Performance Analysis of R-DCN Architecture for Next Generation Web Application Integration

[1]C.C.Udeze, [2]Okafor, K.C
[1&5]R&D Department ELDI, Awka, NASENI
[2]Dept. of Electrical and Electronic Engineering. FUTO, Owerri, Nigeria
[1]udezechidiebele@yahoo.com, [2]arissycline@yahoo.com,

[3]C.C.Okezie, [4]I.O.Okeke.[5]Ezekwe.C.G.
[3]Dept. of Electronic & Computer Engineering, NAU, Awka.
[4]Dept. of Electronics Engineering UNN, Nsukka.
Nigeria
[3]christianaobioma@yahoo.com, [4]ijeomaokeke@yahoo.com,

***Abstract*—With the astronomical growth in online presence vis-à-vis the IT industry across the globe, there is an urgent need to evolve cloud based DataCenter architectures that can rapidly accommodate web application integrations which can serve the energy industries, educational sector, finance sector, manufacturing sector, etc. Previous works in DataCenter domain have not investigated on cloud based DataCenter Servercentric characteristics for evaluation studies. This paper then proposed a Reengineered DataCenter (R-DCN) architecture for efficient web application integration. In this regard, attempt was made to address network scalability, efficient and distributed routing, packets traffic control, and network security using analytical formulations and behavioural algorithms having considered some selected architectures. In this work, a simulation experiment was carried out to study six key performance metrics for all the architectures. It was observed that the network throughput, fault-tolerance/network capacity, utilization, latency, service availability, scalability and clustering effects of R-DCN responses with respect to above Key Performance (KPIs) metrics were satisfactory when compared with BCube and DCell architectures. Future work will show a detailed validation using a cloud testbed and CloudSim Simulator.**

Keywords: Cloud, Architectures, Servercentric, R-DCN, KPIs, Performance, Traffic, Metrics.

I. INTRODUCTION

Majority of enterprise servers and storage systems (such as Enterprise Resource Planning solutions (ERPs), Application servers, E-commerce servers, Security systems (IDS)) are located, operated and managed. It is also referred to as the consolidation point for provisioning multiple services that drive enterprise business processes. For example, financial institutions like banks, educational institutions like universities, internet service providers (ISPs), internet-based organisations such as Google, twitter, face book etc, oil and gas industries all have DataCenters where their data are stored, operated and managed. Some of them have and manage their own DataCenters while others outsource to bigger DataCenters due to high cost of owning, managing and maintaining a DataCenter networks.

In recent times, DCNs have attracted a lot of interest in the enterprise networking industry. They are used to provide data storage and files transfer where end stations are interconnected as clusters and bladed systems [1]. DCN represents the heart of any organization's network and since business continuity is a priority objective to enterprise organizations, operations must maintain a near zero downtime running twenty-four hours in this type of network. From a current report, companies such as Microsoft, Google, and Yahoo have built data centers composed of hundreds of thousands of servers, and the number is still increasing [2], [3], [4]. Also some universities or institutions have thousands of servers or even more in their data centers [5], and these servers need to be efficiently interconnected because of large volumes of data exchange among them, from running web applications such as search, gaming, web mail, to providing infrastructure services such as GFS [6], map-reduce [7] and Dryad [8].

Achieving efficiency in DataCenter server interconnectivity is still a challenging task despite the availability of inexpensive commodity PCs, which has made it possible to expand a DataCenter to a huge number of servers. The major drive towards this efficiency in DataCenter server interconnection network is high availability (i.e. high bandwidth) since more and more enterprise-class mission critical applications are migrating into data centers. Also, this high bandwidth as a result of efficient data center server interconnection facilitates distributed applications requiring frequent data accessing and shuffling [9], [10], [11].

Scalability for future incremental expansion of the network without complexity is a one of the major design goals in the design of DCNs for efficient web application integration in enterprise organizations. In scaling a DC to more than a few thousand of servers, the traditional way of connecting multiple levels of switches into a tree and attaching servers as leaves of the tree faces a lot of difficulties as the number of servers grow.

This tree-based structure does not scale well for two reasons. Firstly, the servers are typically in a single layer-2 broadcast domain [12].

Secondly, core switches, as well as the rack switches, pose as the bandwidth bottlenecks. The tree structure is also vulnerable to single-point-of-failure": a core switch failure may tear down thousands of servers. A quick fix using redundant switches may alleviate the problem, but does not solve the problem because of inherently low connectivity. To achieve efficient data center scalability, networking intelligence is moved from switches to servers, i.e., each

978-1-4799-4998-4/14/$31.00 ©2014 IEEE

server can forward traffic for other servers [13]. Such a DCN is often referred to as a server-centric network. Besides the ability to easily scale the network size, a server-centric network offers other advantages in building large scale data centers. These include cost reduction since the switches can be simple layer-2 plug-and-play devices, and it is more convenient to develop and deploy routing and management mechanisms in server-centric DCNs as servers are easier to program than switches.

A. Research Motivation

DCN must be fault tolerant against various types of server failures, link outages, or server-rack failures. Failures are quite common in current DataCenter [14],[15]. Hardware, software, and power outage problems normally result in various servers, link, switch, rack failures. The growth of the network size leads to individual server and switch failures that may become the norm rather than exception. Fault tolerance in DCN requests for both redundancy in physical connectivity and robust mechanisms in protocol design [12]. This fault tolerance network structure and protocol design together with scalability and high bandwidth formed the motivation for reengineering the traditional DCN in order to achieve efficiency in integrating web applications. To meet these design goals in DCN reengineering, this paper introduced key technologies viz: Server virtualization and VLAN segmentation.

Server virtualization for instance, affects DCNs resource allocation [16] in terms of higher bandwidth provisioning, performance optimization and infrastructure cost reduction in terms of deployment and maintenance [16]. Hence, to enable DataCenters perform multiple processing for many enterprises as well as Internet business applications, such as those used by financial institutions, Google, twitter, facebook etc, server virtualization increases the over-all bandwidth as well as reduces the cost of deployment and maintenance of the IT infrastructure.

Similarly VLAN segmentation in the R-DCN will improves the over-all network security by isolating groups, controls traffic broadcast, and optimizes the over-all network management and performance. Therefore, with virtualization and VLAN segmentation scheme, efficient server interconnection, together with our fault-tolerant routing algorithm (Carrier Sense Multiple Access with Collision Detection plus Traffic Arbitration Messaging Protocol), the proposed R-DCN for efficient web application integration will be achieved.

B. Contributions

The model will address possible server-centric failures such as link, server and rack failures and a multi-tenanted incremental upgrade scheme for the reengineered DCN expansion size. Hence, while considering the server interconnection concept of the DCell architecture, we introduced the following in a previous research [16][22] viz:

- Analytical model that will handle traffic control issues in DCNs.
- A load balancing service which is meant to suppress or normalise failure effect while being fault-tolerant at the same time.
- VLAN segmentation for logical isolation of the servers in the architecture.
- Server virtualization in the reengineered DCN for broadcast traffic regulation optimization.

In designing the R-DCN model, the main goals are to maintain throughput, low latency, fast convergence and scalable network that are flexible with less administrative overhead. The network should be capable of supporting high performance workloads (HPW) besides conventional Ethernet traffic like web applications and cloud services. Meanwhile, multiple flows from various subnets should be localized with VLAN strategies, but still shares link capacity fairly (buffer capacities). The performance of today's DCNs is measured by QoS parameters such as throughput, latency, and service availability. From the application perspective, throughput refers to the data rate (bits per second) generated by the application while latency is all about how fast the network is in packet delivery. Service availability refers to a network being available twenty-four per week to serve its users with no downtime.

II. RELATED WORKS

The design of DCNs has been a very interesting area and many research groups have proposed several architectures. This section will carry out its review on conventional architectures, viz: Fat-tree, Monsoon, BCube, MDCube, VL2, DCell, and Synthesis VLAN architectures.

