American Journal of Computation, Communication and Control
2014; 1(1): 8-17
Published online April 10, 2014 (http://www.aascit.org/journal/ajccc)

Gateway load balancing service in cloud data centre environments using throughput metric index

K. C Okafor[1], Ugwoke, F. N[2], Udeze, C. C[3], Okezie, C. C[3], O. U Oparaku[1]

[1]Department of Electronic Engineering, University of Nigeria, Nsukka, UNN, Nigeria
[2]Department of Computer Science, Michael Opara University of Agriculture, Umudike, Nigeria
[3]Department of Electronics and Computer Engineering, Unizik, Awka, Nigeria

Email address
arissyncline@yahoo.com(Ugwoke, F. N)

Keywords
Cloud,
Datacenter,
Throughput,
Web,
Service,
Load,
Access Layer,
Simulation

Received: March 06, 2014
Revised: March 28, 2014
Accepted: March 29, 2014

Citation
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. *American Journal of Computation, Communication and Control.* Vol. 1, No. 1, 2014, pp. 8-17.

Abstract
In cloud data centre designs, as a result of high density traffic transactions, introducing a gateway load balancer (GLB) to improve server performance considering the demand of clients, will greatly offer robust fault tolerance, while enhancing performance at large. This paper developed, and analysed the performance of a core layer network simulation which hosts a typical http web service while outlining the advantages of load balancing service. This reflects in the throughput response of a two case scenario: *Cloud DCN failure recovery* and *Cloud DCN No_ failure recovery* in a carefully design cloud datacenter setup. From the results of this study, it was observed that the former yielded 99% throughput index while the later offered 97% throughput index. The 3% differential accounts for occasional server downtimes. Consequently, this paper argues that in a data intensive network, any load balancing service will allow load sharing of traffic from access layer endpoints within a common subnet through redundant default gateways, while also providing failure protection. This will offer an additional improvement in a proposed Enterprise Energy Tracking Analytics Cloud Portal (EETACP) in our future work.

1. Introduction

Recently, applications continue to rely increasingly on distributed resources of datacenter networks thereby creating a need for fault tolerance at the core layer. As a result, networking systems in the cloud environment will continue to evolve with increased interface speeds and packet forwarding rates. The underlying complexity of these systems must continue to support virtualization, advanced security mechanism, scalability and stability. Datacenter designs must focus more on the relationship it will have with business processes, rather than on details of balancing access layer traffic, in order to gain competitive advantage from their networked systems.

In the business process of our earlier proposal on EETACP, figure 1 shows its development criteria, but using the public deployment model, a critical issue to be considered for mission critical applications is fault tolerance in the cloud environment.

2. Related Works

While a lot of work have been done on large scale network computing such as datacenter congestion management (TCP/IP networks), and fault tolerant designs, etc [2], [3], [4], [5], [6], [7], and [8], clouds aim to drive the design of the next generation data centres by architecting them as networks of virtual services (hardware, database, user-interface, application logic) so that users can access and deploy applications from anywhere in the world on demand at competitive costs depending on their QoS (Quality of Service) requirements [3]. Developers with innovative ideas for new Internet services no longer require large capital outlays in hardware to deploy their service or human expense to operate it [9].

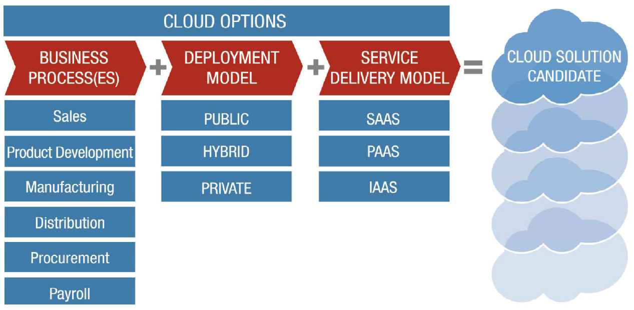

Figure 1. *Cloud Application Framework (Source: Crowe H.L. et al, COSO, 2012).*

In the context of fault tolerance, server, link, switch, rack failures due to hardware, software, and power outage problems presents vulnerabilities. As the network size grows, individual server and switch failures may become the norm rather than exception. Fault tolerance in DCN may requests for both redundancy in physical connectivity and robust mechanisms in protocol design. In all our review studies, failures are quite common in current datacenters [10], [11]. Besides, high network capacity in online infrastructure services will need large amount of network bandwidth to deliver satisfactory runtime performance as well.

Any contribution that will improve the performance index of cloud public delivery model will be widely celebrated. Load balancing is the act of balancing packet load over multiple links to the same remote network [12]. It is a function that spreads the traffic over multiple devices and circuits, rather than sending it all through a single device and circuit. As studied in [12], Open Shortest Path First (OSPF), (a link-state, hierarchical routing algorithm derived from an earlier version of the Intermediate System-Intermediate System (IS-IS) protocol), Enhanced Interior gateway Routing Protocol (IGRP), Interior Gateway Protocol (IGP), and IP all offers load balancing capabilitities which was not discussed in our earlier work in [13].

The paper is organized as follows: In section II, Related works is discussed. We state a hypothesis; describe the general system model with relevant assumptions for the cloud environment. Also, we discuss the advantages of GLB in cloud DCN physical architecture. In section III, an analytical model for throughput metric is presented. In Section IV, We actually present a simulation design of a two case scenario for Cloud DCN fault tolerance system. Section V gives the simulation results to validate our hypothesis. The paper ends with the conclusions and future directions.

