MODULE 6: DATA CENTER NETWORKS

DATA CENTER

- The idea of a data center is not new.
- Densely packed racks of high-powered computers and storage have been around for decades, originally providing environments for mainframes.
- These eventually gave way to minicomputers and then to the servers that we know and deal with today.
- Over time the density of those servers and the arrays of storage that served them have made it possible to host a tremendous amount of compute power in a single room or pod.
- Today's data centers hold thousands, even tens of thousands, of physical servers.
- These data centers can be segregated into the following three categories:

1. **Private single-tenant:**
   - Individual organizations that maintain their own data centers belong in this category.
   - The data center is for the private use of the organization, and there is only the one organization or *tenant* using the data center.
2. **Private multitenant:**
   - Organizations that provide *specialized* data center services on behalf of other client organizations belong in this category.
   - IBM and EDS (now HP) are examples of companies that host such data centers.
   - These centers are built and maintained by the organization providing the service, and multiple clients store data there, suggesting the term *multitenant*.
   - These data centers are private because they offer their services contractually to specific clients.

3. Public multitenant:

- Organizations that provide *generalized* data center services to any individual or organization belong in this category.
- Examples of companies that provide these services include Google and Amazon.
- These data centers offer their services to the public.
- Anybody, whether individuals or organizations, who wants to use these services may access them via the web.

- In the past, these data centers were often hosted such that they could be reached only through private communication channels.
- However, in recent years, these data centers have begun to be designed to be accessible through the Internet.
- These types of data centers are often referred to as residing in *the cloud*.
- Three subcategories of clouds are in common use: *public* clouds, *private* clouds, and *hybrid* clouds.
- In a **public cloud**, a service provider or other large entity makes services available to the general public over the Internet.
- Such services may include a wide variety of applications or storage services, among other things.
- Examples of public cloud offerings include *Microsoft Azure Services Platform* and *Amazon Elastic Compute Cloud*.
- In a **private cloud**, a set of server and network resources is assigned to one tenant exclusively and protected behind a firewall specific to that tenant.

- The physical resources of the cloud are owned and maintained by the cloud provider, which may be a distinct entity from the tenant.
- The physical infrastructure may be managed using a product such as VMware’s vCloud.
- Amazon Web Services is an example of the way a private cloud may also be hosted by a third party (i.e., Amazon).
- An increasingly popular model is the **hybrid cloud**, where part of the cloud runs on resources dedicated to a single tenant, but other parts reside on resources that are shared with other tenants.
- This is particularly beneficial when the shared resources may be acquired and released dynamically as demand grows and declines. E.g. Verizon’s *cloud bursting*

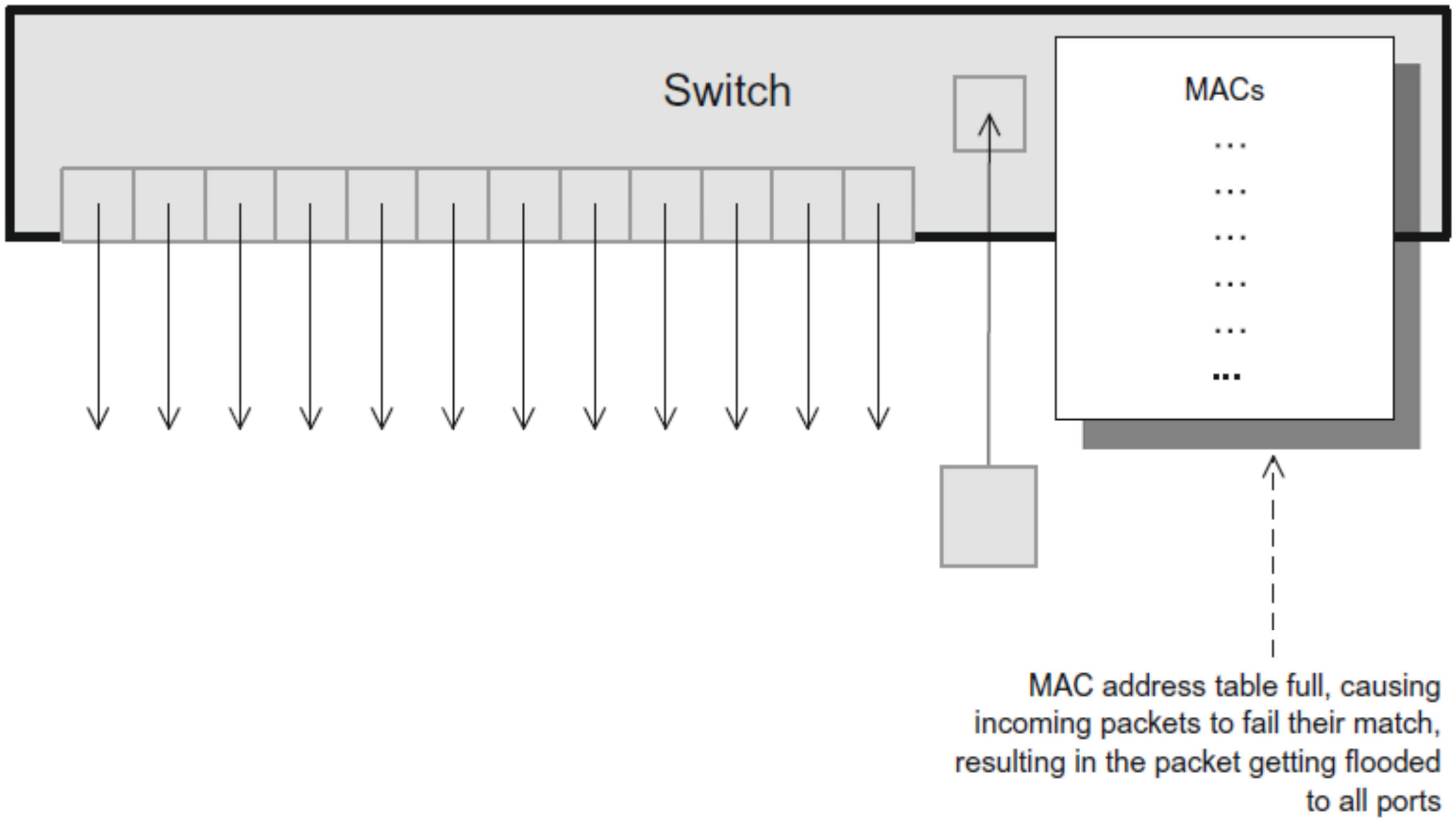
Data Center Need:

1. **Overcoming Current Network Limitations:**

- The potential to easily ignite and tear down VMs is a radical departure from the physical world that network managers have traditionally faced.
- Networking protocols were coupled with physical ports, which is *not* the case moving forward.
- The dynamic nature and sheer number of VMs in the data center have placed demands on the capacity of network components that were earlier thought to be safe from such pressures.
- In particular, these areas include *MAC address table size*, *number of VLANs*, and *spanning tree*.