A. Fat-Tree Architechure

Fat-tree as introduced by [5] enables the use of inexpensive commodity network elements for the architecture. Fig. 2.1 shows the Fat-tree design in which all switching elements in the network are identical. Due to the use of inexpensive commodity network elements, the cost of Fat-tree network is less than traditional one.

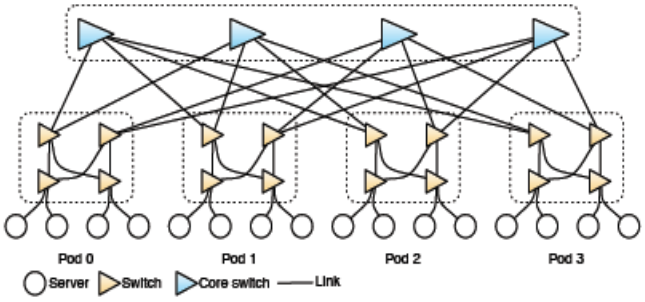

Fig. 1. Fat-tree Architecture [5].

From Fig. 1, the size of Fat-tree depends on the switch properties. Hence, switch with 48 ports can support a network with 27,648 hosts and scaling out to support networks with over 100,000 hosts requires improved switches. Packaging and placement techniques are proposed to address the issue of wiring which is a serious challenge with Fat-tree design.

B. Monsoon Architecture.

The authors in [17] proposed a mesh-like architecture (Monsoon) for cloud services. Monsoon uses commodity switches to reduce the cost and allows powerful scaling up to 100,000 servers. Monsoon as shown in Fig. 2 improves performance by the use of Valiant Load Balancing (VLB). Though Monsoon focuses on layer 2 but the general architecture is divided into Ethernet layer 2 and IP layer 3. The benefits of layer 2 include elimination of the server fragmentation (i.e. all applications can share a huge flat address space), cost savings, and avoiding disturbance of the IP-layer functionality.

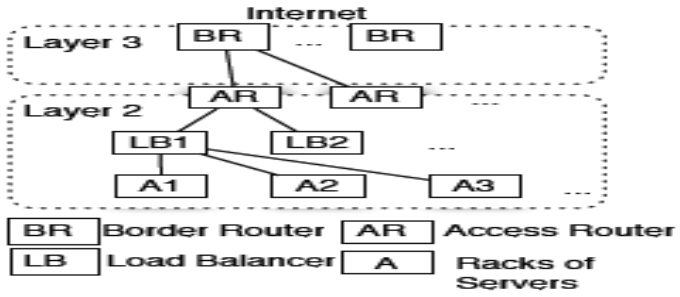

Fig. 2. Monsoon Architecture [17].

From Fig. 2, the upper layer switches should have 144 ports with 10-Gbps while the top-of-rack switch should handle 20 server's 1-Gbps link onto 2 10-Gbps uplinks. Hence, the architecture can allow over 100,000 servers with no oversubscribed links in layer 2. The commodity servers were used to build load balancers (LB), instead of specialized and expensive hardware.

C. BCube and MDCube Architectures.

Modular Data Centers (MDCs) is formed by a few thousands of servers that are interconnected via switches and packed into a 20-feet or 40-feet shipping-container, examples are the BCube and MDCube.

The authors in [18] explained that BCube is designed based MDC concept. Hence, it is a shipping-container that is based on modular data center (MDC). BCube is a server centric design as illustrated in Fig. 3, and it uses only commercial-off-the-shelf (COTS) switches and commodity servers, hence, cost-saving. Each server in the BCube has a small number of network ports that connect to mini-switches. Being a server-centric architecture, BCube uses its servers for routing instead of routers. The authors claim that BCube supports one-to-x (one-to-one, one-to-several and one-to-all) better than Close topology based solutions, such as Monsoon, VL2 and Fat tree. Furthermore, their results show that BCube offers more graceful performance degradation than typical network architectures. The benefits of MDCs are that they offer short deployment time, lower cooling and manufacturing cost, and higher system and power density.

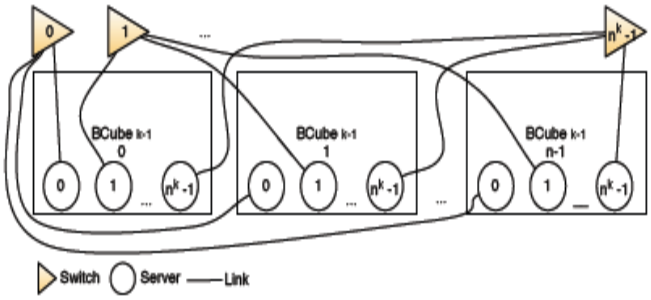

Fig. 3. BCube Architecture [18].

The authors in [19] proposed MDCube architecture that provides good fault tolerance, high network capacity for mega-data centers and manageable cabling complexity. The paper explained that MDCube is a structure constructed for mega-data centers based on containers. Containers in MDCube follow the BCube design, which connects thousands of servers inside the container. In other words, MDCube is a design to achieve a mega-data center using BCube-based containers as building blocks. MDCube is designed in such a way that BCube containers in MDCube are interconnected using high-speed interfaces of switches in BCube. Hence, BCube containers act as a virtual node in MDCube with the MDCube switches being virtual interfaces to these virtual nodes. As illustrated in Fig. 4, MDCube is a server-centric design, thus leaving the logic to the servers. As in server-centric manner, MDCube requires networking stack modifications for load balancing and fault tolerant routing. The author argued that the MDCube inter-container cables number is reduced to almost magnitude of two orders when compared with mega-data centers constructed from single structure designs such as BCube or Fat-tree.

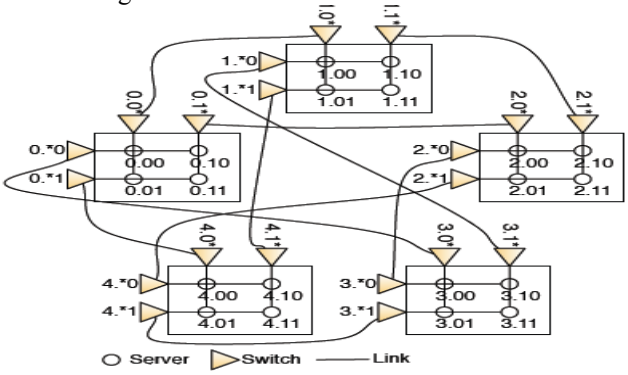

Fig. 4. MDCube Architecture [19].

D. VL2 Architecture.

The author in [20] introduced VL2 network architecture that uses Valiant Load Balancing (VLB) for traffic spreading, address resolution supporting large server pools and flat addressing to avoid fragmentation of resources. Their VL2 topology provides path diversity and can offer agility. Furthermore, the need for over-subscribing links is eliminated in the VL2 network design. It requires a directory service and server agent for VL2 addressing and routing as in the case of

Monsoon. Also, for VL2 addressing and routing design to be enabled, it seems VL2 requires changes to servers' network stacks. The authors' evaluation of VL2 performance indicated that VL2 is efficient and achieves high load balancing fairness. Furthermore, rough cost estimates indicated that a typical network without oversubscribed links costs 14 times more than equivalent VL2 network.
Nevertheless, according to the authors of MDCube [19], VL2 design is still expensive, since they use rather high end switches in the layer 2. Hence, building a 1 million server network would require 448 448-port 10Gbps intermediate and aggregate.