2.1. Research Hypothesis, System Design and Assumptions

A. Research Hypothesis

There is a significant variation in the throughput index of a cloud datacenter network with a failure recovery service compared to the one with no failure recovery service.

B. System Model and Description

Consider figure 2, the users X_1................X_{n+1} utilize a single *gateway* to reach the Internet. In this model, the gateway *f(x)* and *f(y)* are multilayer series switches where *f(x)* represents the client gateway, and *f(y)* represents the server gateway; however, a Layer-3 router can serve same purpose as it can be used interchangeably with *multilayer switch*. The server gateway represents a single point of failure on this network. In the absence of a fault tolerate gateway, if that gateway fails, users will lose access to all resources beyond that gateway. This lack of redundancy in such networks is unacceptable on business-critical systems that require maximum uptime. However, this required a solution transparent to the end user (or host device) as shown in figure 2

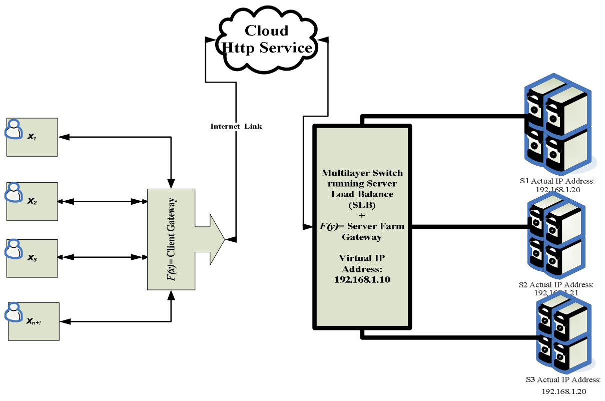
Cloud Http Service
Internet Link
X1
X2
X3
Xn+1
F(x)= Client Gateway
Multilayer Switch running Server Load Balance (SLB) + F(y)= Server Farm Gateway
Virtual IP Address: 192.168.1.10
S1 Actual IP Address: 192.168.1.20
S2 Actual IP Address: 192.168.1.21
S3 Actual IP Address: 192.168.1.20

Figure 2. *System Model for Cloud Datacenter network.*

As shown in figure 2, detailed discussion on server load balancing, otherwise referred to as clustering service is discussed below.

2.2. Server Load Balancing (SLB)

Intelligent Cisco routers/multilayer switches supports the following protocols for datacenter load stabilization in their IOS softwares, viz

i. Hot Standby Router Protocol (HSRP)
ii. Virtual Router Redundancy Protocol (VRRP)
iii. Gateway Load Balancing Protocol (GLBP)

HSRP, VRRP, and GLBP provide gateway redundancy for clients. While HSRP and VRRP do provide redundant gateways for fault tolerance, they do not provide load-balancing between those gateways which is a serious limitation. The SLB service in figure 2 allows a router to apply a virtual IP address (Assuming IP address 192.168.1.10) to a group of servers S_1, S_2,S_3. All of the servers are configured identically (with the exception of their IP addresses), and provide the same function. Having multiple servers for the proposed EETACP, etc allows for both redundancy and load-balancing. As shown in figure 2, clients point to a single virtual IP address to access the server farm. The client is unaware of which server it is truly connecting to. If a specific server fails, the server farm can stay operational. Individual servers can be brought down for repair or maintenance, and the server farm can stay functional. Assume the servers are Web servers hosting the proposed EETACP. To access the Web resource, users will connect to the Virtual IP address of 192.168.1.10. The multilayer switch intercepts this packet, and redirects it to one of the physical servers inside the server farm. In essence, the multilayer switch is functioning as a Virtual Server. For the SLB, there are two load balancing methods available viz:

i. Weighted Round Robin: In this case, the traffic is forwarded to the physical servers in a round robin fashion. However, servers with a higher *weight* are assigned more traffic. This is the default method as used in this work.

Weighted Least Connections: In this case, the traffic is assigned to the server with the least amount of current connections. Appendix 2 shows the SLB Configuration for Cloud DCN on a Cisco MLS

2.3. Advantages of GLB in Cloud DCN

Some identified advantages of GLB in cloud computing environment include:

i. Efficient use of network resources: multiple paths upstream from the gateways can be utilized simultaneously.
ii. Higher availability: GLBP offers enhanced redundancy eliminating single point of failure of the first-hop gateway. An enhanced object-tracking feature can be used with GLBP to ensure the redundancy implementation mirrors network capabilities. This same feature is also available for HSRP and VRRP.
iii. Automatic load balancing: Off-net traffic is shared among available gateways on a per-host basis, according to the defined load-balancing algorithm.
iv. Lower administration costs: Since all hosts on a subnet can use a common default gateway while load balancing is still achieved, administration of multiple groups and gateways is unnecessary.
v. Simpler Access-layer design**:** More efficient use of resources is now possible without configuring additional VLANs and subnets. GLBP can be used if IP hosts on the LAN have a default gateway configured or learned via DHCP. It allows them to send packets to hosts on other network segments while balancing their traffic among multiple gateways.