1.1 MAC Address Explosion

- In switches and routers, the device uses MAC address table to quickly determine the port or interface out of which the device should forward the packet.
- For speed, this table is implemented in hardware.
- As such, it has a physical limit to its size.
- Naturally, device vendors will determine the maximum number of entries to hold in the MAC table based on controlling their own costs, at the same time providing an adequate number of entries for network demands.
- If the table is too large, the vendor will have spent too much money on creating the device.
- If the table is too small, the customer will experience the problems of MAC address explosion.


Switch
MACs
...
MAC address table full, causing incoming packets to fail their match, resulting in the packet getting flooded to all ports

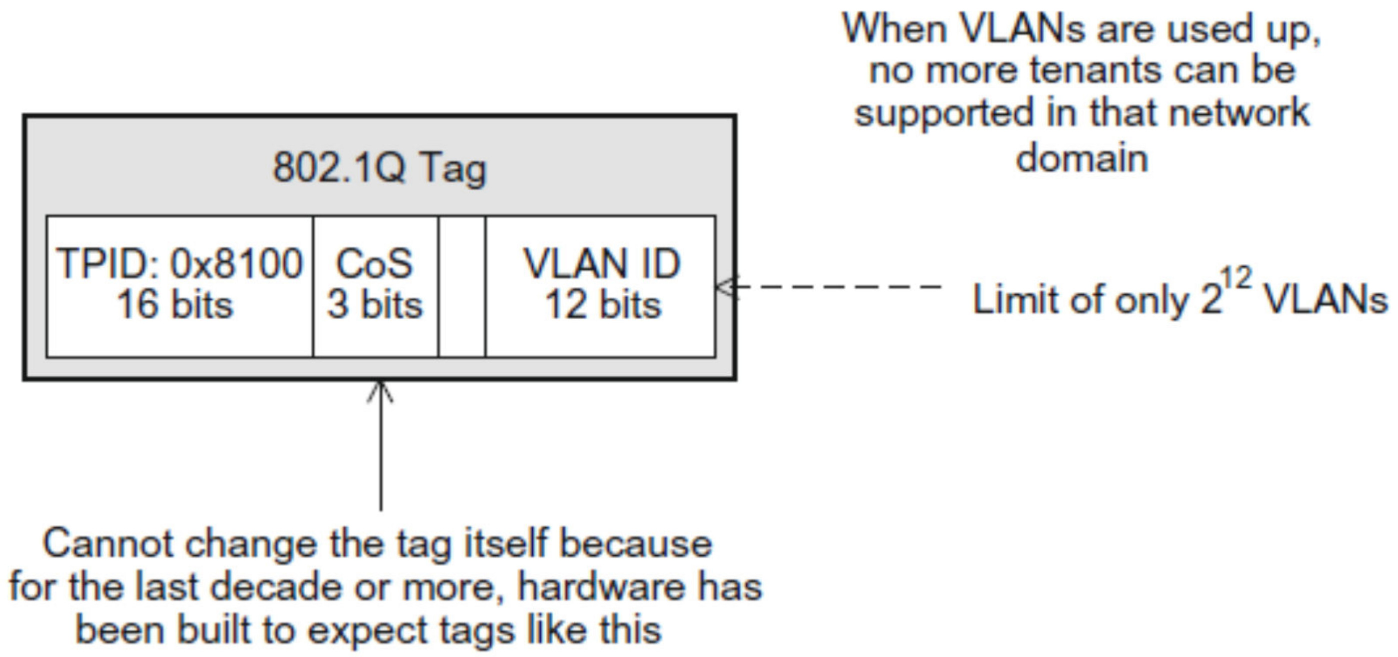
- Networks in the past had manageable limits on the maximum number of MAC addresses that would need to be in the MAC address table at any given time.
- This was partly attributed to physical limitations of data centers and the number of servers that would be part of a single layer two network. It is also affected by the physical layout of the network.
- In the past, physically separate layer two networks remained logically separate.
- Now, with network virtualization and the use of Ethernet technology across WANs, the layer two networks are being stretched geographically as never before.
- With server virtualization, the number of servers possible in a single layer two network has increased dramatically.
- With numerous virtual network interface cards (NICs) on each virtual server, this problem of a skyrocketing number of MAC addresses is exacerbated.
- Layer two switches are designed to handle the case of a MAC table miss by flooding the frame out all ports except the one on which it arrived, as shown in Figure.

- This has the benefit of ensuring that the frame reaches its destination if that destination exists on the layer two network.
- Under normal circumstances, when the destination receives that initial frame, this prompts a response from the destination.
- Upon receipt of the response, the switch is able to learn the port on which that MAC address is located and populates its MAC table accordingly.
- This scheme works well unless the MAC table is full, in which case it cannot learn the address.
- Thus, frames sent to that destination continue to be flooded.
- This is a very inefficient use of bandwidth and can have significant negative impact on performance.

1.2 Number of VLANs

- The introduction of data centers created the need to segregate traffic so as to ensure security and separation of the various tenant networks' traffic.
- The technology responsible for providing this separation is VLANs.
- As long as data centers remained single-tenant networks, the maximum number of 4096 VLANs seemed more than sufficient.
- To have expanded this 12 bit field unnecessarily was not wise, since different tables in memory and in the ASICs had to be large enough to accommodate the maximum number.
- Thus, to have made the maximum larger than 4096 had a definite downside at the time the design work was performed.
- When data centers continued to expand, however, especially with multi-tenancy and server virtualization, the number of VLANs required could easily exceed 4096.

- When there are no more available VLANs, the ability to share the physical resources among the tenants quickly becomes complex.
- Since the number of bits allocated to hold the VLAN is fixed in size and for years hardware has been built that depends on that specific size, it is nontrivial to expand the size of this field to accommodate more VLANs.
- Thus, some other solution is needed to overcome this limitation.