E. DCell Architecture.

In their effort to efficiently interconnect an exponentially increasing number of servers, which is a fundamental challenge in data center networking, the authors in [12] presented DCell which has a recursively defined structure, in which a high-level DCell is constructed from many low-level DCells. In this case, DCells at the same level are fully connected with one another. This fault tolerant architecture scales doubly exponentially as the node degree increases. DCell is fault tolerant since it does not have single point of failure and its distributed fault-tolerant routing protocol performs near shortest-path routing even in the presence of severe link or node failures [12]. In the DCell architecture of figure 5, each server has two links in each DCell. One connects to its mini-switch, hence to other nodes within its own DCell, the other connects to a server in another DCell. Due to this server interconnection, the authors claimed that DCell provides higher network capacity than the traditional tree-based structure for various types of services. The work further stated that DCell can be incrementally expanded and a partial DCell provides the same appealing features. However, the logical isolation of the servers in the architecture and optimization of broadcast traffic regulation cannot be achieved in the DCell architecture due to lack of VLAN segmentation and virtualization instances.

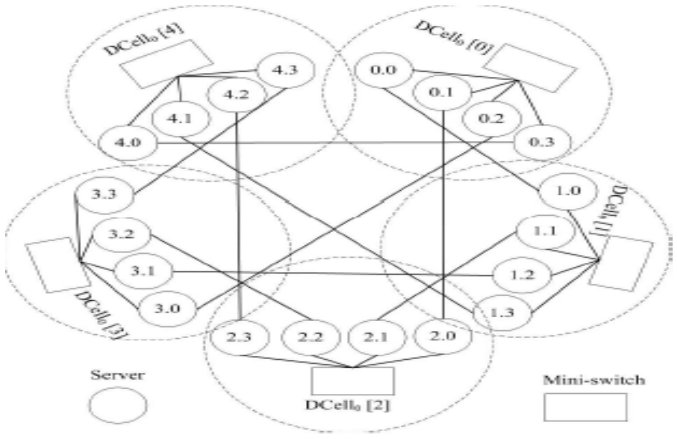

Fig. 5. DCell Architecture [12]

F. Synthesis VLAN Architecture.

An attempt was made in [1] to reengineer the DCN for effective throughput, low latency and server utilization. Fig. 6 showed their model architecture which they called Synthesis VLAN. The authors carried out a detail analysis of Synthesis VLAN and presented an approach to congestion management while showing the simulation results. Their work showed that S-VLAN improves propagation delays in a congested network link thereby enhancing performance and justifying their performance indices 'Quality of Service'. They concluded consequently, that it is obvious that by introducing the VLAN switch(s) in a Data Center domain, a function demarcation between the different services (TCP and UDP) is possible, and the possibility of minimizing the congestion problem is increased with little overhead. Their results finally showed that an improvement in the throughput performance is obtained specially for high offered loads.

However, because their reengineered DCN (Synthesis VLAN) was not based on new technologies such as virtualization and consolidation, their system has some limitations as shown in [21] includes:

- Expensive to acquire and maintain hardware (Servers)
- Poor scalability
- Difficult to replicate servers and services
- Redundancy is difficult to implement
- Vulnerable to hardware outages
- In many cases, there is under-utilization of server processors

From the literature studies, this paper will evaluate its proposed architecture with BCube and DCell architectures as they seem to have a better feature for web application integration. For efficiency in web application integration which is our major drive for this work, such efficient interconnection of DCN servers provides higher network capacity for efficient packet delivery and congestion free network. The system block diagram and the architectural description are detailed in [22].

III. PROPOSED R-DCN ARCHITECTURE

In this paper, the term R-DCN can be used interchangeably C-DCN (i.e. Cloud DCN). The R-DCN model overview shown in Fig. 6 will aid the understanding of the model specifications described in [22]. Now, R-DCN Recursive Construction Algorithm, Logical Isolation of R-DCN Architecture, Modeling MLS Broadcast Traffic Characterization, Mathematical induction algorithm for performance enhancement have been achieved in a previous work [21], [22], [23]. Fig. 6 shows the port scalability interface design for clustered servers. In this model, this paper will derive a scalability model for the R-DCN.

A. Model Design for Scalability.

Fig. 6 shows an enterprise scalable architecture which offers rapid expansion without complexity for future purposes.

Now, let R-DCN$_s$ denote a high level R-DCN subnet and *R-DCN$_{s-1}$* denote a low-level one. Where s is a subnet factor (i.e. s ranges from 1 to 4). Hence, if a high-level subnet is subnet 4, a low-level subnet is subnet 3 and that continues in that order.

Normally a high level *R-DCN$_s$* should be connected from a low level *R-DCN$_{s-1}$.*

If we denote t_s as the number of links in *R-DCN$_s$,*

t_{s-1} as the number of links in *R-DCN$_{s-1}$*,

then the maximum number of R-DCN$_{s-1}$ that will be used to connect to *R-DCN$_s$ is $t_{s-1}+1$*.

Hence, let the number of *R-DCN$_{s-1}$* in a *R-DCN$_s$* subnet be denoted by k_s,

Hence

$$k_s = t_{s-1} + 1 \quad (1)$$

Also let the number of servers in a R-DCN$_s$ subnet be denoted by N_s,

$$N_s = t_{s-1}\,(t_{s-1} + 1) \quad (2)$$

Hence,

$$N_s = k_s * t_{s-1} \quad (3)$$

Equation 3 shows that the total number of servers in a subnet (R-DCN$_s$) is a product of the maximum number *of R-DCN$_{s-1}$* that is used to build it and the number of links in the *R-DCN$_{s-1}$.*

From (3), $N_s = k_s * t_{s-1}$ *for* $s > 0$

$N_s = k_s * t_{s-1} = t_{s-1}\,(t_{s-1} + 1) = (t_{s-1})^2 + t_{s-1}$

By expanding an arbitrary variable $(t_{s-1} + ½)^2$, we have

$(t_{s-1} + ½)^2 = (t_{s-1})^2 + t_{s-1} + ¼$

Therefore, $N_s = (t_{s-1} + ½)^2 - ¼$

Obviously, $N_s = (t_{s-1} + ½)^2 - ¼ > (t_{s-1} + ½)^2 - ½$

Hence,

$$N_s + ½ > (t_{s-1} + ½)^2 \quad (4)$$

Similarly by expanding an arbitrary variable $(t_{s-1} + 1)^2$, we have

$(t_{s-1} + 1)^2 = (t_{s-1})^2 + 2t_{s-1} + 1$

Therefore, $N_s = (t_{s-1} + 1)^2 - t_{s-1} - 1$

Obviously $N_s = (t_{s-1} + 1)^2 - t_{s-1} - 1 < (t_{s-1} + 1)^2 - 1$

Hence,

$$N_s + 1 < (t_{s-1} + 1)^2 \quad (5)$$

Replacing t_{s-1} with l which is the number of links in *R-DCN$_{s-1}$* in equations (4) and (5), we have

$N_s + ½ > (l + ½)^{2k}$ and $N_s + 1 < (l + 1)^{2k}$ respectively for $k > 0$ where k is the scalability factor.

Since $N_s + ½ > (l + ½)^{2k}$ is equivalent to $N_s > (l + ½)^{2k} - ½$ and $N_s + 1 < (l + 1)^{2k}$ is equivalent to $N_s < (l + 1)^{2k} - 1$

Hence,

$$(l + ½)^{2k} - ½ < N_s < (l + 1)^{2k} - 1 \quad (6)$$

Therefore for R-DCN$_s$ with links t_s, the number of servers is bounded as shown in equation (6). The equation shows that the number of servers in a R-DCN scales exponentially as the node degree increases, hence, a highly scalable DCN.

The R-DCN physical architecture with the VLAN segmentation is shown in Fig.7 which is an expansion of Fig. 6 showing a recursive construction in the server to switch port mapping. In the R-DCN physical structure, the servers in one subnet are connected to one another through one of the MLS ports that is dedicated to that subnet. Each server in one subnet is also linked to another server of the same order in all another subnets.

In other words, each of the servers has two links, with one it connects to other servers in the same subnet (intra server connection) and with the other it connects to the other servers of the same order in all other subnets (inter server connection). Apart from the communication that goes on simultaneously in the various subnets, the inter server connection is actually a VLAN connection. This VLAN segmentation of the servers logical isolates them for security and improved network performance.

