Seminar Report on

Data Center Network Design

Presented by

Chayan Sarkar

Roll No - 09305069

M.Tech. - I , CSE

IIT Bombay

email : chayan@cse.iitb.ac.in

http://www.cse.iitb.ac.in/~chayan

Guided by

Prof. Kameswari Chebrolu

April 16, 2010

Abstract

Data Center is a centralized repository where a large number of servers are clustered together to host a number of application and stores a huge amount of data. The primary goal of data center is storage and manipulation of data and serve the user's request efficiently all the time. With the advent of Cloud computing, the number of servers in a data center has become hundreds of thousands and it's going to expand more. Within this huge network, x-to-x traffic pattern will create several design challenges for network designer. So to design a data center, primary goal should be scalability, fault-tolerance, high network capacity [1, 2, 3, 6] etc. Also TCP throughput, efficiency of routing protocol and aggregate bottleneck bandwidth within a data center network will depend on the network design. Current tree-based structure with comparatively small number of servers will not survive in such large clusters.

1 Introduction

A data center, also called a server farm, is a computer house where a large number of computer systems are put together and form a network among them so that communication among them as well as communication with outside of the data center is possible. In Recent year, many large data centers are being built to provide increasingly popular online application services, such as search, e-mails, IMs, web 2.0, gaming etc.Companies need fast Internet connectivity and nonstop operation for establishing their presence on the Internet. Installing such equipment is not possible for many smaller companies. So they depends on data centers for lending them services. As the requirements and applications are increasing, the size of data centers are also increasing. So New technologies need to design to handle the scale and the operational requirements of such large-scale operations.

Apart from this, data centers also host many infrastructure services such as distributed file systems, structured storage, distributed execution engine etc. They generate various kinds of traffic such as one-to-one, one-to-many, one-to-all, many-to-many, all-to-all etc. Data centers require to connect the large number of servers via high-speed links and switches in such a way that they can support all this various kinds of applications and trafiic patterns efficiently. This report is focused on the networking infrastructure inside a data center called *data center networking* (DCN). There are mainly *three* design goals for DCN -

Scaling : Physical interconnection of hundreds of thousands or even millions of servers should be done at small cost, such as a small number of links at each node and no dependence on high-end switches. Data center has to enable incremental expansion by adding more servers into the already operational structure without affecting its operation. The routing protocol design also has to scale.

Fault tolerance : There are various server, link, switch, rack failures due to hardware, software, and power outage problems. As the network size grows, individual server and switch failures may become a normal event. Fault tolerance in DCN requires quick fault detection and availability of alternate physical connectivity.

High network capacity : Many online infrastructure services need large amount of network bandwidth to deliver satisfactory runtime performance. File replication and re-replication are two representative, bandwidth-demanding one-to-many and many-to-one operations. The traffic generated by the MapReduce application forms an all-to-all communication pattern and needs high network capacity from DCN.

To provide smooth service under heavy load, data centers deploy redundant or backup power supplies, redundant data communications connections, environmental controls (e.g., air conditioning, fire suppression) and security devices. This overprovisioning waste a lot of energy and increases exapanses. For better resource utilization and management, DCN structures should support Virtual Machine migration and agility.

Section 2 describes the idea of cloud computing; section 3 describes several scalable, fault tollerent DCN structures and their comparison. Besides section 5 discusses about TCP incast problem in data center and its solution; section 6 describes switching layer for incoming and outgoing data for data centers; finally, section 7 provides an approach towards greening the data center.

2 The Idea of Cloud Computing

The applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services are collectively called Cloud Computing[5]. Cloud means the hardware and software of a datacenter. When general public uses a Cloud and pay according to the usage (pay-as-you-go), it is called Public Cloud. The service being sold by Cloud is called Utility Computing.

2.1 Cloud User & Cloud Provider :

A small company will be beneficial after being a cloud user for many reasons - Cloud computing provide its users the illusion that it can have infitite computing resource if needed, the Cloud users don't need to plan far ahead or over-provision for peak demand, the users can pay on short term basis for computing resources etc.