2.4. Design Goals

In designing a fault tolerant Cloud datacenter, the main goals is to maintain high throughput with near zero downtime. It should be stable and robust. In the following, goals are explained in details [2]:

- High Throughput: Since the demand for data exchange and resources in cloud environment is enormously high compared with other networks, throughput maximization is indispensable, and this is characterized by maximized link utilization.
- Stability: The stability of a system in general depends on the control target [2]. Since, server centric datacenters are involved in high speed computations, our design objective must consider the respective individual flows and convergence rates of the links in active states.
- Low Queuing Delays: Since, the servers supports and runs mission critical applications, the higher the throughput, the higher the link utilization which often leads to long queuing delays. As such, to avoid or maintain the delays within the lowest threshold while achieving, high utilization, the load balancer must be configured to optimize these variables.

2.5. Assumptions/Design Specifications

Following the block diagram overview of the cloud DCN model shown in figure 2, our design will focus on the two layers: access layer and GLB/ speed redundancy layer. Recall that MLS is the major component in the GLB/speed redundancy layer, while servers interconnected through gateway, are the major components of the GLB/Speed core layer. The cloud DCN port architectural model overview is shown in figure 3. This will facilitate the understanding of the model specifications described in [14]. The model specifications are as follows:

- Let C_DCN be an acronym chosen for the Cloud DCN. C_DCN was designed to have four subnets (subnet 1-4) which were called C_DCN_{sa}, C_DCN_{sb}, C_DCN_{sc}, C_DCN_{sd} interconnected as shown in figure 3, where s is a subnet factor such that s > 0. Each C_DCN uses High Performance Computing

(HPC) servers and a Multi-Protocol Label Switch (MLS) layered in linearly defined architecture. Since our designing of datacenter network is for efficient server load balancing and application integration, we will need one (4-port) MLS switch and few servers, hence, the choice of four subnets. Virtual server instances running on the HPC servers made up for further need of hardware servers in the network.

- Servers in C DCN_s are connected to MLS port of the load balancer corresponding to it, and owing to the running virtual instances V*i*, a commodity 4-port switch with 40GB/s per port serve the design purpose. Also, each of the C DCN_s is interconnected to each other through the MLS switch ports.
- The virtualized server used in this work has two ports for redundancy (in Gigabytes). Each server is assigned a 2-tuple [a_1, a_0] in consonance with its ports (a_1, a_0 are the redundant factors) together with a VLAN ID *(1 to 1005*).
- Cisco Ws-C3560-44Ps-E IOS version 12.2 was the MLS used in this work, hence, the number *1005* is the maximum number of VLAN that can be created in it. The switch is a multilayer commodity switch that has a load balancing capability. This capability together with its VLAN capability was leveraged upon to improve the overall Cloud DCN stability.
- Each server has its interface links in Cloud DCN_s. One connects to an MLS, and other servers connects as well but all segmented within their subnets its VLAN segmentation, see figure 4.
- C-DCN_s servers have virtual instances running on it and are fully connected with every other virtual node in the architecture.

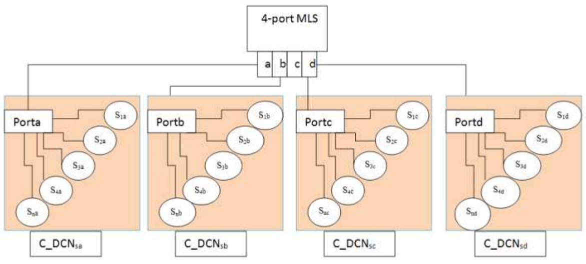

Figure 3. *Cloud DCN Port Architectural Model.*

3. Analytical Algorithms and Characterizations

3.1. Cloud-DCN Construction Algorithm

The C-DCN recursive construction algorithm has two sections. The first section checks whether C-DCN_s is constructed. If so, it connects all the n nodes to a corresponding multi-label switch (MLS) port and ends the recursion. The second section interconnects the servers to the corresponding switch port and any two servers are connected with one link. Each server in the C-DCN_s network is connected with 10GB links for all VLAN*id*. The C-DCN physical architecture with the VLAN segmentation is shown in figure 4 while the linear construction algorithm is depicted in Algorithm 1 below. In the C-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.

As such, 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. Together with server virtualization which ultimately improves the network bandwidth and speed, this VLAN segmentation gives each C-DCN_s (subnet) the capacity to efficiently support enterprise web applications (EETACP, Web Portals, Cloud applications such as software as a service) running on server virtualization in each MLS.

```
Algorithm 1: C_DCN Construction Algorithm.
/* l stands for the level of C_DCN_s subnet links,   n is the number of nodes in a C_DCN_s,
pref is the network prefix of C_DCN_s  s s is the number of servers in a C_DCN_s,*/
Build C_DCN_s (l, n, s)
Section I:  /* build C_DCN_s */
If (l = = 0)
For (int i = 0; i < n; i++)  /* where n is=4*/
Connect node [pref, i] to its switch;
Return;
Section II:  /*build C_DCN_s servers*/
For (int i = 0; i < s; i++)
Build C_DCN_s ([pref, i], s)
Connect C_DCN_s (s) to its switch;
Return;
```