Increase in subnets results in better broadcast regulation for the connected servers. Also, an increasing the number of links leads to increased redundancy which could lead to latency challenges in the long run.

Together with server virtualization which ultimately improves the network bandwidth and speed, this VLAN segmentation gives each R-DCN$_s$ (subnet) the capacity to efficiently support enterprise web applications (Enterprise Resource Planning, Web Portals, Cloud applications such as software as a service) running on server virtualization connected to each MLS.

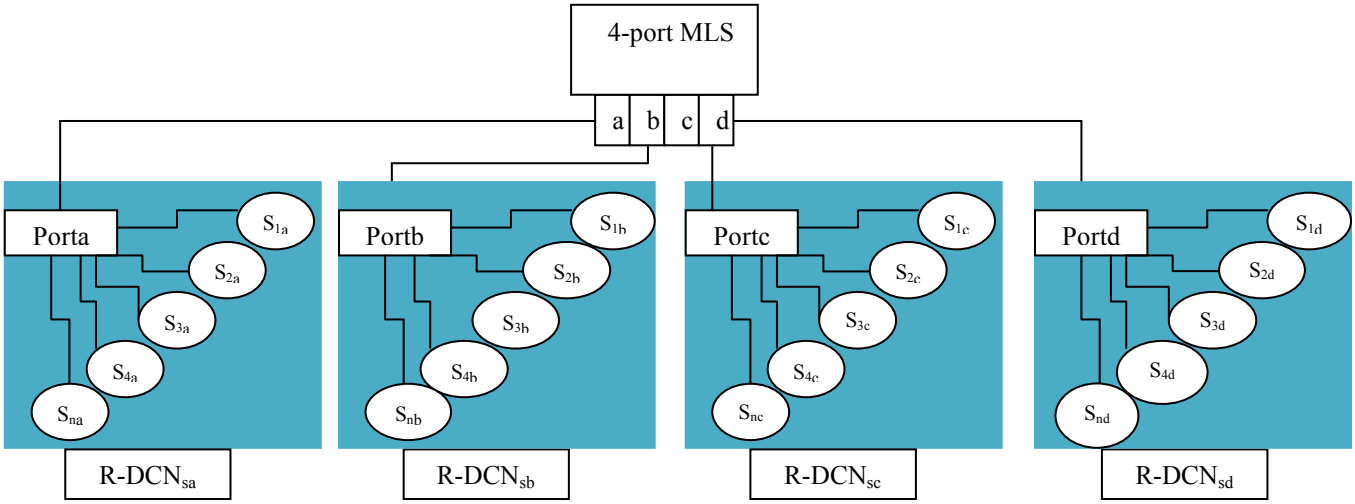

Fig. 6. R-DCN Architecture

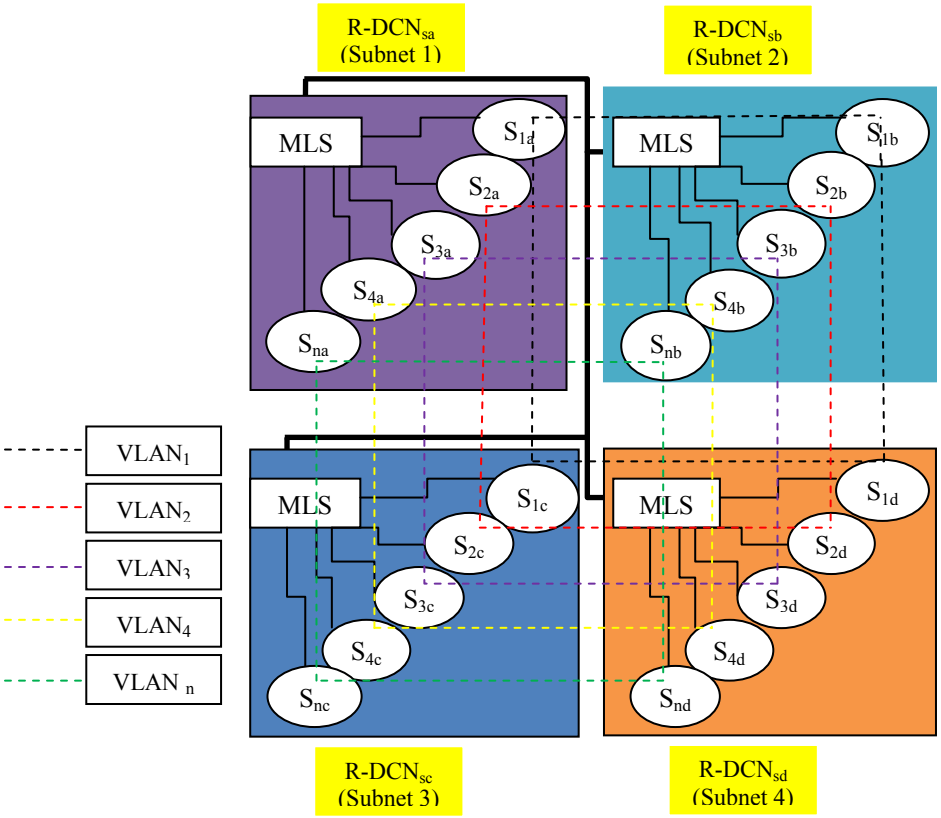

Fig. 7. R-DCN Physical Architecture with VLAN Segmentation.

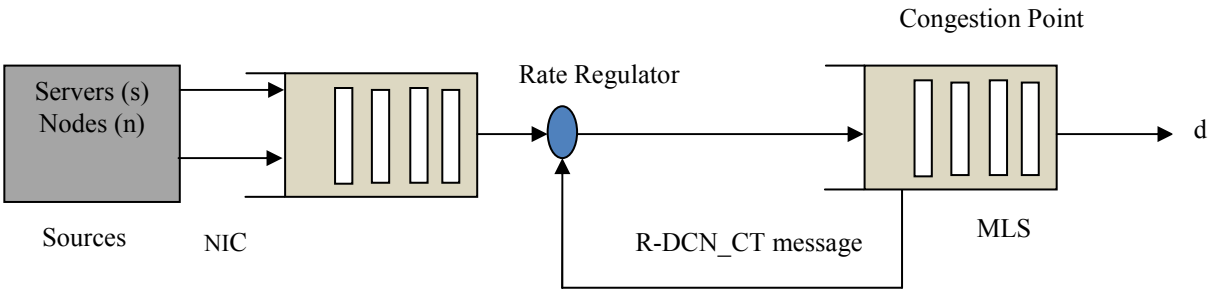

Fig. 8. R-DCN_Traffic Control Model

B. Modeling Traffic Stability for R-DCN.

Request or demand arrives randomly in the MLS, not necessarily in a deterministic fashion. This work assumed that the packet arrival follows the stochastic process such that the packet size is exponentially distributed, and the system is considered as an M/M/1 queuing system. An M/M/1 queue represents the queue length in a system having a single server, where the arrivals are determined by a stochastic process and the job service time has an exponential distribution. The buffer size of the switch (MLS) is of infinite size.

For the system (R-DCN), capacity management and optimum utilization will address broadcast oscillation (congestion) and instability. To address this situation, adapting little's law [24] which takes care of the system response time and scheduling distribution, will optimize traffic flow.

If the average arrival rate per unit time is denoted by λp (pps) and μp is the average service rate per unit time, then from Little's law, the average delay (in seconds), D is given by:

$$D = 1/(\mu p - \lambda p) \quad (7)$$

And the traffic demand, a (referred to as offered load or offered traffic in R-DCNs), is given by

$$a = \lambda p * \mu p \quad (8)$$

The system is considered stable only if $\lambda p < \mu p$. If on the other hand, the average arrivals happen faster than the service completions $(\lambda p > \mu p)$, the queue will grow indefinitely long and the system will not have a stationary distribution (the system is unstable).