Table 1: Top 10 Obstacles to and Opportunities for Growth of Cloud Computing.

	Obstacle	Opportunity
1	Availability of Service	Use Multiple Cloud Providers; Use Elasticity to Prevent DDOS
2	Data Lock-In	Standardize APIs; Compatible SW to enable Surge Computing
3	Data Confidentiality and Auditability	Deploy Encryption, VLANs, Firewalls; Geographical Data Storage
4	Data Transfer Bottlenecks	FedExing Disks; Data Backup/Archival; Higher BW Switches
5	Performance Unpredictability	Improved VM Support; Flash Memory; Gang Schedule VMs
6	Scalable Storage	Invent Scalable Store
7	Bugs in Large Distributed Systems	Invent Debugger that relies on Distributed VMs
8	Scaling Quickly	Invent Auto-Scaler that relies on ML; Snapshots for Conservation
9	Reputation Fate Sharing	Offer reputation-guarding services like those for email
10	Software Licensing	Pay-for-use licenses; Bulk use sales

A company should shift towards a cloud provider for many reasons - Company need to invest a lot of money initially and it will get its money's worth, it can leverages existing investments and customar relationships, it can defend a francise, become a standard through popularity etc.

2.2 Obstacles and Opportunities :

The top 10 obstacles to the growth of Cloud Computing and how to overcome them are listed in the table 1 (taken from [5]). The first three are technical obstacles to the adoption of Cloud Computing, the next five are technical obstacles to the growth of Cloud Computing once it has been adopted, and the last two are policy and business obstacles to the adoption of Cloud Computing.

2.3 Conclusions

2.3.1 Positive points :

(i) The illusion of infinite computing resources available on demand, thereby eliminating the need for Cloud Computing users to plan far ahead for provisioning. (ii) The elimination of an upfront commitment by Cloud users, thereby allowing companies to start small and increase hardware resources only when there is an increase in their needs. (iii) The ability to pay for use of computing resources on a shortterm basis as needed and release them as needed, thereby rewarding conservation by letting machines and storage go when they are no longer useful.

2.3.2 Negetive points :

(i) Applications Software needs to both scale down rapidly as well as scale up, which is a new requirement. Such software also needs a pay-for-use licensing model to match needs of Cloud Computing. (ii) Infrastructure Software needs to be aware that it is no longer running on bare metal but on VMs. Moreover, it needs to have billing built in from the beginning. (iii) The oppertunities described for the obstacles are on theoretical basis, no actual implementation is done.

3 Scalable, Flexible, Fault Tolerant DCN structures

3.1 Problem statement :

With a large number of servers how to design a servers, how to design a scalable, flexible, fault-tolerant data center network which will ensure high network capacity.

3.2 Challenges to design highly efficient DCN :

Each application in a data center is hosted on its own set of servers (potentially virtual). Each application has one or more public IPs to which clients communicates. Inside a data center, requests are spread among a set of servers who serves this request. Public IP of an application is called VIP. For each VIP, a pool of DIPs are assigned to serve the request.

The convential approach has the following problem - Hierarchical nature of the network imposes poor server to server connetivity. As the capacity of the links between access routers and borders routers is much less than the sum of capacity links of the servers connected with the access routers, the upper link becomes congested and full server link capacity can not be utilized. Popular load balancing techniques requires that all DIPs in VIP's pool should be within a same layer 2 domain. So if the application requires more servers, it cannot use servers from other layer2 domain - resulting underutilization of resources.

Apart from the convential challenges, the new architecture should meet the following challenges - The architecture should allow any server as a member of the server-pool behind any VIP, so that server pools can be dynamically shrunk or expanded. Many data center applications,

generate large the amount of traffic within the data center. The architecture should provide as much bandwidth as possible between every pair of servers in the data center, regardless of where they are located.

