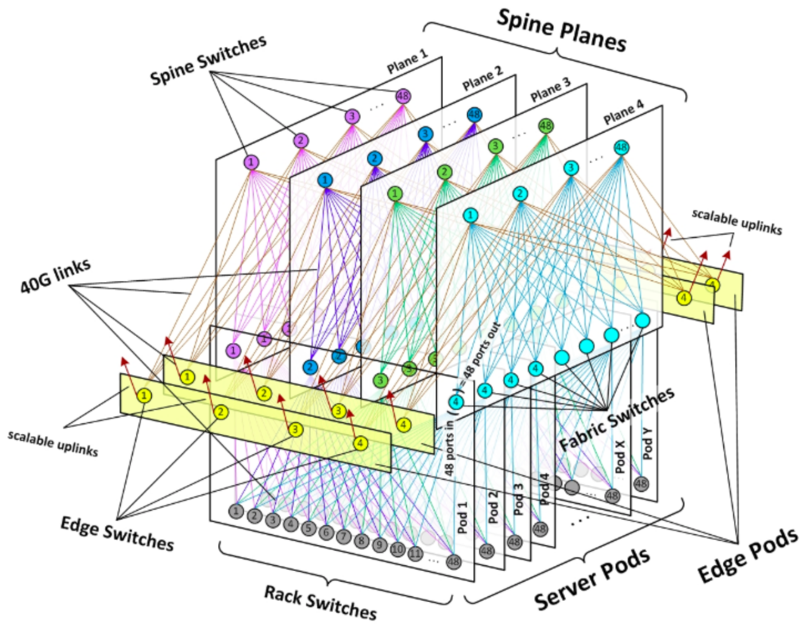
As far as architecture scalability is concerned, DCell is a highly scalable architecture in fact a four-tier DCell with only six servers in cell0 can host about 3.26 million servers. However, Dcell also has problems because the bandwidth of the cross section and the latency of the network are an important problem of this architecture.The DFR protocol is used for protocols and fault tolerance in DCell networks. DFR is a distributed and fault-tolerant routing protocol for DCell networks without global connection status. DFR manages three types of errors: server error, rack error and connection error. Rack failure occurs when all machines are in "rack failed" (for example, two to switch or a power outage). Connection failure is basic since all faults cause a connection error. Therefore, the management of connection errors is a fundamental part of DFR. DFR uses three local redirection techniques, local link-state and jumps back to resolve the link error, server error and rack error, respectively.



```
DFR(pkt) /*pkt is the received packet*/
if ($pkt.dst == self.uid$) { deliver($pkt$); return; }
if ($self.uid == pkt.proxy$) $pkt.proxy$ =NULL;
if ($pkt.proxy$!=NULL) $dcn\_dst = pkt.proxy$;
else $dcn\_dst = pkt.dst$;
$path$ = DCellRouting($self.uid, dcn\_dst$);
$(n_1, n_2)$ = FirstLink($path, b$);
if ($(n_1, n_2)$ ==NULL) $dij\_dst = dcn\_dst$;
else
   if ($(n_1, n_2)$ fails) goto local-reroute;
   else $dij\_dst = n_2$;
$next\_hop$ = DijkstraRouting($pkt, dij\_dst$);
if ($next\_hop$!=NULL)
   forward $pkt$ to $next\_hop$ and return;
else if ($self.uid$ and $pkt.dst$ are in a same $DCell_b$))
      drop $pkt$ and return;
local-reroute:
$pkt.retry$ −−;
if ($pkt.retry$ == 0){drop($pkt$); return;}
$pkt.proxy$ = SelectProxy($uid, (n_1, n_2)$))
return DFR($pkt$);
```

5 Conclusion

As seen, it is possible to identify different topological solutions for a datacenter. However, it is difficult to say which of these can be considered the best because everything depends on which of the advantages of a topology are preferable over others and which disadvantages can be sustained without weighing too much on the management of the data center or without having future problems. However, opting for one architecture over another is not enough to guarantee efficiency and cutting-edge datacenter, since a good topology is nothing compared to the use of efficient routing algorithms. Not only that, assuming you have a good datacenter with a good topology and a decent Routing algorithm (which would still require smarter and more expensive switches but in a datacenter this would require a very large expense) what would happen without fault management? Without a good tolerance to breakdowns you could even have a very good system that would not have the possibility of using its potential due to various faults. So when you need to see if a datcenter is efficient you have to take into account several factors. In addition, data loggers and above all topologies are always evolving and there are various cases of large companies that have introduced noteworthy improvements. One of the most advanced companies is Facebook. Facebook Introducing data center fabric, the next-generation Facebook data center network going from a data center networks built using clusters to this ”top-down” approach with greater gradual scalability that overcomes the limits of clusters topology. The physical structure is instead shown in the following figure.



References

[1] *Guida al data center: cos'è, come funziona, classificazione e vantaggi.* Online: https://www.zerounoweb.it/techtarget/searchdatacenter/guida-al-data-center-cos-e-come-funziona-classificazione-e-vantaggi/

[2] *Data center network architectures.* Online: https://en.wikipedia.org/wiki/Data_center_network_architectures

[3] Hakim Weatherspoon, *Data Center Network Topologies: FatTree.* CS 5413: *High Performance Systems and Networking*, Cornell University, 2014. Online: https://www.cs.cornell.edu/courses/cs5413/2014fa/lectures/08-fattree.pdf

[4] *Demystifying DCN Topologies: Clos/Fat Trees.* Online: https://packetpushers.net/demystifying-dcn-topologies-clos-fat-trees-part2

[5] *Fat Tree Visualizer.* Online: http://liulonnie.net/ftree-vis/

[6] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi†, Yongguang Zhang, Songwu Lu *DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers*,Microsoft Research Asia. Online: http://www.sigcomm.org/sites/default/files/ccr/papers/2008/October/1402946-1402968.pdf