When VLANs are used up, no more tenants can be supported in that network domain
802.1Q Tag
TPID: 0x8100 16 bits
CoS 3 bits
VLAN ID 12 bits
Limit of only 2^{12} VLANs
Cannot change the tag itself because for the last decade or more, hardware has been built to expect tags like this

1.3 **Spanning Tree**

- In the early years of Ethernet, bridges were built as *transparent* devices capable of forwarding packets from one segment to another without explicit configuration of forwarding tables by an operator.
- The bridges learned forwarding tables by observing the traffic being forwarded through them.
- Distributed intelligence in the devices was capable of collectively determining whether there were loops in the network and truncating those loops so as to prohibit issues such as broadcast storms from bringing down the network.
- This was accomplished by the bridges or switches collectively communicating with one another to create a *spanning tree*, which enforces a loop-free hierarchical structure on the network in situations where physical loops do exist. This spanning tree was then calculated using the *Spanning Tree Protocol*

- The process of determining this spanning tree is called *convergence*, and in the early days it would take some time (dozens of seconds) for the spanning tree to converge after a physical change to the networking devices and their links had taken place.
- Over time, through improvements to STP, the process of converging the spanning tree increased in speed.
- For example, with the improvements in recent versions of the standard, convergence times for conventionally large networks have decreased dramatically.
- Despite these improvements, STP still leaves perfectly functional links unused and with frames being forwarded to the root of the spanning tree, which is certainly not universally the optimal path.
- Data centers need more *cross-sectional* bandwidth.
- By this we mean using the most efficient path between any two nodes without imposing an artificial hierarchy in the traffic patterns.
- The fact that the fluidity of data center virtualization has increased the frequency of changes and disruptions, thus requiring re-convergence to occur more often, has only added to the inefficiency of STP in the data center.
- STP was simply not built for the modern data center world of virtual switches and ephemeral MAC addresses.

2. Adding, Moving, and Deleting Resources

- Today's virtualized data centers are able to make changes much more quickly than in the past.
- Networks need to adapt in order to keep pace with the virtualization capabilities of servers and storage.
- Speed and automation are of critical importance when it comes to handling the rapid changes demanded by virtualized servers and storage.
- It is clearly important that changes to networking infrastructure take place at the same rate as is possible with servers and storage.
- Legacy networks require days or weeks for significant changes to VLANs and other network plumbing needs.
- A large part of the reason for this time requirement is that the repercussions of a mistaken network change can easily impact all the data center resources, including not only networking but also its compute and storage components.
- Coordinating changes with all these disparate departments is unavoidably complex.
- These changes need to have the ability to be automated so that changes that must happen immediately can take place without human intervention.
- Legacy protocols were designed to react *after* the newly ignited service comes online.
- With SDN one can use the foreknowledge that a new service is about to be initiated and proactively allocate the network capacity it will require.

3. Failure Recovery

- The size and scale of data centers today make recovery from failure a complex task, and the ramifications of poor recovery decisions are only magnified as scale grows.
- Determinism, predictability, and optimal re-configuration are among the most important recovery-related considerations.
- With the distributed intelligence of today’s networks, recovery from failures results in unpredictable behavior.
- It is desirable that the network move to a known state, given a particular failure.
- Distributed protocols make this difficult to do.
- A complete view of the network is required to make the recovery process yield the best result.

4. Multitenancy

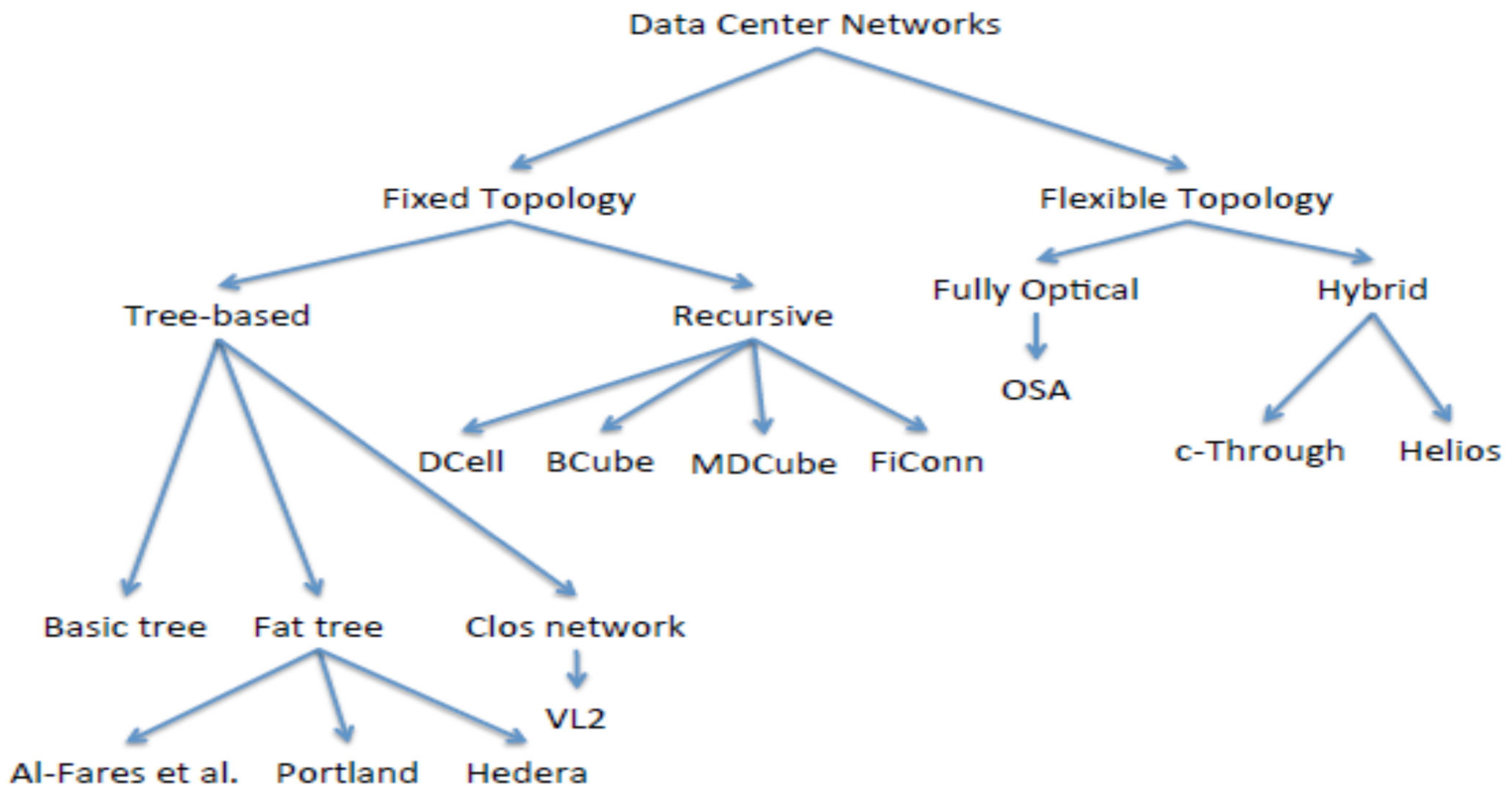
- Data center consolidation has resulted in more and more clients occupying the same set of servers, storage, and network.
- The challenge is to keep those individual clients separated and insulated from each other and to utilize network bandwidth efficiently.
- In a large multitenant environment, it is necessary to keep separate the resources belonging to each client.
- For servers this could mean not mixing clients' virtual machines on the same physical server.
- For networks it could mean segregation of traffic using a technology that ensures that packets from two distinct tenants remain insulated from one another.
- This is needed not only for the obvious security reasons but also for QoS and other service guarantees.

