FAULT TOLERANCE CAPABILITY OF CLOUD DATA CENTER

Humphrey Emesowum
School of Computing, University of Portsmouth
Buckingham Building Lion Terrace PO1 3HE
Portsmouth, United Kingdom
humphrey.emesowum@port.ac.uk

Athanasios Paraskelidis
School of Computing, University of Portsmouth
Buckingham Building Lion Terrace PO1 3HE
Portsmouth, United Kingdom
athanasios.paraskelidis@port.ac.uk

Mo Adda
School of Computing, University of Portsmouth
Buckingham Building Lion Terrace PO1 3HE
Portsmouth, United Kingdom
mo.adda@port.ac.uk

ABSTRACT

In this era of big data and internet of things, the need for performance improvement in cloud data center is unavoidable. This has led to several designs of data center network topologies with the aim of achieving a data center that has the capability of tolerating fault during multiple failures. In this paper, we proposed improved variants of fat-tree interconnections to mitigate the challenges of fault tolerance. The availability of alternative paths for congestion control and fault tolerance gave Fat-tree an edge over other data center architectures, thereby becoming a widely used architecture for data center.

Our focus is on client to server communications in a cloud data center network as explained in Fig. 7, hence simulation of HTTP application was carried out on different variants of fat tree designs. The simulation results with Riverbed showed that our proposed hybrid designs outperformed the Single fat tree designs as the number of link failures increase.

Keywords: Fault Tolerance; Reversed Hybrid; Cloud Data Center; Congestion Control.

I. INTRODUCTION

As the use of cloud data center network increases due to the rapid growth of internet-based applications, the emergence of Internet of Things, and Big Data transfer and analytics; the size and deployment of interconnection networks are expected to increase. Currently, network designs are built to accommodate larger applications, congestion control, increase in throughput, low latency, reliability, and of course fault tolerance - which is the cornerstone. Therefore, to achieve a good level of performance and reliable communication flow in a network, there must be a provision that will tolerate failures of devices and links [1, 2]. In view of this, to achieve a manageable level of fault tolerance in interconnection network, creation of alternative paths between source and destination is imperative for a graceful performance degradation during multiple failures.

The fat-tree topology originates from the fixed topology used in designing data center networks, which is a subset of the tree-based topology [3]. The tree-based topologies, which comprises Basic tree, Fat-tree and Clos network, and other variants are commonly used in the design of data center networks [4, 5, 6]. Fat-tree has undergone different developmental stages since its inception, because of its unavoidable contribution in the design of data center network. According to [7, 8], the conventional fat-tree has a switch-port speed that is unmanageably high towards the root of the topology, a single point of failure, and lacks scalability. The authors of [9] established that at later stage of the developmental stage, generalized fat-tree (GFT) evolved, which is made up of switches of the same radix and same speed port in all the network levels, but with limited paths from source to destination. To achieve flexible performance requirements, the extended generalized fat tree (XGFT) was introduced by Ohring et al; for more routing capacity, performance requirement, and allowing variable number of switch ports to be used at different levels of the network [7, 9, 10]. Nevertheless, the Z-Fat-tree, which is the bedrock of our work, is also a variant of Fat tree introduced by [11]. This improved version of fat tree helps define the number of root nodes per zone or subtree, and adds a degree of connectivity for maximal fault-tolerance. This helps to extract the full benefit of Fat-tree, for design of cloud data center networks that meets its increasing demand; with improved reliability, fault tolerance and alleviate other technical challenges [12,13]. In our previous work [14], we simulated FTP and EMAIL applications using our proposed hybrid and reversed hybrid designs. But, in this paper, an extension of our previous work, we simulated HTTP applications to ascertain our claim that our proposed designs are better than the single fat-trees even with the same number of resources. And the summary of the results (Table 2) proved us right.