3.3 Design goal for modern & future data center :

Apart from the main design goals of a data center described in introductory section, new design challenges imposes couple of new goals - Any Virtual machine should be able to migrate to any physical machine without changing their IP addresses i.e. without breaking preexisting TCP connections and application-level state.
Another desirable property is agility which is the capacity to assign any service to any server. Agility ensures improved management, cost savings and load balancing.
Within the data center, any host should be able to efficiently communicate with any other host along any of the available physical communication paths, i.e. network bisection width should be higher.
As the number of servers increases failures will be a common rather exception. So failure detection should be fast and efficient.
With large scale and future expansion switches should not need to configure before deployment.
Within a data center traffic will be high. So forwarding should be efficient and loop free.
Any kind of traffic should be serviced to the upper limit of underlying physical layer's capacity.

3.4 DCell :

DCell is a network structure for data center, which is able to scale exponentially[2].

3.4.1 Solution Approach :

There are four components in a DCell - DCell network structure which is scalable, distributed and efficient routing algorithm on DCell network, to overome various types of failures (link/server/rack) a fault-tolerant routing and upgradation scheme which will be able to add new servers easily with the data center. DCell is a recursively defined architecture. Servers in DCell have multiple ports. Each server is connected with a single mini-switch and with many other servers via communication link. DCell[0] is the basic biulding block to construct larger DCell. It consists of n servers and they are conneted to a mini-switch. DCell[1] is formed using n+1 DCell[0]s. Each server in DCell[1] is assigned a tuple [a1, a0] where a1 is DCell[0] identifier and a0 idenfies a server within that DCell[0]. Now n+1 DCell[0]s are connected as following - Connect server [i,j-1] with server [j,i] for every i and every j¿i.
DCellRouting for unicasting is based on divide-and-conquer technique. For finding route between a *src* and *dst* first find the level in which they are within the same DCell. Suppose, they are both in DCell[k], but not in same DCell[k-1]. First find the nodes A and B which connects the DCell[k-1]s where *src* and *dst* resides. Then find route between *src* and A and *dst* and B. Final route is the combination of these three routes.
For Broadcasting, the *src* delivers packet to all of its neighbours. When a node receives a new broadcast packet, it forwards it to all of its neighbours except from which it comes. An older packet (duplicate) is not forwarded and dropped.
DCell Fault-tolerant Routing (DFR) introduces local-reroute which locally reroutes packets to bypass failed links.
DCells are built in top-down approach. To construct a DCell[k], build as many (incomplete) DCell[k-1]s as possible and fully connect them.

3.4.2 Positive points :

(i) A DCell with small server node degree can support several millions servers without using expensive core-switches. With server node degree, DCell scales double exponentially. (ii) DCell's routing protocol is fault-tolerant. Without using global states, it performs closely to the optimal shortest-path routing. (iii) DCell supports x-to-x traffic pattern with high capacity. Bisection width of DCell is quite higher than tree based structure.

3.4.3 Negetive points :

(i) In time of failure, TCP throughput dropped close to zero and recovered after few seconds. (ii) Lower layer links carry much traffic. So the aggregate bottleneck throughput is reduced. (iii) DCell uses more and longer communication links compared with typical tree based structure. DFR uses local rerouting to bypass failed link which might increase path length and local reroute is not loop free.

3.5 PortLand :

PortLand is a scalable, fault tolerant routing and forwarding protocol for data centers at layer2[6].

3.5.1 Solution Approach :

At the heart of PortLand, it employs a fabric manager (a user process running on a dedicated machine) to guide

with ARP requests, fault tolerance, multicast and maintain network configuration state (soft). Depending on the host's location in the topology, each host is assigned a hierarchical Pseudo MAC (PMAC) addresses. All packets are forwarded based on PMAC address. Portland assigns a unique pod number to edge switches and an unique position number to a physical machine within each pod using a Location Discovery Protocol. Edge switches assign a 48bit PMAC of the form pod.position.port.vmid to all Virtual Machines running on its directly connected hosts. An edge router receives all ARP request from its directly connected hosts and forwards it to the fabric manager. Fabric manager send PMAC as ARP reply if it is present at its PMAC table. Edge router then sends ARP reply to the end host. Otherwise, the fabric manager broadcasts to all end host to retrieve the mapping. PortLand uses a Location Discovery Protocol to work switches in plug-and-play manner. PortLand switches periodically send a Location Discovery Message (LDM) to all of its ports to set their positions or to monitor liveness when position is already set. Switches begin packet forwarding only after establishing their location. The main concern of PortLand's routing protocol is to detect switch and link failure/recovery.