5. Traffic Engineering and Path Efficiency

- As data centers have grown, so has the need to make better use of the resources being shared by all the applications and processes using the physical infrastructure.
- In the past this was small issue because overprovisioning could make up for inefficiencies.
- But with the current scale of data centers, it is imperative to optimally utilize the capacity of the network.
- This entails proper use of monitoring and measurement tools for more informed calculation of the paths that network traffic takes as it makes its way through the physical network.
- To understand traffic loads and take the appropriate action, the traffic data must be monitored and measured.
- Gathering traffic intelligence has often been a luxury for networks.
- But with the need to make the most efficient use of the links and bandwidth available, this task has become an imperative.

- State-of-the-art methods of calculating routes use link-state technology, which provides the network topology for the surrounding network devices that are part of the same autonomous system (AS).
- This allows path computation to determine the *shortest path*.
- However, the shortest path as defined by traditional technology is not always the optimal path, since it does not take into consideration dynamic factors such as traffic load.
- One of the reasons for the increasing attention on traffic engineering and path efficiency in the data center has been the rise of *East-West* traffic relative to *North-South* traffic.
- The traffic types are defined as : East-West *traffic is composed of packets sent by one host in a data center to another host in that same data center. Analogously, North-South traffic is traffic entering (leaving) the data center from (to) the outside world.*
- Facebook provides a good example of what is driving the growth in East-West traffic.
- When you bring up your newsfeed page on Facebook, it has to pull in a multitude of data about different people and events in order to build the webpage and send it to you.
- That data resides throughout the data center.
- The server that is building and sending you your Facebook newsfeed page has to pull data from servers in other parts of the data center.
- That East-West traffic is at least as large as the North-South data (i.e. the final page it sends you) and in fact can be larger if you account for the fact that servers are moving data around even when no one is requesting it (via replication, staging data in different data centers, and so on).

DATA CENTER NETWORK TOPOLOGIES



- All the topologies discussed in this section can be classified into two categories: fixed and flexible.
- If the network topology cannot be modified after the network is deployed, we classify it as a fixed architecture; otherwise it is a flexible architecture.
- notations used:

  n: The number of ports in a switch in an architecture.

  k: The number of ports in a server in an architecture.

  N: The total number of servers inside a data center network.
- It should be noted that n and k may vary according to the position of the node.

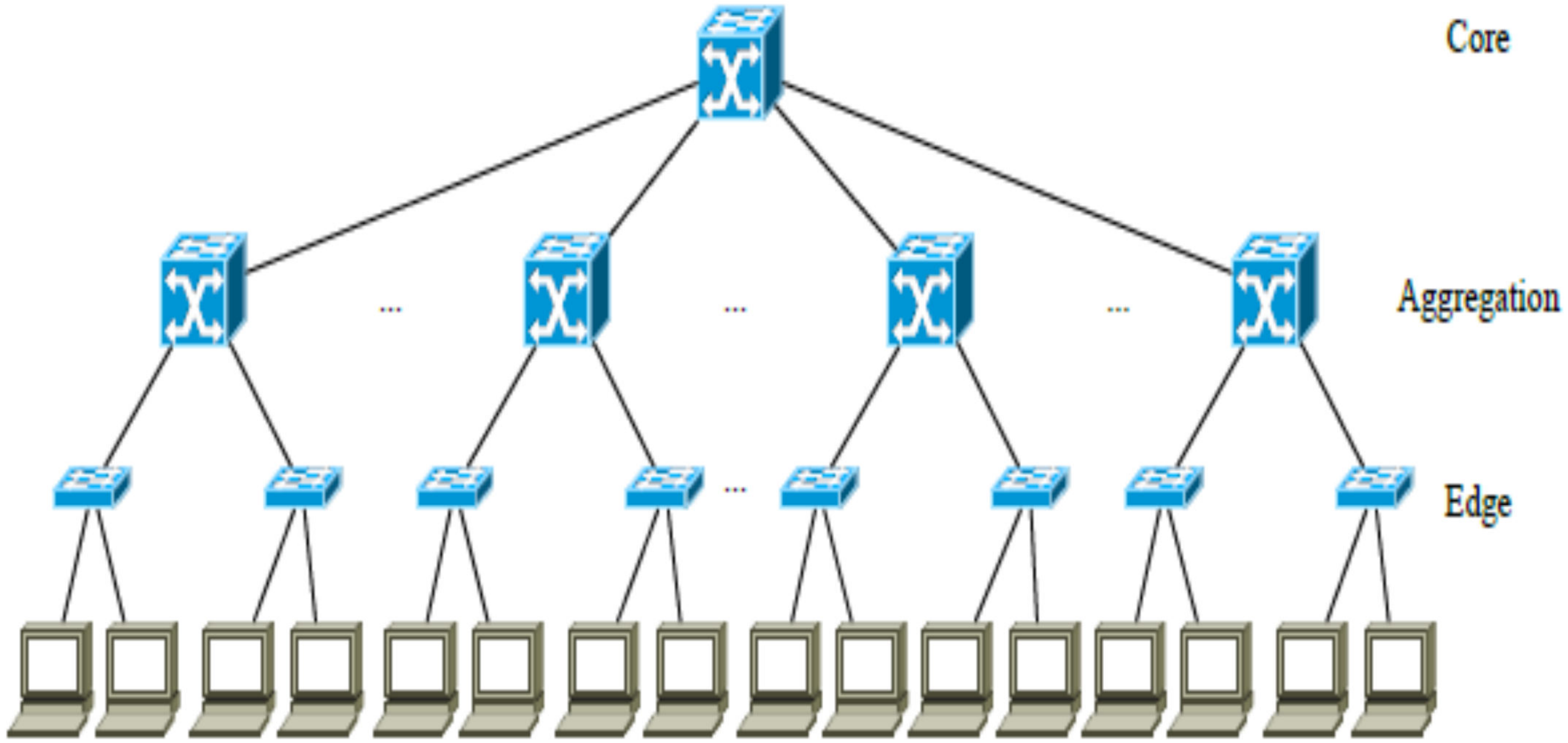
A. Fixed Architectures

1. Tree-based Topologies :

- Standard tree based architectures and their variants are widely used in designing data center networks.