Subsequently, in section II, we reviewed concisely some contributions to fault tolerance capability on data center. The section III gives details of the model description of architectures used - the mathematical equations for the switch connectivity and port mapping respectively. Then in section IV, the results of HTTP application simulated on these

topologies at different number of failed links using Riverbed tool, are compared to one another. Finally, section V is where conclusion is drawn as per the level of fault tolerance capability of each data center design.

1. REVIEW OF RELATED WORKS

Generally, across several data center network topologies, there have been some contributions to work around the challenges of traffic congestion and faults in a network; though some of these contributions have their pros and cons. The use of separation technique to improve the performance of data center network was proposed by the authors of [15]. Their work was to make sure that there is no coexistence of different traffic on same transmission path. So that big data traffic will be transmitted via a path different from that used to transmit the ordinary data traffic, thereby making the source to destination transmission paths congestion free. We agreed on this concept but also argue that the authors failed to put into recognition the bedrock of effective and reliable network performance, which is fault tolerance. Consequently, our proposed hybrids (H_2^+) and reversed hybrid (H_2^-) design proved that even having same number of servers with the single fat trees ($\boldsymbol{Z}$), are still better used for data center.

The authors in [16] proposed mechanism that enables a reliable data delivery as links failure occurs in data center. This mechanism also evenly distributes traffic and recovers failure thereby causing a reduction in the operational cost of data centers. This architecture is made up of a.) path-level failure detection for detecting and recovering of failures; b.) a precomputed multipath routing ensuring continuous transmission during failure; c.) and local adaptation to path failure that enables traffic rebalancing from unhealthy path to healthy path. However, the inability of this architecture to proactively sense a faulty device (because its path failure detection is a reactive mechanism), is one of its shortcomings. As a result, there is a time lag for the traffic rebalancing from unhealthy paths to the healthy ones; which will reduce performance in data center. On the contrary, our hybrid designs can sustain the communications proactively without a waste of time failure detection and rebalancing of traffic.

The authors in [17] discussed the advancement in data center using packet-based optical interconnection networks for meeting network traffic requirements, reduction in power consumption, and all-to-all connectivity. According to them, packet-based optical switching is comparable to the common networking structures use in building nowadays data centers. Using the optical packet-based switching, encourages an array of fixed tunable transmission to address specific port destination by choosing the appropriate wavelength. An example of this kind of optical interconnection network is the DOS architecture proposed by X. Yin et al., [18]. This is based on arrayed waveguide grating router that allows different input to get to same output at the same time via different wavelengths; resulting to low latency even at high input loads. However, as we already mentioned in our previous paper [19] optical interconnection used in data center only provides it with high capacity, low power, and low latency; while the challenges of fault tolerance, network scaling, and reduced cost of deployment are still issues confronting this architecture [17]. But with our proposed improved version of Fat-tree hybrids, fault tolerance, network scalability, and cost-effectiveness are guaranteed.

In another instance, the authors in [2] worked on achieving fault tolerance in a data center by hosting a virtual data center on a physical data enter. The virtualization was basically on server failures, whereby virtual machines were relocated from failed host servers to the healthy servers. By doing this, they could recover failure and reduce the effect of server failures on the virtual data center with the help of their proposed load balancing scheme. Nevertheless, their work is targeted only at server failure recovery. Secondly, the process of relocating the failed virtual servers to the healthy ones is time consuming; therefore, defeats its original plan to improve fault tolerance in data center. Nevertheless, our improved fat tree designs (which is typically based on the commonest failures in data center – links failures) sustain fault tolerance capability in real-time to achieve graceful performance degradation.

Finally, according to [20], the authors of "On Performance Evaluation of Fault-Tolerant Multistage Interconnection Networks"; after a wide survey on fault tolerance properties, established that current approaches put in place in multistage interconnection networks could not consider dynamic fault tolerance. They argued that continuous addition of hardware to multistage interconnections networks for its enhancement will not produce a better performance when compared to the original network. On this note, we believe that our proposed designs (especially the reversed hybrid H_2^-), which shows that fault tolerance and congestion control can not only be realized by adding extra hardware, is a good contribution to the cloud data center because of the increasingly growth of internet applications. Then with our other proposed hybrid (H_2^+), based on the throughput, when compared to the single fat-trees shows that the trade-off for using more switches in the design is worth it for better fault tolerance and general performance enhancement.