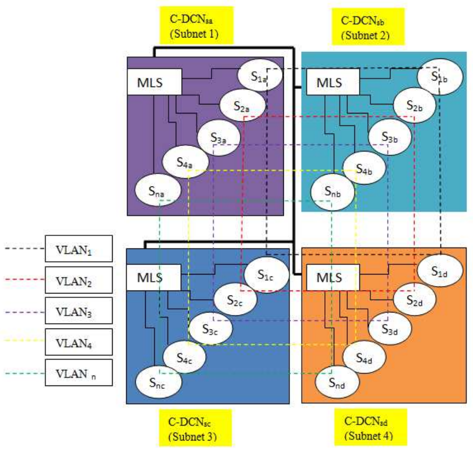

Figure 4. *Cloud DCN Physical Architecture with VLAN Segmentation.*

3.2. Logical Isolation of Cloud DCN Architecture

The application of VLAN in each subnet creates full logical isolation of the Cloud-DCN architecture as shown in figure 4. In order to achieve this, each server and nodes in Cloud-DCN$_s$ is assigned virtualization identity , [V_{id} = av_1, av_2 ……… av_{n-1}] and VLAN identity ($V_l$id) between *1* and *1005,* where av_1, av_2 ……….. av_{n-1} is the virtualization instances on C_DCN$_s$ servers. As such each server can be equivalently identified by a unique V_lid in the range $V_lid \leq 1005^*$.

Hence the total of V_lid for servers in the Cloud-DCN$_s$ is

$$V_l\text{id} = \sum_{id=1}^{id=N} \text{Vlid} * \text{Vs} \qquad (1.1)$$

Where N is the maximum number of VLAN, and Vs is the virtual instances in the C-DCN$_s$.

The mapping between a unique $V_l$id and the C-DCN$_s$ servers considering that there are four C-DCN$_s$ is given in equation (1.2)

$$\text{C-DCN mapping} = 4 * Vl\text{id} * \text{Vs} \qquad (1.2)$$

Following the Cloud-DCN architecture in figure 4, in order to minimize broadcast storms and reduce network traffic/demand density, a VLAN mapping scheme of the servers in the Cloud-DCN$_s$ was applied resulting to the simulation model in figure 5a, 5b

Consider Cloud-DCN$_{sa}$, Cloud -DCN$_{sb,}$ Cloud -DCN$_{sc}$ and Cloud_DCN$_{sd}$ with servers S_1 to S_n. The servers in each of the Cloud-DCN$_s$ are mapped into different VLANs with their corresponding ids as follows:

VLAN$_1$ → S_{1a}, S_{1b}, S_{1c}, S_{1d}....................S_{1n}
VLAN$_2$ → S_{2a}, S_{2b}, S_{2c}, S_{2d}....................S_{2n}
VLAN$_3$ → S_{3a}, S_{3b}, S_{3c}, S_{3d}....................S_{3n}
VLAN$_4$ → S_{4a}, S_{4b}, S_{4c}, S_{4d}................... S_{4n}
VLAN$_n$ → S_{na}, S_{nb}, S_{nc}, S_{nd}................... S_{nn}

Where S_{1a}, S_{2a}, S_{3a}, S_{4a} are the servers in Cloud_DCN$_{sa}$
S_{1b}, S_{2b}, S_{3b}, S_{4b} are the servers in Cloud-DCN$_{sb}$
S_{1c}, S_{2c}, S_{3c}, S_{4c} are the servers in Cloud-DCN$_{sc}$
S_{1d}, S_{2d}, S_{3d}, S_{4d} are the servers in Cloud-DCN$_{sd.}$

With this VLAN mapping scheme, a logical isolation of the Cloud-DCN architecture was achieved as shown in the mode of figure 4. This make for fluid flexibility, improved network security, agility and control of traffic flow in the Cloud-DCN architecture.

3.3. Modeling Traffic Stability for Cloud-DCN

Request or demand arrives randomly in the Multilabel switch, 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 (C_DCN), capacity management and optimum utilization will address broadcast oscillation (congestion) and instability. To address this situation, adapting Little's law 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:

$$D = 1/(\mu p - \lambda p) \qquad (1.3)$$

And the traffic demand, *a* (referred to as offered load or offered traffic in C_DCNs), is given by *a* = λp *

$$\text{Mp} \qquad (1.4)$$

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

3.4. Cloud-DCN Fault Tolerant Algorithm (GLB)

The Traffic algorithm in Cloud-DCN architecture is modelled for effective fault tolerance and failure suppression which makes for greater efficiency in web application integration. The Cloud-DCN algorithm is shown in Algorithm 2.

The procedure in Algorithm 2 normalizes and stabilizes traffic flow in the proposed DCN. In initialization, the rate controller and Ethernet interfaces are initialized while enabling the bus arbitration in ports. In the ports the peripheral component interconnect extension and the MAC are defined while calling the subroutine for more addition of ports in the DCN switch. For each subnet, traffic scheduling is asserted true while enabling the maximum bandwidth for the medium of traffic propagation. On the switch, MAC address mapping is assigned multiplexer switch arbitration bus which suppresses collision types (unicast, broadcast, multicast). For Round Trip Time (RTT), unicast data flows with their frame sizes and packet length are scheduled for two-way handshake (transfer). At the instance of correspondence between a scheduled destination address and rate controller buffer, data, and packet length from the port are established for transfers. The process is repeated throughout the entire period of the DCN traffic initiation. At each point, normalisation of the rate controller, the data length, and the buffer sizes is carried out while consistently suppressing collision forms in the DCN.

Conventionally, in DCN flooding of packets from an active port to destination addresses is done with a compromise to the DCN resources. With GLB beside collision suppression, fair scheduling and sharing of resources is an optimal feature that will enhance service availability and reliable throughput. Hence, with GLB as an improvement to CSMA/CD, utilization of resources by heavy web application servers will maintain dynamic stability without compromise to other QoS metrics.