1.1 Basic Tree:

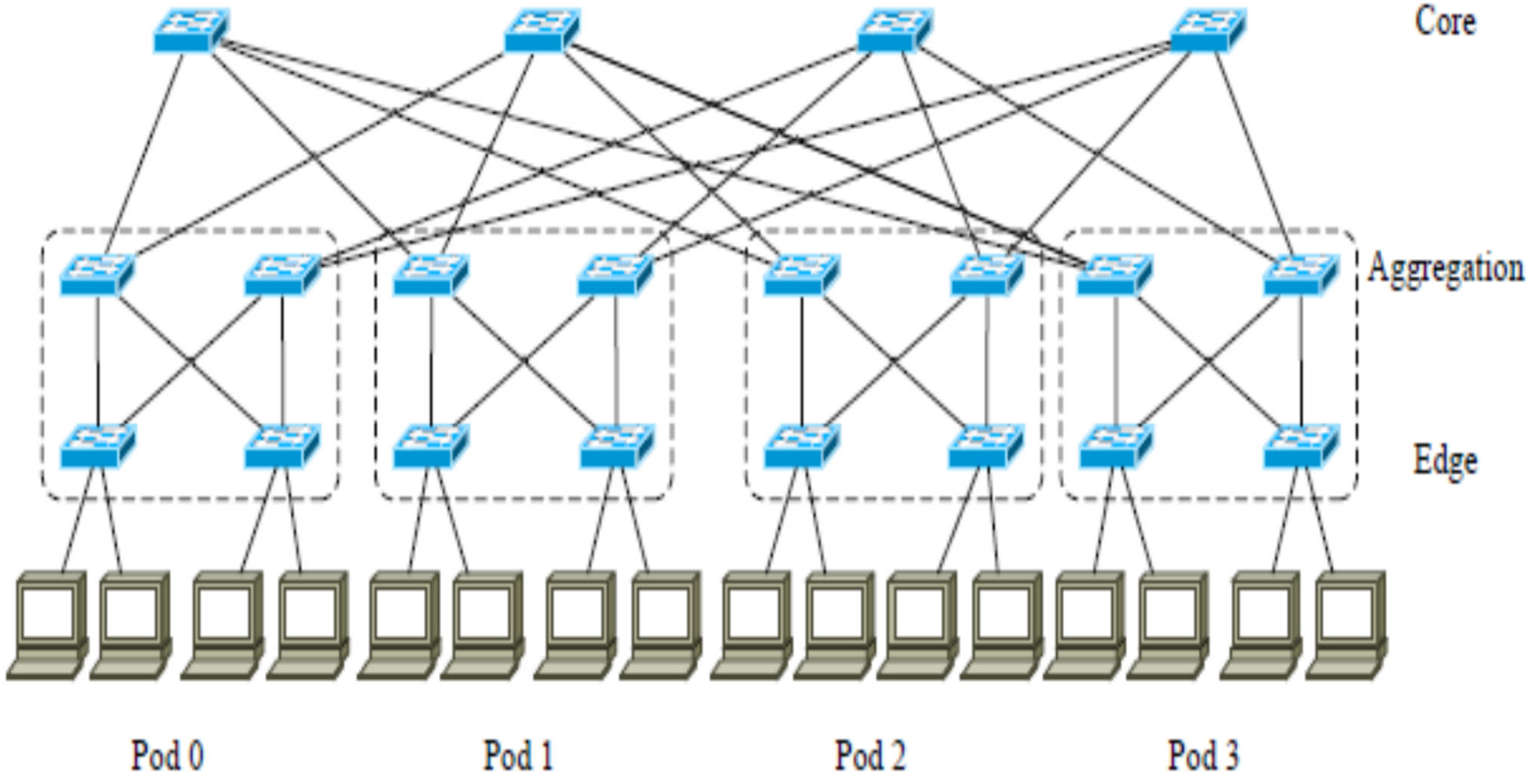
- Basic tree topologies consist of either two or three levels of switches/routers, with the servers as leaves.
- In a 3-level topology, there is a core tier at the root of the tree, an aggregation tier in the middle, and an edge tier of switches connecting to the servers.
- In a 2-level topology, there is no aggregation tier.
- There are no links between switches in the same tier, or in nonadjacent tiers.
- In a basic tree topology, the higher-tier switches need to support data communication among a large number of servers.
- Thus switches with higher performance and reliability are required in these tiers.
- The number of servers in a tree architecture is limited by the numbers of ports on the switches