Conventionally, in DCN flooding of packets from an active port to destination addresses is done with a compromise to the DCN resources. With a load balancing service [22], beside collision suppression, fair scheduling and sharing of resources which are the optimal feature that will enhance service availability and reliable throughput, the scalability factor in Equation (6) with resource allocation procedures will enhance the utilization of resources by heavy web application servers while maintaining dynamic stability without compromise to other QoS metrics.

C. Analytical Model for R-DCN Traffic Control.

The R-DCN physical architecture as shown in Fig. 6 introduced a traffic control issue which is addressed in this section. The system model for R-DCN traffic control is shown in Fig. 8. R-DCN_CT messages uses 802.1Q tag format[26]. Its content comprises of Mac header (14) bytes, data (46-1500bytes and the CRC checksum (4bytes) [27]. In the model, the R-DCN subnets of the architecture have three major components which are the Nodes, Multilayer Switches,(MLS) and servers. For the purpose of this model, these three VLAN-tags, V*id* are: VLAN-1, VLAN-2, and VLAN-3.

At the core switch (MLS) let the R-DCN carrier sense multiple access with collision detection and Traffic Arbitration Messaging Protocol (CT) be given as R-DCN_CT as shown in Fig. 8.

From Fig. 8, the R-DCN_CT is modelled as a rate-based closed-loop feedback control in the subnet. We assume that servers running virtualization instances are equipped with R-DCN rate regulator (RR) which sharpens traffic (packet/frames). Traffic control at MLS which is the possible congestion point (CP) is a function of monitoring the length of output queues and making traffic control decision. If traffic poll at MLS is high leading to congestion, the MLS advertises signals to sources using R-DCN_CT messages. The R-DCN_CT messages contain details for the sources to adjust their flow rates. The sources reset to the received R-DCN_CT and updates the rate of its regulator.

Other assumptions that were made in the R-DCN analytical model design:

- The R-DCN_CT messages are packet advertised signals sent through MLS in the subnets
- The messages are VLAN-tagged to ensure coexistence and interoperability in the subnets.
- Owing to the short diameter of the R-DCN subnets, the propagation delay/latency is very small.

Links are assumed to be 40GBps in capacity and the MLS is a combined input output queued switch having buffers of First-In First-Out (FIFO) output queued.

- Parameters such as R-DCN_CT message, $Qsc_{max,}$ etc, can be determined with Etheral wireshark tool based on real measurement [28].
- Let equilibrium or stability level be Q_{eq} set at the MLS. This is the optimized packet size that should be in queue.
- Let $q_{off\ (t)}$ indicates the instantaneous web load while $q_{delta(t)}$ indicates the rate of change of the web load. The weighted sum is an approximate prediction of the future web load.

A threshold is set to indicate tolerable congestion levels on the R-DCN link given by Qsc_{max}. The MLS basically counts the number of arrivals *(A)* and departure packets *(d)* and samples the incoming packets with a steady state probability *Ps*.

When the packet is sampled, the MLS determine the congestion level on the R-DCN link and may send R-DCN_CT message to the source terminal of the sampled packet. If the traffic congestion is extreme, the MLS may send a 'PAUSE' message. The key details of the R-DCN_CT message are the source Ethernet type and link congestion measure ei as well as destination congestion point ID (CPID) and capacity of the link C.

Now the Ethernet-Type tells the MLS and sources about the R-DCN_CT. The CPID is the id for link congestion (MAC address of MLS interface). *ei* is the buffer link congestion measure fed back to the source.

From Fig. 8, the key measure of congestion on a link is *ei* which consists of a weighted sum of the instantaneous queue offset and queue variation over the last sampling interval. This is given by the equation (9) and (10);

$$e_i = q_{off}(t) - Wq_{delta} \quad (9)$$

Where Wq_{delta} = queue weight given by $W(q_a - q_d)$, and
$q_{off}(t)$ = instantaneous queue offset given by
$q_{off}(t) = q(t) - Q_{eq}$.
Hence,

$$e_i = (q(t) - Q_{eq}) - W(q_a - q_d) \quad (10)$$

q_{delta} is the queue variation over the last sampling interval and is defined as the difference in the number of packets that arrived q_a and the number of packets that were served q_d since the last sampling event. From Fig. 8, the algorithm showing traffic control Equilibrium Level (Q_{eq}) with possible event in R-DCN subnets is presented in Algorithm I.

Algorithm I: R-DCN traffic control for Equilibrium Level (Q_{eq}).

*R-DCN (TC) /*Traffic control decision at Qeq*/*
1 *If (q(t) < Qeq) && (q_a = = q_d), {ei > 0} return;*
2 *If (q(t) < Qeq) && (q_a >> q_d), {ei < 0} return;*
3 *If (q(t) > Qeq) && (q_a = = q_d), {ei < 0} return;*
4 *If (q(t) > Qeq) && (q_a << q_d), {ei > 0} return;*
5 *Return R-DCN (TC);*

The first line in the algorithm 4 is the case where the queue length is short and the sources can increase their rates. Line 2 is the case where even though the queue length is small, it is increasing, and as the traffic congestion is building up, the sources are signaled to decrease their sending rates. In line 3, the large queue indicates that the links are congested, and the sources are signaled to decrease their sending rates. In line 4, even though the queue is large at the moment, it is decreasing and so the sources are signaled to increase their rates.
From the model of Fig. 8, the sources adjusts their rates using the enhanced Additive and Multiplicative Decrease (AIMD) algorithm which has been proven to be sufficient, efficient and shows fairness. Also, the algorithm for R-DCN traffic control for feedback conditions is given in Algorithm II. There are two sections in the algorithm, one is when the traffic packet does not contain rate regulator, and the other is when the traffic packet does contain rate regulator. Note that under the same traffic conditions in both sections, the feedback messages (R-DCN_CT messages) sending are the same.

Algorithm II: R-DCN Traffic Control for Feedback Conditions

If the traffic packet does not contain rate regulator tag:
If (q(t) < Q_{eq}) {No R-DCN_CT message sent}; return;}
If (Q_{eq} < q(t) ≤ Q_s) {Normal R-DCN_CT message sent}; return;}
If (q(t) > Q_s) {R-DCN_CT message sent}; return;}

If the traffic packet does contain rate regulator tag:
If (q(t) < Q_{eq}) && (CPID = = MLS CPID) {No R-DCN_CT message sent}; return;}
If (Q_{eq} < q(t) ≤ Q_s) {Normal R-DCN_CT message sent}; return;}
If (q(t) > Q_s) {R-DCN_CT message sent}; return;

Besides, Algorithm III shows the security algorithm based on VLAN tagging. On receipt of a packet, an MLS processes it and performs the flow look up for matching or no matching. The flow table is checked for a matching flow entry. An MLS Open flow security algorithm in relation to VLAN is shown in algorithm III. The algorithm has two parts, one for setting VLAN ID and the other for setting the VLAN priority. From the algorithm, in the first case if no VLAN is present, then a new header is added and is the VLANID is specified while the VLAN priority is set zero. But if header already exists, the VLAN ID is replaced with a specified value. In the second part which is for setting VLAN priority, after adding a new header if not present, the VLAN priority is set, while VLAN ID is set to zero. But if the header already exists, the priority field is replaced with a specified value.

Algorithm III: MLS Security Algorithm vis-à-vis VLAN
Part 1: /* Set VLAN ID*/

If no VLAN is present; Add a new header;
Specify its VLAN ID;
Set VLAN priority to zero;
Else if VLAN header is already exists;
Replace the VLAN ID with the specified value;
Part 11: /* Set VLAN Priority*/
If no VLAN is present; Add a new header;
Set its VLAN priority;
Set VLAN ID to zero;
Else if VLAN header is already exists;
Replace the priority field with the specified value;

IV. R-DCN MODEL VALIDATIONS

A. Simulation Design.

For further validation the proposed R-DCN architecture, the design parameters obtained from experimental testbeds (UNN DCN) [21] was used to develop a generic template in OPNET IT guru (a simulation tool) [25]. The tool has a rich library of object palette for configurable vendor devices. However, based on the experimental measurement carried out on the testbed, the metrics for performance evaluations of R-DCN in comparison with two other data center network architectures viz: DCell and BCube were generated. On the generic OPNET template shown in Fig. 9, three scenarios were created, one for R-DCN, one for DCell, and one for BCube.