Algorithm 2: Cloud_DCN Traffic Algorithm.

```
Procedure: trafficController: Public {SERVER 1:N}
  {
    Set
      Normalization UiXi ==TAMP==0
      RateController ==0
      ServerEthernet = = Ethernet Initialization
    Define Abitration Bus {Ports}
                Define PCIx MAC (TF)
                Addports(Cloud-DCNports);
              Data Packets: ➔ MAC Address []
              Cloud-DCN.Subnet(1),transferScheduled(True) { }
              int uploadData() const { return MAC Address};
              setUploadLimit(int bytes Per Second)
              {upLimit = 10 Gbps; }
            Map MacAddres: ➔ Multiplexer Switch Abitration Bus
            SetDownloadLimit(int bytesPerSecond);
            Data ←: Public Unicast data:
                Assign Data bytes: ➔ Length (L);
                    ScheduleTransfer();
  };
  If Cloud_DCN BUFFER && RateController == Destination Address)
  {
            Connect(L,Data (readyToTransfer());
            Complete Scheduled Transfert->setReadBufferSize();
          Output.Network buffer(port);
          ScheduleTransfer();
  }
    For (i =0;i++)
    Clou-DCN BUFFER==RateController ➔ UiXi
  {
  Normalize(PortContention,Collision,Saturation&&SwitchPoison)
    SIGNAL(readyToTransfer()),
      C-DCN->setReadBufferSize(0);
  }
```

4. Simulation Analysis

4.1. Design Context

As for the simulation testbed used for Cloud DCN simulation, a generic template for running C-DCN was developed using OPNET IT guru as a simulation tool. The equivalent system model is shown in figure 2. The C-DCN architecture of figure 5a is made up of following major component vis:

- The *N* number of C-DCN subnets with their Media Access Control (MAC) controller and their application data blocks.
- The MLS switch model comprising of the First-in-First-out (FIFO) queue, connecting a server farm gateway with a Gigabit Etherent link.
- The http IP Cloud.
- Entity sources ie end users.

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. All links and nodes passed the test in all cases. Figure 5b shows OPNET screenshot for simulation sequence used in the analysis. In all the simulations, we enabled the essential attributes for each of the two scenarios on the template. Simulation completed successfully and the results collated in the global and object statistics reports. The simulation plot of the Cloud DCN model under study is shown in figure 6.

4.2. Simulation Results/Hypothesis Validation

4.2.1. C-DCN Model Validations

For further validation of our C-DCN, we used the design parameters obtained from our experimental testbed to develop a generic template in OPNET modeller (a simulation tool). Based on some experimental measurement carried out on the testbed, we used the throughput metric for performance evaluations of fault tolerance throughput index. On the generic OPNET template shown in figure 5a, two scenarios were created, one for Cloud DCN Fault tolerance and one for Cloud DCN No Fault tolerance. 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 in this work as shown in Appendix 1. The C-DCN used only a low-end MLS with the gateway load balancing functions. It also uses traffic routing algorithms viz: VLAN, feedback mechanisms, server virtualization, and convergence with randomization. The experiment only used the throughput parameter for fault-tolerance analysis against the two scenarios. The load balancer can detect when a server fails and remove that machine from the server pool until it recovers. In this scenario, server3 fails 5 minutes into the simulation. 30 minutes later, the server recovers. Appendix 1 shows the OPNET Guru Simulation Data set generated from figure 5b after building the trace file statistics in figure 5a. Figure 5c shows the packet animation flow of traffic from users as depicted in figure 1 previously discussed.

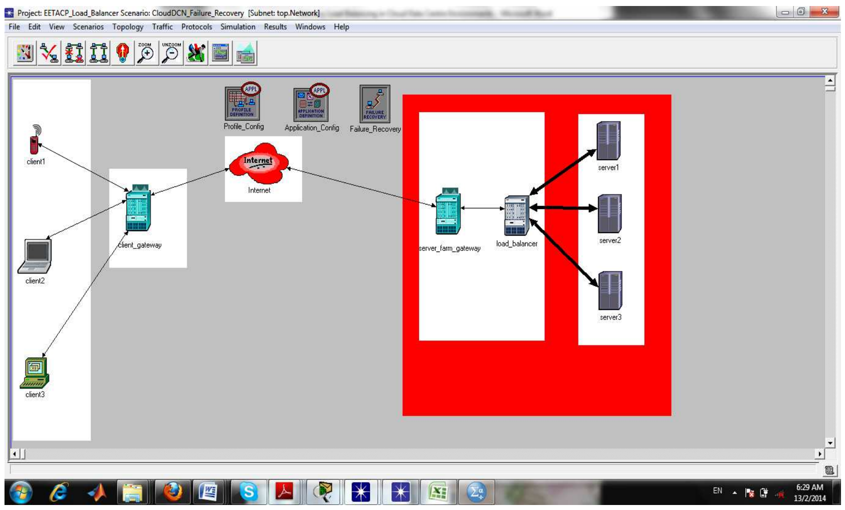
Project: EETACP_Load_Balancer Scenario: CloudDCN_Failure_Recovery [Subnet: top.Network]
File Edit View Scenarios Topology Traffic Protocols Simulation Results Windows Help
Profile_Config
Application_Config
Failure_Recovery
client1
client2
client3
client_gateway
Internet
server_farm_gateway
load_balancer
server1
server2
server3