3-level tree topology

1.2 Fat Tree:

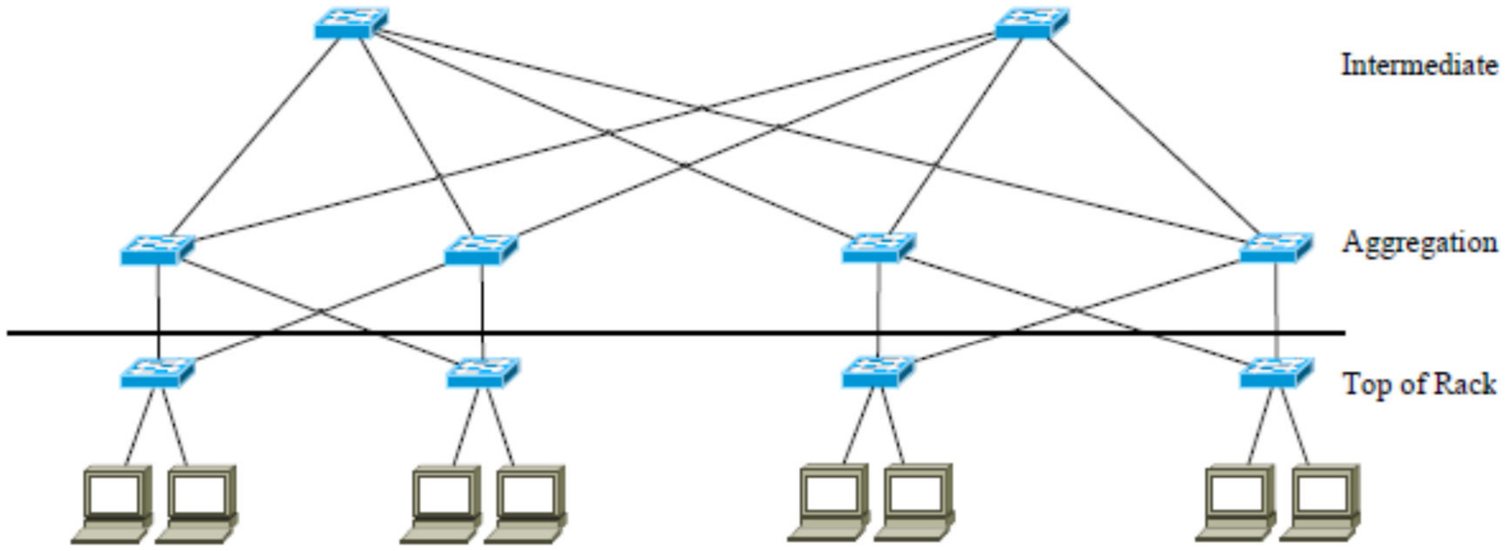
- Fat tree is an extended version of a tree topology.
- Fat trees have been applied as a topology for data centers by several researchers.
- A fat tree is a network based on a complete binary tree.
- Each n-port switch in the edge tier is connected to n/2 servers. The remaining n/2 ports are connected to n/2 switches in the aggregation level.
- The n/2 aggregation-level switches, the n/2 edge-level switches and the servers connected to the edge switches form a basic cell of a fat tree, which is called a pod.
- In the core level, there are $(n/2)^2$ n-port switches, each one connecting to each of the n pods.
- Figure shows a fat tree topology with n = 4.
- Unlike in basic tree topology, all the 3 levels use the same kind of switches.
- High-performance switches are not necessary in the aggregate and core levels.
- The maximum number of servers in a fat tree with n-port switches is $\frac{(n)^3}{4}$.



A 3-level Fat Tree Topology

- **1.3 Clos Network/ multi-rooted tree :**
- Greenberg et al. proposed a data center network architecture using a Clos Network topology.
- A Clos Network is also a multi-rooted tree.
- When deployed in data center networks, a Clos Network usually consists of three levels of switches: the ToR switches directly connected to servers, the aggregation switches connected to the ToR switches, and the intermediate switches connected to the aggregation switches.
- These three levels of switches are termed “input”, “middle”, and “output” switches in the original Clos terminology.
- The number of the switches is determined by the number of ports on the intermediate switches and aggregation switches.
- If each of these switches has n ports, there will be n aggregation switches and n/2 intermediate switches.
- There is exactly one link between each intermediate switch and each aggregation switch.
- The remaining n/2 ports on each aggregation switch is connected to n/2 different ToR switches.
- Each of the ToR switches is connected to 2 different aggregation switches, and the remaining ports on the ToR switches are connected to servers.

- There are $\frac{n^2}{4}$ ToR switches because to each pair of aggregation switches, n/2 ToR (Top of Rack) switches are connected.
- While intermediate switches and aggregation switches must have the same number of ports, the number of ports on a ToR switch is not limited.
- If n ToR ports on each TOR switch are connected to servers, there will be $\frac{n^2}{4}\times$n ToR servers in the network.
- Figure shows a Clos Network with n = 4, n2/4 ToR = 4.