For each of the scenarios, the attributes of the three architectures were configured on the template and the simulation was run. Afterwards, the OPNET engine generates the respective data for each of the QoS investigated on in Fig. 9

Essentially, R-DCN, BCube and DCell share several similar design principles such as providing high capacity between all

servers and embracing end-systems. R-DCN uses only a low-end switch and provides better one-to-x support at the cost of multi-ports per-server and is able to decouple IP address and server location by introducing a directory service. It also uses algorithm I, II and III (as shown above), traffic control with VLAN as well as feedback mechanisms, server virtualization, and convergence with randomization. BCube uses active probing for load-balancing. DCell employed neighbour maintenance, link state management, prototyping, and fault tolerance schemes in its design philosophy.

Three experiments were carried out to study parameters such as throughput, fault-tolerance, network capacity, utilization, latency, service availability, scalability and clustering effects of R-DCN. The R-DCN responses with respect to these parameters were compared against BCube and DCell data center architectures.

Before the simulation, link consistence tests were carried out randomly to ascertain any possibility of failure in all cases. A randomly selected nodes and servers routes packets in a bidirectional way from the access layer to the core layer and vice versa. In context, an investigation on both the path failure ratio and the average path length for the found paths was carried out and all the links found satisfactory. In Fig. 9, a node consistence test based on the animation packet flow was carried out. All links and nodes passed the test in all cases. Also, Fig. 9 shows OPNET screenshot for simulation sequence used in the analysis.

In all the simulations, we enabled the essential attributes for each of the three scenarios (R-DCN, DCell, and BCube) on the template beginning the simulation at 11:31:35 Sun Oct 28 2012. The Simulation completed successfully and the results collated in the global and object statistics reports. The simulation gave total events = 11665328, Average Speed (176162 events/sec), Time Elapsed (1 min. 6 sec.), Simulated (10 hr. 0 min. 0 sec.) and Simulation Log: 3130 entries. The simulation plots of the three DCN architectures under study (R-DCN, DCell and BCube) are shown in the plots from Figs. 5.20, to 5.25.

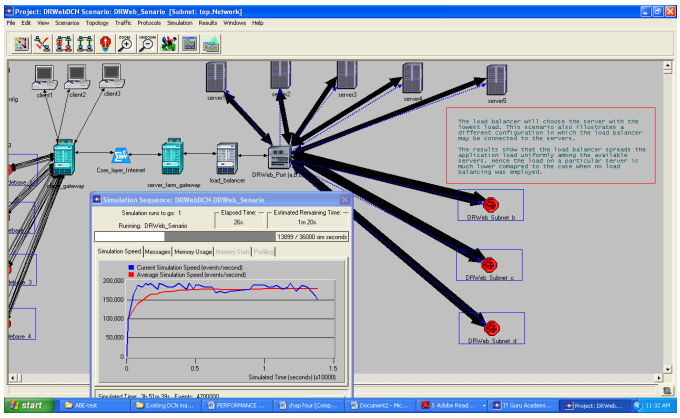

Fig. 9. OPNET Screenshot for Consistence Test with Simulation Analysis

B. *Performance Evaluations*

From Fig. 9, the simulation testbed for R-DCN validation was developed. We generated the following metrics for the three scenarios which were analyzed also.

1) The Throughput Response (R-DCN, BCube and DCell)

Throughput being the data quantity transmitted correctly starting from the source to the destination within a specified time (seconds) is quantified with varied factors including packet collisions, obstructions between nodes/terminal devices between the access layer and the core layer and more importantly at the core layer of the R-DCN. It is measure in bytes/sec or bits per secs. Fig. 10 shows the throughput response of R-DCN, BCube and DCell respectively. During the simulation, throughput as a global statistics was being measured and compared. BCube had an initial interesting throughput response which was not sustained (28.65%) while R-DCN maintained a relatively stable throughput response (36.98%). DCell gave a throughput response (34.38%) that is better compared with Bcube. The average throughput in a network with R-DCN has highest throughput compared with the average throughput in a network with BCube and DCell. The main reason for this is the enhanced controlled algorithms introduced in R-DCN design.

Again, in all cases, the introduction of a load balancer is expected to balance the traffic at the core, but it was observed that the network model of R-DCN in its topological layout had a significant effect on the throughput. Again, this work attributes this observation to the fact that the two-tier topology of R-DCN is communicating on the basis of reduced policy implementation at the expunged aggregation tier of the BCube and DCell models. This makes for efficiency and as such the total load of the network is divided among the two-tier only on 40% (access layer): 60% (core) leading to lesser collisions and lesser packet drops which could likely occur.

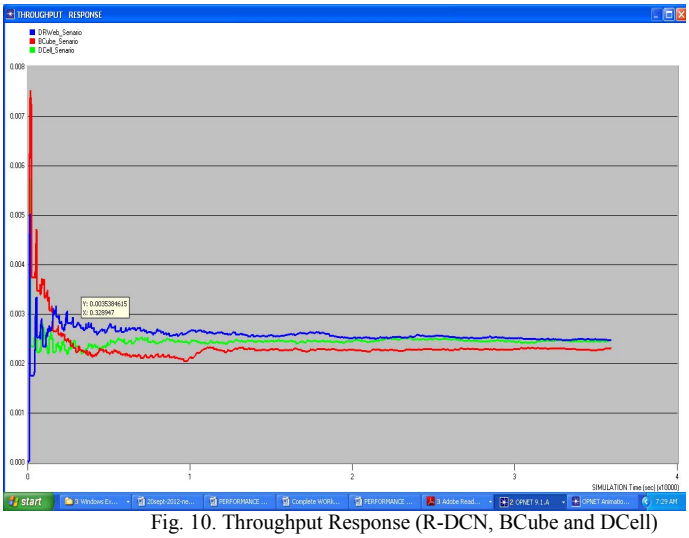

Fig. 10. Throughput Response (R-DCN, BCube and DCell)

2) Service Availability.

In this research, service availability is a function of fault-tolerance and failure proof. In the simulation experiment, a TCP connection between servers at the core layer and the access layer was setup. Different network services including databases, E-mailing, web browsing, FTP, http and other TCP & UDP services were introduced on the three scenarios. To study the performance under link failures (downtime), we momentarily unplugged the link in R-DCN subnet_3 at time 34s and then plugged in at time 42s in the simulation panel. We also shut down some selected servers at time 104s in order to assess the impact of node failures. After both failures, the routing path quickly converged and the path returned to the original status. The TCP traffic was maintained as well and the CPU utilizations are about 40%, and 45%, for sender, and receiver, respectively. This is a different scenario for BCube and DCell. In context, this work makes two observations from Fig. 11. First, R-DCN is a very resilient to both failures. The TCP throughput is recovered to the best value after only a few seconds compared to BCube and DCell. Second, the R-DCN implementation detects link failures and node failures much faster than BCube and DCell because it uses medium sensing network capacity, its fault tolerant and suppression routing. Other factors could include the R-DCN logical isolation of nodes and its analytical traffic control scheme. The plot in Fig. 11 shows that R-DCN has more service availability (51.40%) initially compared with Bcube (32.50%) and DCell (16.01%) on the flow traffic.