Figure 5a. *Simulation Testbed for Cloud-DCN validation.*

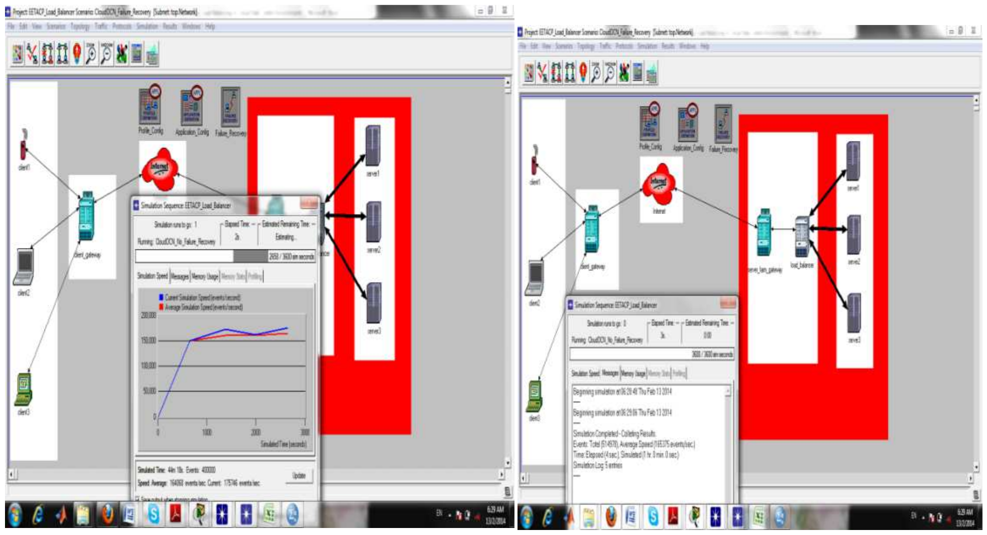
Simulation Sequence: EETACP_Load_Balancer
Simulation Speed
Messages
Memory Usage
Current Simulation Speed (events/second)
Average Simulation Speed (events/second)
Simulated Time (seconds)
Update

Figure 5b. *OPNET Screenshot for Simulation Sequence used in the Analysis.*

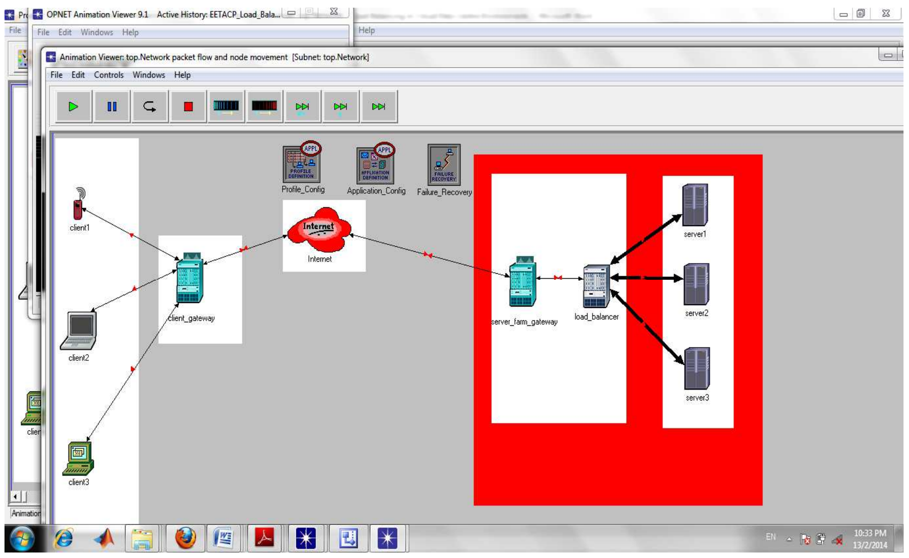
OPNET Animation Viewer 9.1 Active History: EETACP_Load_Bala...
Animation Viewer: top.Network packet flow and node movement [Subnet: top.Network]
File Edit Controls Windows Help
Profile_Config
Application_Config
Failure_Recovery
client1
client2
client3
client_gateway
Internet
server_farm_gateway
load_balancer
server1
server2
server3

Figure 5c. *OPNET Screenshot for the Simulation Packet Flows (Consistence Test).*

4.2.2. Throughput Response Evaluations

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 cloud-DCN. During the simulation, throughput as a global statistics was being measured and compared. Figure 6 shows that the average throughput index as achieved in the simulation. Interestingly, both scenarios, had an initial interesting throughput response which was sustained while Cloud_DCN with fault tolerant maintained a very stable throughput response.

The average throughput in a network with load balancing service has highest throughput compared with the average throughput in a network without a fault tolerant service. The main reason for this is stems from GLB layer 2 introduced in C_DCN design leveraging its advantages as discussed previously.

Again, in all cases, the introduction of a load balancer was expected to balance the traffic at the core, but it was observed that the network model of C_DCN as well as its topological layout had a significant effect on the throughput. Again, this work attributes this observation to the fact that the three-tier topology is communicating on the basis of reduced policy implementation. 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. From figure 6, the Cloud DCN with failure mechanism offered 99% throughput index while without GLB, it yielded 97%. This result validates our research hypothesis stated earlier in this work. We argue that in all ramifications, cloud datacenter environments with no fault tolerance mechanisms will fail in the face of mission critical applications being hosted therein.