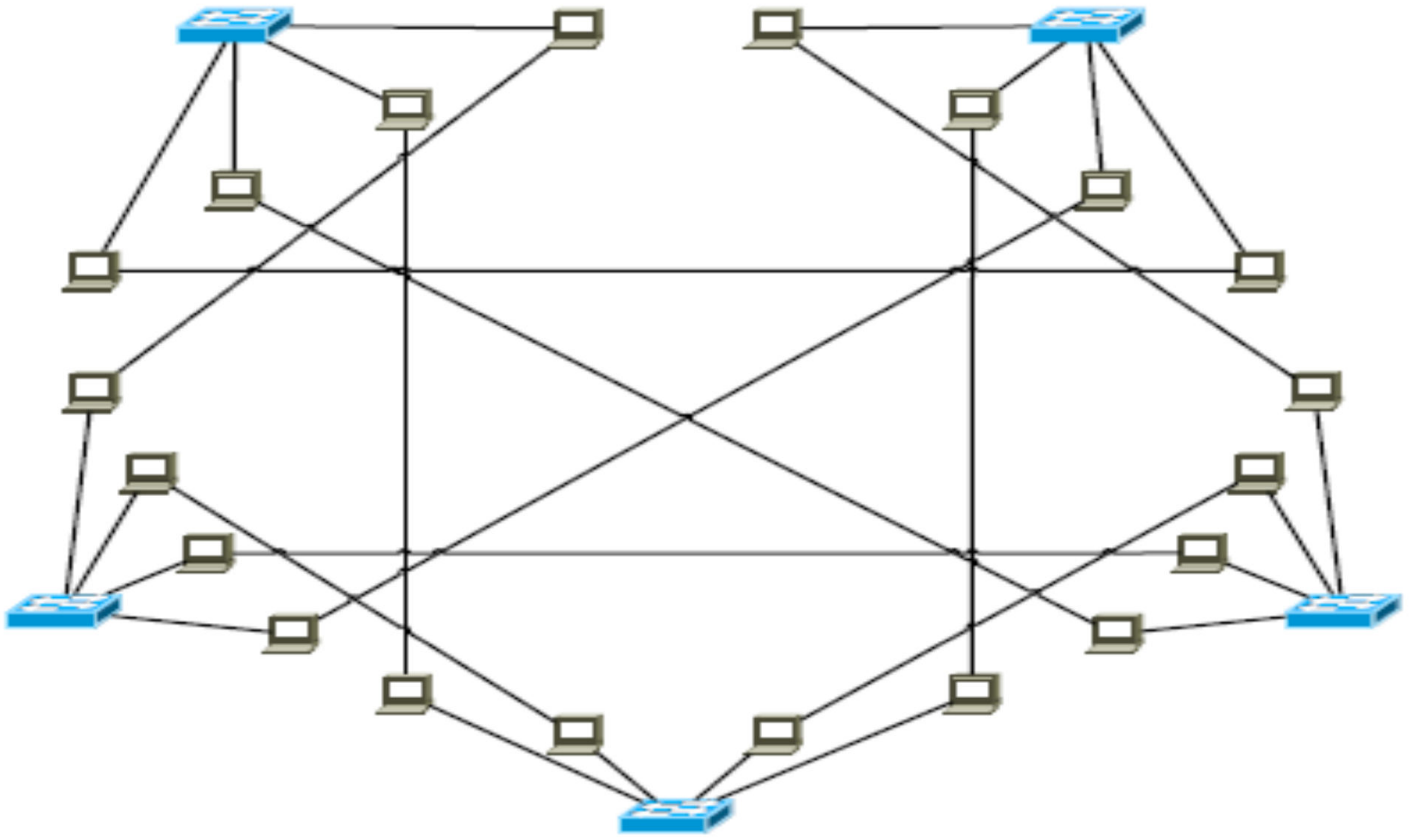
A Clos Network Topology

Fixed Architecture 2: Recursive Topologies

- Tree-based topologies can scale up by inserting more levels of switches, while each server is connected to only one of the bottom level switches.
- The recursive topologies still have levels/tiers as in the tree-based topologies.
- However, recursive topologies use lower level structures as cells to build higher level structures, and the servers in recursive topologies may be connected to switches of different levels or even other servers.
- There are multiple network ports on the servers of recursive topologies, making them significantly different from the tree-based topologies.
- Graphs, rather than multi-rooted trees, are suitable to depict recursive architectures.
- Some representative recursive topologies will be discussed in the following.

2.1 DCell:

- The most basic element of a DCell, which is called $DCell_0$, consists of n servers and one n-port switch.
- Each server in a $DCell_0$ is connected to the switch in the same $DCell_0$.
- Let $DCell_k$ be a level-k DCell.
- The first step is to construct a $DCell_1$ from several $DCell_{0s}$.
- Each $DCell_1$ has n+1 $DCell_{0s}$, and each server of every $DCell_0$ in a $DCell_1$ is connected to a server in another $DCell_0$, respectively.
- As a result, the $DCell_{0s}$ are connected to each other, with exactly one link between every pair of $DCell_{0s}$.
- A similar procedure is used to construct a $DCell_k$ from several $Dcell_{k-1s}$.
- In a $DCell_k$, each server will eventually have k + 1 links: the first link or the level-0 link connected to a switch when forming a $DCell_0$, and level-ɪ link connected to a server in the same $DCell_i$ but a different $Dcell_{i-1}$.



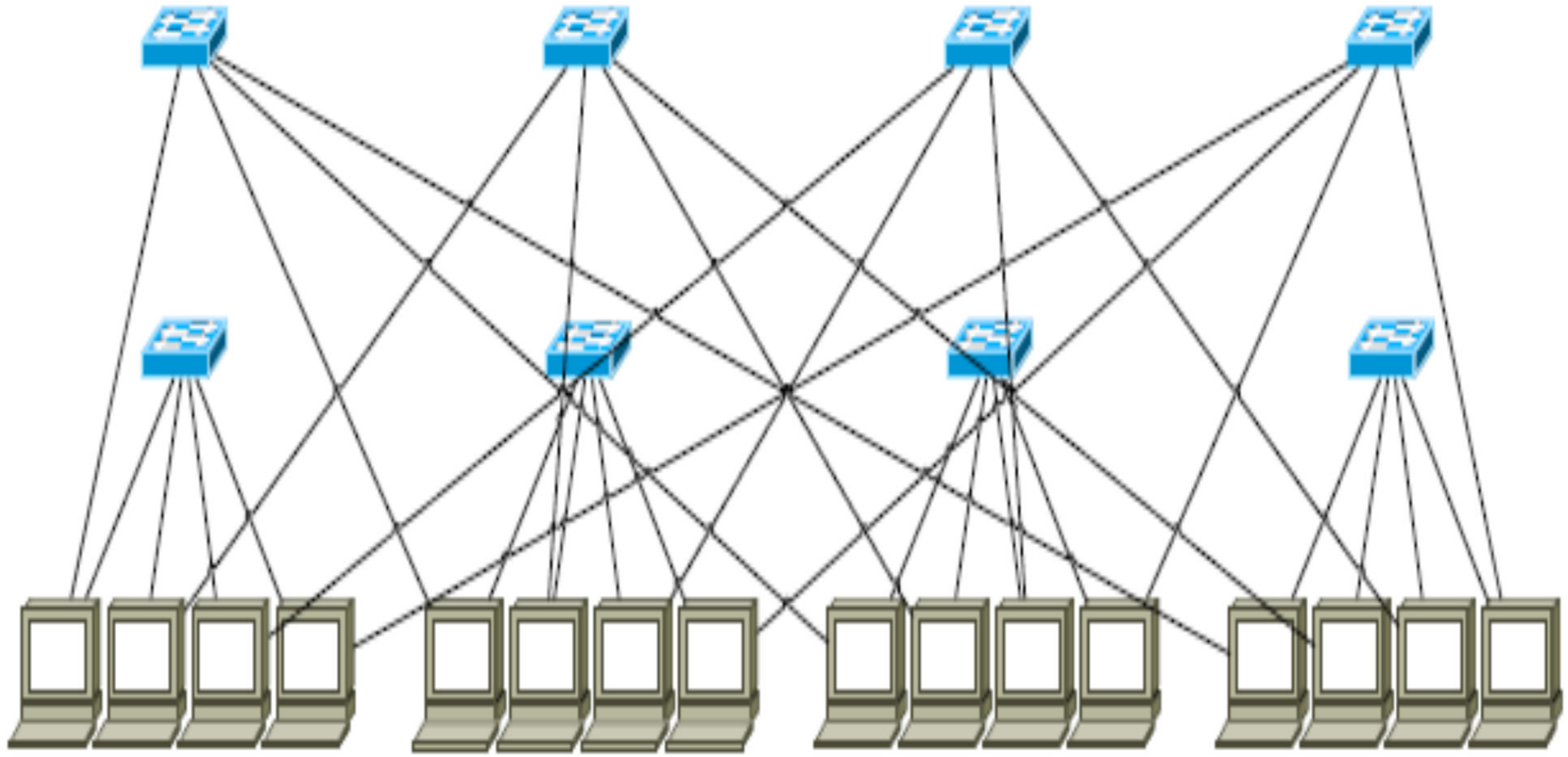
. 5. Constructing a DCell_1 from DCell_0s with $n = 4$

- Assume that each $Dcell_{k-1}$ has t_{k-1} servers, then a $DCell_k$ will consist of t_k $Dcell_{k-1}$s, and consequently t_{k-1} t_k servers.
- Obviously, we have $t_k = t_{k-1} \times (t_{k-1} + 1)$.
- Figure shows a $DCell_1$ when n = 4.
- It can be seen that the number of servers in a DCell grows double-exponentially, and the total number of levels in a DCell is limited by the number of NICs on the servers in it.
- DCell can scale to very large number of servers using switches and servers with very few ports.
- For example, when n = 6, k = 3, a fully constructed DCell can comprise more than 3 million servers

2.2 Bcube:

- BCube is a recursive topology specially designed for shipping container based modular data centers.
- Building a data center cluster in a 20- or 40-foot shipping container makes it highly portable.
- As demands change at different data centers, the whole cluster can be readily moved.
- While the deployment time is considerably shorter, the disadvantage of this environment is that due to operational and space constraints, once deployed in the field, it is difficult to service the cluster.
- The most basic element of a BCube, which is named as $BCube_0$, is also the same as a $DCell_0$: n servers connected to one n-port switch.
- The main difference between BCube and DCell lies in how they scale up.
- BCube makes use of more switches when constructing higher level architecture.
- While constructing a $BCube_1$, n extra switches are used, connecting to exactly one server in each $BCube_0$.