3.5.2 Positive points :

(i) Using Portland VM migration can be done without affecting ongoing Tcp connections. (ii) The forwarding protocol is provably loop free. ARP and routing messages are not broadcasted generally. (iii) Fault tolerant routing makes it very useful. PortLand requires O(n) communication and processing(in traditional protocol it is O(n2)), one message from the switch detecting failure to the fabric manager and, in the worst case, n messages from the fabric manager to affected switches.

3.5.3 Negetive points :

(i) Portland is designed on baseline multirooted network topology which is known for data center environment. In future, if any other topology evolve then portland may not be useful. (ii) If centraliged fabric manager fails the whole Portland scheme will fail. Also with increasing servers and intercommunication between servers may create bottleneck at fabric manager. (iii) Switch position may be set manually with administrative intervention, violating some of the original goal. (iv) Testbeds were very small compared to actual data center. So, the results may vary from theory and test results in practical situation.

3.6 VL2 :

VL2 is network architecture for huge data center, which spreads loads equally among the servers [1].

3.6.1 Solution Approach :

VL2 uses Valiant Load Balancing (VLB) to spread destination independent traffic among multiple servers who is serving it. VL2 network is built from low-cost switches arranged into Clos topology to support large path diversity. VL2 uses IP routing and forwarding technologies and TCP's end-to-end congestion control mechanism.
To achieve agility, VL2 employs a directory system which maintains mapping between application-specific address (AA) and Location-specific address (LA). A service request comes with AA and depending on the loads it is mapped to LA of a less loaded server. AA addresses for applications remains same, no matter what is its location (LA) due to virtual machine migration.
To achieve arbitrary traffic matrices VL2 deploys to mechanisms - VLB and ECMP. VLB distributes traffic across a set of servers using flows. ECMP distributes traffic across equal-cost paths. In VL2 performance isolation is done by spliting the intermiadte nodes in equal ratio to prevent any hot spots in the data centers via randomization.

3.6.2 Positive points :

(i) Random flow spreading in VL2 gives network utilization close to optimal resource utilization. (ii) The Virtual Layer 2 provides flat addressing. As a result any service can be assigned to any server - agility achieved. (iii) VL2 provides uniform high capacity between any two servers. Performance of one traffic can not be affected by traffic of another application.

3.6.3 Negetive points :

(i) How to add new servers to the server pool (scale-up) is not discussed. (ii) Tunneling of each packet may cause overhaed and increase delay (as latency within a data center is very low). (iii) The directory system may face bottleneck at heavy load.

3.7 BCube :

BCube is a network architecture designed for shipping container based, modular data centers[3].

3.7.1 Solution Approach :

BCube Architecture has two parts - the BCube structure and BCube Source Routing.
`BCube Structure :` BCube is a server-centric network

structure. There are two types of devices which forms BCube structure - servers with multiple network ports and COTS miniswitches which connects servers at different layers. A BCube[k] is constructed recursively from n BCube[k-1]s and nk̂ n-port switches. In a BCube structure switches never directly connect to other switches and they just do forwarding. In a BCube[k], there are k+1 parallel paths between any two servers.

`BCube Source Routing (BSR) :` It is a reactive routing protocol. When a new flow comes, the source sends probe packets over multiple parallel paths. The destination returns a probe response to the source. On receiving the responses, the source uses a metric to select the best path. Source specifies the total path to the destination in the packet header. In this way, it can control the routing path without any interaction with the intermediate servers. Intermediate servers do not involve in routing. They just forward packets by looking at the packet headers which simplifies their functionalities. In large cluster, link state broadcasting will create a large number of traffic. THis is avoided by using reactive probing. In BSR packet are transffered in order as a flow uses one path at a given time.