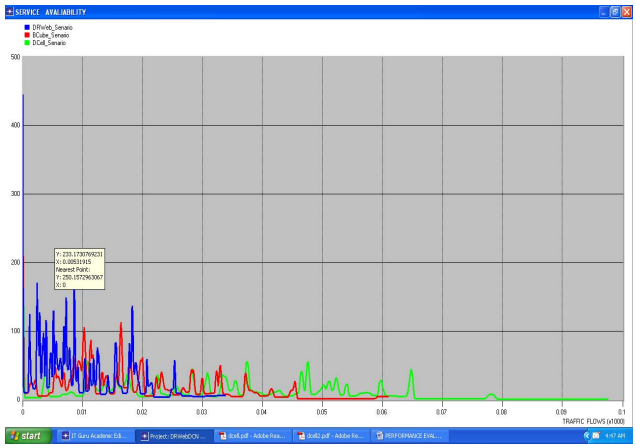

Fig. 11. Service Availability (R-DCN, BCube and DCell)

3) Latency Effects.

In this case, an analysis on the network behavior with respect to the three scenarios was carried out. Fig. 12 shows the end-to-end latency result of the R-DCN two-tier topology with the three-tier BCube and DCell topologies. As depicted in the plots, the latency response shows great similarity but for the R-DCN the latency response was about 0.000106 millisec (31.98%) which is much lower compared with that of Bcube (33.90%) and DCell (34.1%). The reason for this is that in R-DCN, there is additional traffic optimization by the control algorithm as well as the enhanced network topology which reduces the transmission time between the access and the core layer even when the links are busty and busy. The core layer is highly redundant with little routing policy in R-DCN and as such can easily take packets from the access layer with very little wait states.

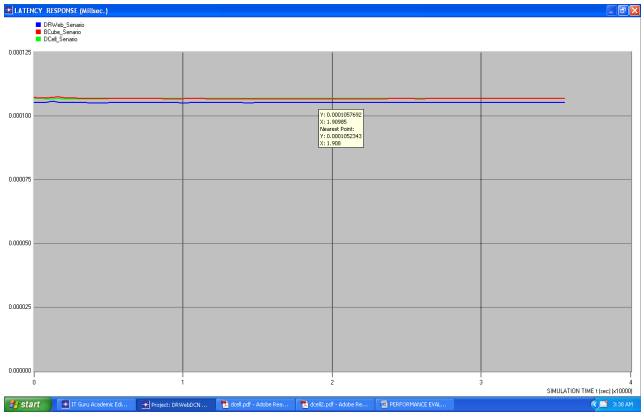

Fig. 12. Latency Effects (R-DCN, BCube and DCell)

4) Utilization Response

Fig. 13 shows resource utilization responses elicited from the three case scenarios in the DCN design. Generally, in DCN environment, latency, throughput, server resources utilization, and link bandwidth are vital resources considered in congestion management. With the traffic load sources in the R-DCN topology used in this research, an initial gradient was established by BCube before resource allocations was fairly distributed. With a connection request, feasible regions of resource allocation are first established. Resource utilization in BCube regions is quite high at the beginning but later became the lowest. R-DCN and DCell have a fairly uniform resource utilization, even though R-DCN maintained a higher resource utilization that DCell for most of the period. However, among the three architectures, DCell has the best optimal server resources utilization followed by R-DCN and then BCube. Since average utilization will have prolonged life span implications on the core of the DCN design, Dcell (27.84%) has edge over R-DCN (31.96%) and Bcube (40.21%) as it has a lower resource utilization response initially. After a while they all followed a similar pattern as the load increases.

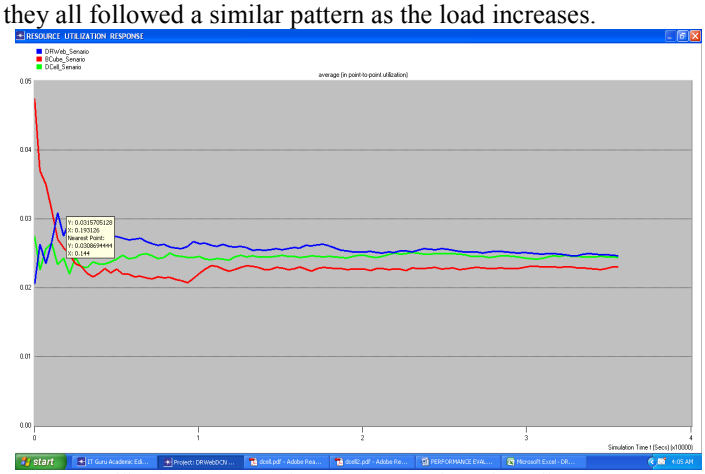

Fig. 13. Utilization Response plot (R-DCN, BCube and DCell)

5) Server Clustering Effects on Utilization.

The R-DCN was configured with Solaris based buffer adjustment for a buffer size of 49152bytes, clock based delayed ACK mechanism, Round Trip Time (RTT) gain of 0.125 and deviation gain of 0.25 with enabled fast retransmit and unlimited active connection threshold in the server clustering deployment. As shown in Fig. 14, R-DCN provides much better support for efficient access layer and core layer one-to-x traffic utilization compared with BCube and DCell. R-DCN provides the lowest aggregate bottleneck throughput since the server clustering enhances utilization leading to a better throughput. The server clustering yields moderate utilization as a result of the above implementations. Without server clustering, the path length and the number of wires in BCube and DCell are larger than those of R-DCN. Moreover, R-DCN does not degrade gracefully as server resources increases, but will need upgrade to support advanced routing and packet forwarding functionalities.

R-DCN also being better than BCube and DCell in terms of clustering yields a moderate utilization cycles. With virtualization, TAMP routing and VLAN instances, R-DCN builds complete graphs at each subnet level, resulting to a doubly exponential growth with moderate CPU utilization cycles. As a result, R-DCN can handle web application integrations with high flexibility and with lesser resource drain. This is not true for BCube and DCell.

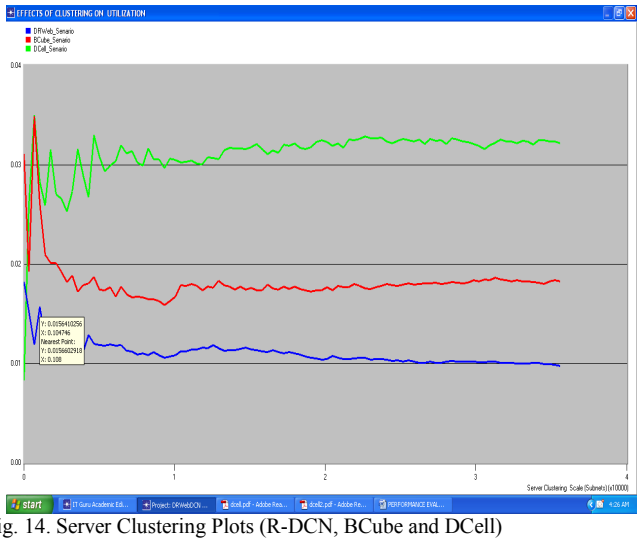

Fig. 14. Server Clustering Plots (R-DCN, BCube and DCell)

6) Scalability Response (R-DCN, BCube and DCell)

Fig. 15 shows the scalability response for R-DCN, DCell and BCube. Scalability of a DCN is defined by its expansibility without complications with respect to future demand for an increase. In other words, a highly scalable network can be expanded to accommodate increasing number of users or demand without any complication. From the Fig. 15, the three architectures have similar scalability response in as shown in a linear. It is difficult to say which one has a better scalability response between DCell and R-DCN in that at some points, however, DCell with 33.35% is better and at some other points, R-DCN with 33.65% is better; but obviously their scalability is better than BCube with 32.98%. This can be attributed to the similarity in the server interconnection concept between the two architectures (R-DCN and DCell).