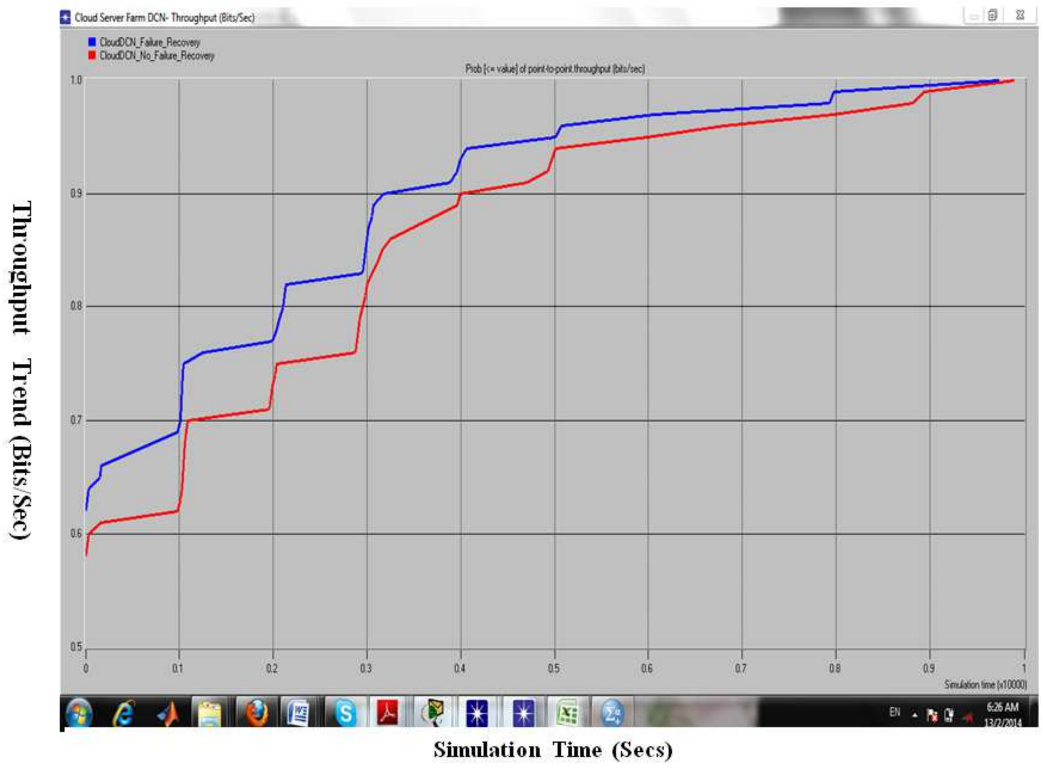

Figure 6. Throughput Index Analysis for C_DCN.

5. Conclusion and Future Works

This paper have presented a throughput index metric in cloud DCN for efficient web application integration. Apart from the literature review carried out, the study, we also developed the system model with Mathematical model for scalability, and logical isolation of Cloud DCN load balancer MLS architecture. The advantages of GLB was outlined. Using the OPNET simulator, we simulated Cloud-DCN and compared the results of the two case fault scenarios. Our discovery showed that C_DCN performed much better (99%) under fault tolerance mechanism compared with the case with no fault tolerance mechanism (97%) thereby validating our stated hypothesis. This work has made significant contributions to the body of knowledge in the following areas:

1. A Cloud DCN model that is very efficient with respect to web application integration, scalable, service-oriented, and responsive to business needs, with rapid service delivery has been realised.

2. An enhanced throughput index comparison between a two case fault tolerance scenario to validate a stated hypothesis
3. Traffic control issues in DCNs have been handled through the analytical model proposed in this work.

Our conclusion therefore, is that the proposed Cloud datacenter 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 we recommend the Cloud DCN to enterprise organisations for greater efficiency in web application integration vis-à-vis their data center network. This will form the basis of our implementation of EETACP as well as the TCP communication protocol into the datacenter in our future works

Acknowledgements

This part of an ongoing PhD work of which the lead author wish to appreciate Prof. O. U. Oparaku, the HOD of Electronic Engineering and my supervisor. Also, I wish to thank the R&D group of Cisco Systems for their expert and meaningful contributions. Also, I wish to thank my co-authors for contributing and sponsoring the publication of this research work.