3.7.2 Positive points :

(i) BCube significantly accelerates one-to-x traffic patterns and provides high network capacity for all-to-all traffic. BCube has multiple parallel paths between a source and destination server. (ii) In case of fault, performance of BSR does not dropped drastically, performance degrades gradually as the fault increases. (iii) Lowend COTS miniswitches provides better performance-to-price ratio. As the switches are not programmed, there is no need to upgrade them.

3.7.3 Negetive points :

(i) BCube uses more wires and switches than tree structures. (ii) For partial BCube, which is very likely, the BSR does not work well for some server pairs. (iii) Experiments are done on very small testbeds compared to actual server pools.

3.8 Comparison :

The relative comparison between the previously described four structures are listed in table 2.

Table 2 : Comparison between different DCN architecture.

	DCell	PortLand	VL2	BCube
Architecture	Non-hierarchical, recursive structure	Multi-rooted Tree	Clos network	non-heirarchical, recursive structure
Centralized Elements		Fabric manager	Directory System	
Routing	Divide-and-conquer, fault-tolerant, uses local rerouting	hierarchical PMAC fault-tolerant, proxy-based ARP	encapsulation, fault-tolerant, random traffic spreading	probing message, fault-tolerant, auto load-balanced
Traffic treatment	level-0 carry higher traffic	not discussed	equally distributed among servers	equally distributed
Main Contribution	Double Exponentially scalable	Supprot VM migration, efficient fault-tolerant routing	high network capacity, promises agility	high network capacity, graceful performance degradation

4 TCP Incast problem and solution for Data Centers

4.1 Problem Statement :

In a data center with thousands of servers and with high bandwidth and low delay, the TCP throughput reduces drastically when many servers communicate simultaneously with one receiver - called TCP incast problem[7]. Increasing TCP throughput by overcoming TCP incast problem in challenging for data centers.

4.2 Challenges with overcoming TCP incast problem:

High bursty and fast data transmission overfill Ethernet switch buffers which is a normal situation in data center. As a result packets are dropped at the buffer, causing TCP timeouts and TCP throughput is reduced. As TCP minimum retrasmission timeout is 200ms, TCP experience several 200ms (minimum) timeouts in case of severe packet loss. In case of barrier-synchronized request, a receiver cannot proceed further unless it get response from all the servers. So if packet of one flow is dropped, the whole connection will be suffered. For TCP incast to take place, the network should have high bandwidth, low delay with small switch buffer, and data is stored across many servers in small pieces which is very likely situation in data centers. Increasing switch buffer may be one solution for this problem, but with a cost.

4.3 Solution Approach :

Buffer overflow is unavoidable, but how to maintain high throughput. Solution is to reduce TCP minimum retransmission timeout so that if packets get dropped it can be retransmitted fast. It is necessary to reduce because Min. retx. timeout because it (200ms) is much higher than RTT (¡40ms) in data center.
Eliminating RTO min improves the performance for certain number of servers. But, in data center with thousands of servers, just eliminating RTO min. can't improve throughput significantly, because TCP implementation uses coarse-grain periodic timer. So tighter TCP timeout requires fine-grained RTT measurement. The generic time of day (GTOD) framework uses CPU cycle counter to give nanosecond resolution timekeeping to kernel and other application. To implement microsecond granularity, low resolution timer is replaced by high resolution one and some TCP constants are redefined.
TCP RTOmin helped to avoid spurious retransmission. But experiment shows that in wide area also finer granularity is safe.
Delayed ACKs are implemented to reduce the traffic for ACK with intention to piggy-back ACK with data. Experiment shows in wide area transfer, delayed acks decreasesTCP throughput.

4.4 Positive points :

(i) TCP catastrophic throughput collapse can be prevented using fine-grained TCP retransmission. (ii) The wide-area evaluation support that the modification specified are safe for TCP-based cluster communication. The modification will not cause heavy overhead or extensive changes. (iii) This implementation will also be helpful for latency-sensitive data center application which requires low response time.

4.5 Positive points :

(i) The high resolution timer could lead to overhead in terms of interrupts. (ii) Eliminating RTOmin may cause spurious retransmission in wide area. It's effect is not measured extensively. (iii) If ACKs are delayed, then throughput gain may not be achieved much. So how to eliminate or avoide delayed ACK is not discussed.