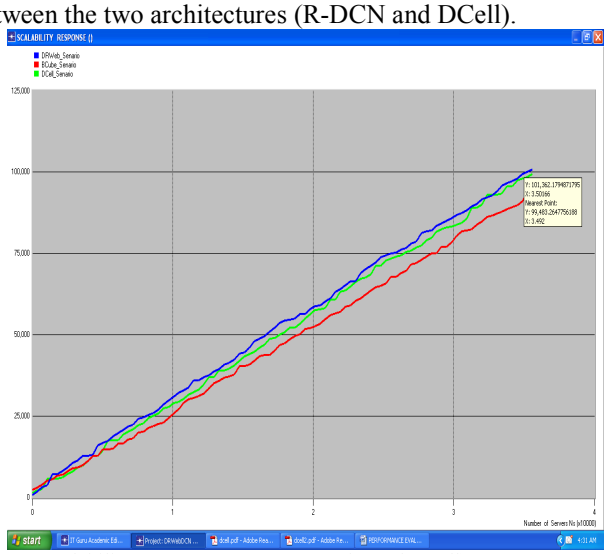

Fig. 15. Scalability Response (R-DCN, BCube and DCell)

V. Conclusion and Future Work

In this paper has presented Reengineered DCN (R-DCN) for efficient web application integration in enterprise organizations. Apart from the literature review carried out on the various architectures, this work also carried out a study on typical enterprise traditional DCN (Swift Network DCN and UNN DCN) [21]. A testbed validation of this research has been carried out in a previous work [21]. It was showed that the traditional DCNs have limitations with respect to throughput, latency, scalability, efficiency in web application integration, etc. R-DCN was therefore designed to take care of them. Mathematical models for scalability, logical isolation of R-DCN architecture, traffic control issues, security algorithm, etc, were some of the key design attributes of the proposed R-DCN. From a carefully designed simulation analysis, our discovery showed that R-DCN performed relatively well in relation to DCell and Bcube in terms of the evaluation metrics. The results of R-DCN simulation from the managed scenarios showed that it will handle web application integration efficiently which is our main goal for the reengineering of traditional DCNs.

Again we discovered that R-DCN shows a better performance in terms of throughput response, service availability, scalability, network latency, etc, than DCell and BCube architectures on the basis of the simulation context. Our conclusion therefore, is that the proposed R-DCN architecture will be very efficient, scalable, cost effective, service-oriented, and responsive to business needs, with rapid service delivery, and one that can provide tighter alignment with business goals. Hence, this paper recommends the R-DCN to enterprise organizations for greater efficiency in web application integration vis-à-vis their DCNs. Future work will show a detailed validation using a cloud testbed and CloudSim Simulator.

References

[1] K.C.Okafor, and T.A.Nwodoh, "Synthesis VLAN Approach to Congestion Management in DataCenter Ethernet Networks",International Journal of Electrical &Telecommunication Systems Research, Vol5,No.5, 2011.

[2] T. Hoff, "Google Architecture" [online]. Avaialble: http://highscalability.com/googlearchitecture, Jul 2007.

[3] L. Rabbe, "Powering the Yahoo! network"[online].Available: http://yodel.yahoo.com /2006/11/27/powering-the-yahoo-network, Nov 2006.

[4] A. Carter, "Do It Green: Media Interview with Michael Manos" [online]. Available: *http://edge.technet.com/Media/Doing-IT-Green/*, Dec 2007

[5] M. Al-Fares, A. Loukissas and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture", In *Proceedings of ACM SIGCOMM'08*, Aug 2008, pp 63-74.

[6] S. Ghemawat, H. Gobio, and S. Leungm, "The Google File System", In *Proceedings of ACM SOSP'03*, 2003.

[7] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", In *Proceedings of OSDI'04*, 2004.

[8] M. Isard, M. Budiu, Y. Yu and etc., "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks", In *Proceedings of ACM EuroSys'07*, 2007.

[9] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, R.E. Gruber. "Bigtable: A Distributed Storage System for Structured Data", in: OSDI '06: Proceedings of the 7th Symposium on Operating Systems Design and Implementation, Berkeley, CA, USA, 2006. USENIX Association, pp. 205–218.

[10] J. Dean, S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters", in: OSDI '04: Proceedings of the 5th Symposium on Operating Systems Design and Implementation, 2004, pp. 137–150.

[11] S. Ghemawat, H. Gobioff, S.-T. Leung, "The google file system, SOSP '03: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles", ACM Press, New York, NY, USA, 2003, pp. 29–43.

[12] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi*y*, Yongguang Zhang, Songwu Lu*z*, "DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers", in Proc. of SIGGCOM 2008.

[13] Yong Liao, Jiangtao Yin, Dong Yin, Lixin Gao, "DPillar: Dual-port server interconnection network for large scale data centers", Computer Networks, March 2012, pp 2132-2147.

[14] L. Barroso, J. Dean, and U. HÄolzle. "Web Search for a Planet: The Google Cluster Architecture", *IEEE Micro*, March-April 2003.

[15] S. Ghemawat, H. Gobio®, and S. Leung. "The Google File System", In *ACM SOSP'03*, 2003.

[16] K.C Okafor, O.U Oparaku, F.N Ugwoke,Udeze.C.C, "An investigation into EETACP Resource Allocation from Multiple Pools in Cloud DataCenter Servers Using Discrete Event Simulation", (Unpublished)

[17] A. Greenberg, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. "Towards a next generation data center architecture: scalability and commoditization", in PRESTO '08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow, pages 57–62, New York, NY, USA, 2008. ACM.

[18] C. Guo, G. Lu, D. Li, H.Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. "BCube: a high performance, server-centric network architecture for modular data centers", in SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication, pages 63–74, New York, NY, USA, 2009. ACM.

[19] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang. "Mdcube: a high performance network structure for modular data center interconnection", in CoNEXT '09: Proceedings of the 5th international conference on Emerging networking experiments and technologies, pages 25–36, New York, NY, USA, 2009. ACM.

[20] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. "Vl2: a scalable and flexible data center network", in Proceedings of the ACM SIGCOMM 2009 conference on data communication, pages 51–62, New York, NY, USA, 2009.

[21] Udeze C.C, Okafor Kennedy.C., Ugwoke F. N, U.M.I.Onwusuru, "An Evaluation of Legacy 3-Tier DataCenter Networks for Enterprise Computing Using Mathematical Induction Algorithm", Computing, Information Systems, Development Informatics & Allied Research Vol. 4 No. 4 December, 2013.Pp1-10.

[22] K.C. Okafor, Ugwoke, F. N, Udeze, C. C, Okezie, C. C, O. U Oparaku, "*Gateway Load Balancing Service In Cloud Data Centre Environments Using Throughput Metric Index",* Accepted in American Journal of Computation, Communication and Control, AASCIT, USA, 2014; Vol.1 , No. 1, April 2014, Pp.8-17, 2014, Online: http://www.aascit.org/journal/archive2?journalId=901&paperId=324

[23] Udeze Chidiebele.C, "Re-Engineering DataCenter Networks for Efficient Web Application Integration in Enterprise Organisations", PhD thesis, Unizik, February, 2013.

[24] M. Veeraraghavan, Derivation of Little's Law, Feb. 10, 2004: Online: http://www.ece.virginia.edu/mv/edu/715/lectures/littles-law/littles-law.pdf

[25] www.riverbed.com-Online: https://supportstaging.riverbed.com/bin/support/static//doc/opnet/17.5.A/online/itguru_17.5.PL5/Models/wwhelp/wwhimpl/common/html/wwhelp.htm#href=Applications_Model_desc.02.05.html&single=true

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

[27] Lammle, T. Cisco Certified Network Associate study Guide, Sixth Edition, ISBN: 978-0-470-11008-9, 2007.

[28] A. Jardosh, K. Ramachandran, K. Almeroth, and E. Belding-Royer. "Understanding Link-Layer Behavior in Highly Con-gested IEEE 802.11b Wireless Networks". In *Proceedings of ACM SIGCOMM Workshop E-WIND*, Philadelphia, PA, August 2005.