- Therefore a BCube_1 contains n BCube_0s and n extra switches(If the switches in the BCube_0s are taken into consideration, there are totally 2n switches in a BCube_1).
- More generally, a BCube_k is constructed from n Bcube_{k-1}s and n^k extra n-port switches.
- These extra switches are connected to exactly one server in each Bcube_{k-1}.
- In a level-k BCube, each level requires n^k switches (each of which is an n-port switch).
- BCube makes use of more switches when constructing higher level structures, while DCell uses only level-0 n-port switches.
- Both however require servers to have k+1 NICs.
- The implication is that servers will be involved in switching more packets in DCell than in BCube.
- Figure shows a BCube_1 with n = 4.
- Just like the DCell, the number of levels in a BCube depends on the number of ports on the servers.
- The number of servers in BCube grows exponentially with the levels, much slower than DCell.
- For example, when n = 6, k = 3, a fully constructed BCube can contain 1296 servers.
- Considering that BCube is designed for container based data centers, such scalability is sufficient



Constructing a BCube_1 from BCube_0s with $n = 4$

B. Flexible Architectures

1. c-Through:

- c-Through is a hybrid network architecture which makes use of both electrical packet switching network and optical circuit switching network.
- The hybrid network of c-Through, or the so-called HyPaC (Hybrid Packet and Circuit) Network, consists of two parts: a tree-based electrical network which maintains connectivity between each pair of ToR switches, and a reconfigurable optical network which offers high bandwidth interconnection between certain racks.
- Due to the relatively high cost optical network and the high bandwidth of optical links, it is unnecessary and not cost effective to maintain an optical link between each pair of racks; instead c-Through connects each rack to exactly one other rack at a time.
- As a result, the high capacity optical links are offered to pairs of racks transiently according to the traffic demand.
- The estimation of traffic between racks and reconfiguration of the optical network is accomplished by the control plane of the system. Figure shows a c-Through network

SDN USE CASES

- After much conversation around SDN theory, a number of SDN use cases are finally starting to emerge. We rounded up five examples of SDN in action.

1. Video and collaboration applications
2. Converged storage
3. A network exchange
4. Mobile network service orchestration
5. Scalable data center networks

1. Video and collaboration applications

- Session border controllers combined with Juniper's network virtualization platform, which, in turn, lets providers create capacity on demand for certain communication sessions.
- They can also set policies that adjust the Quality of Service level for each session.
- This allows better control over the network and new services that require dynamic service-level agreements.
- Providers can ensure better control over Quality of Service without having to expand capacity, since they'll be able to both provision and de-provision the network space based solely on need.

2. Converged storage

- Edgenet, a data services provider, is using SDN to create a programmable fabric across data center and storage technologies.
- The company opted for SDN with the goal of virtualizing the network and making it agile enough to keep up server and storage virtualization.
- With the combined technology, Edgenet is offering Software as a Service and data services.
- The company implemented NEC Corporation of America's Programmable Flow SDN ecosystem, which includes an OpenFlow controller, as well as high-speed physical switching and virtual switching that supported a Hyper-V environment and enables network virtualization provisioning.
- An added perk of adopting the SDN architecture was the company's ability to build a multipath, east-west fabric that ultimately will allow it to implement a fully converged storage and data center network.

3. A network exchange

- An SDN-based networking exchange was created by a group of organizations in Atlanta and allows carriers, research institutes and enterprises to interconnect physical networks using SDN.
- Members can also stretch virtual network segments or tenants across multiple physical domains with policy intact, which essentially allows them to form a private Internet using Layer 2 interconnect.
- The goal of the project is to eventually afford large institutions and enterprises the opportunity to avoid using public Internet and instead create private networks that host sensitive data and multimedia applications.
- Exchange directors haven't disclosed the full range of hardware and software they're using, however, they have confirmed Brocade is providing a cross-connect fabric.
- The group is using OpenFlow controllers and switches, but that isn't the only SDN route the group is exploring.
- The exchange plans to try out Cisco onePK and other approaches, as well. The exchange is also testing out "inter-controller protocols" that are still in development.

4. Mobile network service orchestration

- Recently, both SDN and network functions virtualization (NVF) have come center stage in mobile operator and other service provider networks.
- The technologies can enable resource and service orchestration with dynamic provisioning that can take minutes as opposed to months.
- Specifically, looking at control plane functions like a user data repository (UDR), SDN and NVF can aid with provisioning lead times.
- In addition, other issues solved by SDN and NVF include the separation of application logic and enforcement from corresponding subscriber data.
- The technologies address issues in two ways -- through dynamic resource orchestration and intelligent service orchestration.
- Dynamic resource orchestration, which uses NVF, requires a uniform virtualization stack across applications, often in the context of private cloud for operators.
- Intelligent service orchestration typically involves the principles of SDN, which includes switches, routers and applications at Layer 7 that are programmed from a centralized controller.
- Working together, these two approaches are said to dramatically cut down the long lead times that are associated with dimensioning and provisioning additional resources for an existing application.

5. Scalable data center networks

- SDN switches are being used to test a new data center network architecture at the University of Illinois at Urbana-Champaign's Ocean Cluster for Experimental Architectures in Networks.
- The new architecture allows engineers to incrementally scale bandwidth between servers and do so without spending significant money on hardware.
- The organization installed 13 Pica8 switches that have a total of 670 ports.
- The switches will be "sliced up" so they resemble a number of smaller switches; this will allow them to act as a large data center network to form a test bed for a range of SDN applications.
- Engineers will implement a new data center architecture in the test bed that strays from typical fat-tree topology and allows them to add switches with varying numbers of ports in order to incrementally scale bandwidth between servers.
- SDN controllers offer visibility and routing in this complex environment.