5 Switching Layer for Data Centers

5.1 Problem Statement :

To protect, manage and improve the performance of applications and services run by data centers, they need to deploy a variety of middleboxes, like firewalls, load balancers, SSL offloaders, web caches, and instrusion prevention boxes,

5.2 Challenges in Middlebox Deployment:

The main challenge in deploying middlebox is that there are no available protocols and mechanisms to explicitly insert these middleboxes on the path between endpoints. While deploying middleboxes, the following issues pose great challenges - 1) Correctness : Under all network conditions traffic should traverse middleboxes in the same sequence as specified by the network administrator . 2) Flexibility : With the change of application requirement the sequence of middleboxes should be easily (re)configured, 3) Efficiency : Traffic should not traverse unnecessary middleboxes.

5.3 Solution Approach :

With lack of protocol and implement policy in exiting protocol, it is better to design a new layer-2 for data centers consisting. A policy-aware switching layer or PLayer is designed with inter-connected policy-aware switches or pswitches[4]. Two principles guided the design of PLayer to satify desired properties: (i) Separating policy from reachability, and (ii) Taking middleboxes off the physical network path. A centralized policy and middlebox controller handles PLayer by setting up and maintaining the rule tables at the various pswitches. When the pswitch receives a frame, it (i) Identify the previous hop traversed by the frame, (ii) Determine the next hop to be traversed by the frame, and (iii) Forward the frame to its next hop. PLayer uses 5-tuples composed of source and destination IP, port numbers and protocol types to distinguish different traffic and to adapt of policy. A pswitch composed of two independent parts the Switch core (provides regular ethernet switching functionality) and the Policy core (redirects frames to the middleboxes according to policy).

5.4 Positive points :

(i) Simple implementation fulfils desirable properties. PLayer guarantees correct middlebox traversal under churn. (ii) To ensure that resources are not wasted serving unnecessary traffic or packets get stuck on inactive network paths, middleboxes are taken off the physical path. It prevents large scale traffic shifts on middleboxes. Only necessary traffics traverse the middlebox. (iii) Addding, deleting and modifing policies are easier as they are specified in a centralized policy controller and all pswitches uses same policy set.

5.5 Negative points :

(i) There is some bandwidth overhead and latency increase as the packets needs to travel the PLayer each time before traverse through a middlebox. (ii) 5-tuples may not be sufficient for distinguishing different types of traffic. (iii) The PLayer is designed assuming hierarchical network structure. In other network structure using PLayer may not be efficient.

6 Greening the Data Centers

6.1 Problem Statement :

Energy consumption in modern data centers is an major issue as there is a lot of wastage of energy in a data center for overprovisioning, Heat Dissipation etc. In future the number of data centers and their size is going to expand. So, greening approaches should be taken.

6.2 Challenges in Greening :

Data centers are built with thousands of serevrs at a place (centralization trend). Data centers need to over-provision to serve smoothly during peak load. As the number of server increases, more are more equipments need to be deployed for heat dissipation. As a data centers are built at one location, with increased distance to end-users, intermediate network equipments also consumes more energy. All of these obvious issues related to current data center design increases energy consumption.

6.3 Solution Approach :

One approach to reduce energy consumption at data center is a new distributed computing platform called Nano Data Centers (NaDa)[8]. NaDa is based on P2P architecture, but it is coordinated and managed by an ISP which controls the home gateways to provide computing and storage. The gateways act as nano data centers. Nada saves energy by reusing the already committed base-line power of DSL gateways because they are already on the provide internet connectivity. The success of Nada largely depends on the number of the devices which are on at a point of time.