Appendix 1. OPNET Guru Simulation Data Set Results

Simulation Time	CloudDCN_Failure_Recovery: Network.server_farm_gateway <-> load_balancer [0].point-to-point.throughput (bits/sec) -->.none	CloudDCN_No_Failure_Recovery: Network.server_farm_gateway <-> load_balancer [0].point-to-point.throughput (bits/sec) -->.none
0	0.62	0.58
16	0.63	0.59
32	0.64	0.6
160	0.65	#N/A
167.1111	0.66	0.61
979.5556	0.69	0.62
1011.556	0.7	0.63
1027.556	0.73	0.64
1043.556	0.75	0.66
1059.556	#N/A	0.68
1075.556	#N/A	0.69
1091.556	#N/A	0.7
1251.556	0.76	#N/A
1959.111	#N/A	0.71
1975.111	#N/A	0.72
1991.111	0.77	0.73
2023.111	#N/A	0.74
2039.111	0.78	0.75
2071.111	0.79	#N/A
2103.111	0.8	#N/A
2119.111	0.81	#N/A
2135.111	0.82	#N/A
2874.667	#N/A	0.76
2906.667	#N/A	0.78
2922.667	#N/A	0.79
2954.667	0.83	#N/A
2986.667	0.85	0.81
3002.667	0.86	0.82
3018.667	0.87	#N/A
3050.667	0.88	0.83
3066.667	0.89	#N/A
3114.667	#N/A	0.84
3162.667	#N/A	0.85
3178.667	0.9	#N/A
3242.667	#N/A	0.86
3886.222	0.91	#N/A
3966.222	0.92	0.89
3998.222	0.93	0.9
4062.222	0.94	#N/A
4705.778	#N/A	0.91
4929.778	#N/A	0.92
5009.778	0.95	0.94
5073.778	0.96	#N/A
5989.333	#N/A	0.95
6069.333	0.97	#N/A
6808.889	#N/A	0.96
7932.444	0.98	#N/A
7980.444	0.99	0.97
8816	#N/A	0.98
8944	#N/A	0.99
9747.556	1	#N/A
9747.556	#N/A	#N/A
9904	#N/A	1
9904	#N/A	#N/A

Appendix 2. SLB Configuration for Cloud DCN

Two separate elements need to be configured with SLB, the Server Farm, and the Virtual Server. To configure the Server Farm:

```
MLSwitch(config)# ip slb serverfarm Cloud_FARM
MLSwitch(config-slb-sfarm)# predictor leastconns
MLSwitch(config-slb-sfarm)# real 192.168.1.20
MLSwitch(config-slb-real)# weight 150
MLSwitch(config-slb-real)# inservice
MLSwitch(config-slb-sfarm)# real 192.168.1.21
MLSwitch(config-slb-real)# weight 100
MLSwitch(config-slb-real)# inservice
MLSwitch(config-slb-sfarm)# real 192.168.1.22
MLSwitch(config-slb-real)# weight 75
MLSwitch(config-slb-real)# inservice
```

The *ip slb server farm* command sets the server farm name, and enters SLB Server Farm configuration mode. The *predictor* command sets the load balancing method.

The *real* command identifies the IP address of a physical server in the farm, and enters SLB Real Server configuration mode. The *weight* command assigns the load-balancing weight for that server. The *inservice* command activates the real server

Configuration For Virtual Server

```
MLSwitch(config)# ip slb vserver VSERVERNAME
MLSwitch(config-slb-vserver)# serverfarm Cloud_FARM
MLSwitch(config-slb-vserver)# virtual 192.168.1.10
MLSwitch(config-slb-vserver)# client 192.168.0.0 0.0.255.255
MLSwitch(config-slb-vserver)# inservice
```

The *ip slb vserver* command sets the Virtual Server name, and enters SLB Virtual Server configuration mode. The *serverfarm* command associates the server farm to this Virtual Server.

The *virtual* command assigns the virtual IP address for the server farm.

The *client* command specifies which clients can access the server farm. It utilizes a wildcard mask like an access-list. In the above, *client 192.168.0.0 0.0.255.255* would allow all clients in the 192.168.x.x Class B network. The *in service* activates the Virtual Server.

References

[1] Crowe Horwath LLP , Warren Chan , Eugene Leung, Heidi Pili, "Enterprise Risk management For Cloud Computing ", The Committee of Sponsoring Organizations of the Treadway Commission (COSO), July, 2012.

[2] Jinjing and Raj Jain, "Analysis of Backward Congestion Notification (BCN) for Etherent In Datacenter Applications," In Proc. IEEE INFOCOM, 2007.

[3] Rajkumar Buyya, Anton Beloglazov, and Jemal Abawajy, " Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges",

[4] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, Songwu Lu, "DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers", *SIGCOMM'08,* August 17–22, 2008, Seattle, Washington, USA.

[5] Haitao Wu, Guohan Lu, Dan Li, Chuanxiong Guo, Yongguang Zhang, "MDCube: A High Performance Network Structure for Modular Data Center Interconnection", *CoNEXT'09,* December 1–4, 2009, Rome, Italy.

[6] Feng Huang, Xicheng Lu, Dongsheng Li and Yiming Zhang, " A Fault-tolerant Network Architecture for Modular Datacenter", International Journal of Software Engineering and Its Applications Vol. 6, No. 2, April, 2012, Pp 93-106 .

[7] Dan Li, Chuanxiong Guo, Haitao Wu, Kun Tan,Yongguang Zhang, Songwu L, "FiConn: Using Backup Port for Server Interconnection in Data Centers".

[8] Yong Liao, Jiangtao Yin, Dong Yin, Lixin Gao, "DPillar: Dual-port server interconnection network for large scale data centers, Elsevier Computer Networks 56 (2012), Pp.2132–2147, 2012 ,doi:10.1016/j.comnet.2012.02.016

[9] M. Armbrust et. al. *Above the Clouds: A Berkeley View of Cloud Computing*. Technical Report No. UCB/EECS-2009-28, University of California at Berkley, USA, Feb. 10, 2009.

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

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

[12] Todd lammle, " Cisco Certified Associate Network Guide", Sixth Edition, Wiley Publishing, Inc., Indianapolis, Indiana, 2007.

[13] Udeze C.C, Okafor Kennedy.C., Ugwoke F. N, U.M.Ijeoma, "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.*

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