`VoD service Architecture in NaDa :`
`Gateways` - Gateways are provided extra storage and bandwidth resources. In VoD application, they store full or partial replicas of video content objects and act as a server. If some gateways don't have a video content it can download it from its neighbouring gateways. On the upload line, NaDa gateways have one dedicated virtual circuit for conventional Internet use and another one is allocated for NaDa use through which neighbouring gateways can download content from it. Two virtual circuits are given different capacities.
`The Tracker` - The coordination among all VoD-related activities in NaDa are done the Tracker. Tracker monitors the availability and content stored in each gateways. If some gateway asks for a content, tracker provides a list of gateways who has the content. The tracker also informs gateways for content upgradation.
`Content servers` - Content servers actually stores the content in a real data center. Content servers provides content to the gateways in offline mode. Number of content server are small and overprovision is not required which leads to energy saving. If no gateway provide a content, the content server serves the content online like usual data center.
`Content placement strategy` - To make it more manageable, contents are split into smaller chunks called data windows and each data windows are replicated (same no

of replica for each chunk) among the gateways. Number of replica depends on the popularity of the movie. Movies are classified into three groups. The Most popular, "hot" contents, should be replicated on all gateways. The subsequent popular movies, "warm" contents are replicated minimally so that their request can be served from gateways. The less popular movies, constituting "cold" content, are not stored in NaDa and if they are requested, they are served from the content servers.

6.4 Positive points :

(i) With few GB of memory per Gateway in a metropolitan area deployment, experiment for VoD application shows that NaDa can save up to 60% of the energy spent by legacy data centers to provide the same services, (ii) Using NaDa number of content servers and network resourses can be reduced and as a result power consumption is reduced. (iii) NaDa reduces service delay, communication cost as most of the are closer to the user and also saves energy in powering and cooling the network equipment. (iv) The authors developed a model to evaluate the energy needed to provide services in both centralized data centers and Distributed NaDa.

6.5 Negetive points :

(i) Success of NaDa depends on the gateway storage capacity and uplink bandwidth. It needs lot of investment from an ISP to provide necessary uplink bandwidth and additional storage capacity to the gateways. (ii) ISP take advantage of the user's own electricity to offer services to others. So there is a concern of 'incentive'. (iii) Practical deployment issues and feasabilities are not described. As, there is a large number of data centers holding millions of contents, which contents should be stored in gateways. (iv) There is an issue of data security as data stored in distributed manner. Also generalised NaDa architecture is not described.

7 Conclusion

DCell structure is perfect regarding scalability as it is double exponentially scalable. But for one-to-x traffic pattern it does not provide good throughput because alternate path lengths between server pairs in DCell are not same. Also its aggregate bottleneck bandwidth is much lower than BCube. With the use of many swicthes and wiring (which is affordable in container based data center) BCube provide best one-to-x throughput and its performance degrades gracefully in case of faults. VL2 provides high capacity between servers. Using its directory structure and valiant load balancing, traffic isolation, load distribution and agility is achieved. Also with a centralized dictory, load balancing can be implemented in PLayer. But VL2 as well as PortLand did not considered scalabity issues. With its flat addressing, PortLand supports VM migration. With effective VM migration VMs can be clustered around few servers in case of light load and remaining servers can be switched off if possible.

References

[1] Albert Greenberg et al. Vl2: A scalable and flexible data center network. *In SIGCOMM*, 2009.

[2] C. Guo et al. Dcell : A scalable fault tolerant network structure for data center. *In SIGCOMM*, 2008.

[3] Chuanxiong Guo et al. Bcube: A high performance, server-centric network architecture for modular data centers. *In SIGCOMM*, 2009.

[4] Dilip A. Joseph et al. A policy-aware switching layer for data centers. *In SIGCOMM*, 2008.

[5] Michael Armbrust et al. Above the clouds: A berkeley view of cloud computing. *TR UCB/EECS-2009-28*, February 10, 2009. `http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html`.

[6] Radhika Niranjan Mysore et al. Portland: A scalable fault-tolerant layer 2 data center network fabric. *In SIGCOMM*, 2009.

[7] Vijay Vasudevan et al. Safe and effective fine-grained tcp retransmissions for datacenter communication. *In SIGCOMM*, 2009.

[8] Vytautas Valancius et al. Greening the internet with nano data centers. *In ACM CoNEXT*, pages 37–48, 2009.