2. MODEL DESCRIPTION

Generally, fat trees can be defined in several ways. In [20] with the notation $FT(h;m_1,m_2..,m_h;w_1,w_2..,w_h)$, it is succintly defined thus: h represents switch levels of the tree numbered from 0 at the bottom. The sequence m_l, m_2 represent the number of children each switch at level1 and level2 has respectively; while w_l, w_2 represent the number of parent-switches of a host and a switch at level$l-1$ and level1 has respectively.

As already stated in our previous works [14, 19], in constructing the variants of improved version of fat tree we used in comparing the fault tolerance capability of cloud data center, it is appropriate to look at how we derived the switch level relationship, switch connectivity and port mapping through mathematical equations. By default we use full connectivity to connect the servers at level*0* to level1 switches for each zone/subtree, and the numbering of switches and its ports at every level are from left to right starting from zero. Where there is no extra links used, the switch to switch connection is done by connecting each lower level switch to the quotient gotten from the divisor (the greatest common

divisor (gcd) of R_{n+1} and R_n,) and the dividend (R_{n+1}); i.e. $R_{n+1}/gcd_{(Rn+1, Rn)}$.

Furthermore, where extra links are used in the connection, we introduced the pattern used by the authors of Z-Fat tree *[11]*. The Z-Fat tree describes the number of root nodes per zone in its semantics and adds a degree of connectivity, with the notation: Ƶ (h; z_1, z_2, …, z_h; r_1, r_2, …,r_h; g_1, g_2, …,g_h). Where h refers to the number of levels, z_n represents the number of zones/subtree at level n, r_n is the number of root nodes within each of the zones z_{n+1}, and g_n specifies the degree of explicit connectivity at level n (Fig. 1a).

Therefore, for our single topologies: Fig.1 Ƶ(2;4,6;4,8,1,1), the sequence r_1 =4 and r_2 =8 refers to the number of root nodes inside each of the zones z_2 and z_3 respectively. The sequence g_1=1 and g_2=1, indicates there are no extra connections. For Fig. 1: Ƶ (2;4,6;4,8;1,2), we have the sequence r_1 =4 and r_2 =8 refers to the number of root nodes inside each of the zones z_2 and z_3 respectively. The sequence g_1=1 and g_2=2, indicates there are extra two links at level 2. And Fig. 3, Ƶ (2;4,6;4,8;1,4) shows the sequence g_1=1 and g_2=4, indicates there are extra 6 links at level 2.

For the Hybrid FT (H_2^+) designs, the same semantics used in Z-fat tree is applicable. For example for Fig. 4 H_2^+(2;6,4;2,8;1,1), the sequence r_1 =2 and r_2 =8 refers to the number of root nodes inside each of the zones z_2 and z_3 respectively. But at levels 1 and 2, r_1 and r_2 are doubled because it is a hybrid. The sequence g_1=1 and g_2=1, indicates there are no extra connections. The same process is applicable to Fig. 5 H_2^+(2;4,6;4,8;1,1) The sequence r_1=4 and r_2=8, refers to the roots of the zones z_2 and z_3 respectively; the sequence g_1=g_2=1 is the explicit degree of connectivity with no extra links.

For the Reversed Hybrid FT (H_2^-), the same Z-fat-tree notation still holds, but the topology is divided into two parts. With the left-hand-side an exact replica of the right-hand-side in a reversed form (from level1 to level2). So that Fig. 6, H_2^- (2;6,4;2,8;1,1) shows that the sequence r_1 =2 and r_2 =8 refers to the number of root nodes inside each of the zones z_2 and z_3 respectively. The sequence g_1=1 and g_2=1, indicates there are extra connections. These sequences stand for each side of the topology in reversed form, thus it is called a reversed hybrid.

A. Designs of Single FTs (Ƶ)

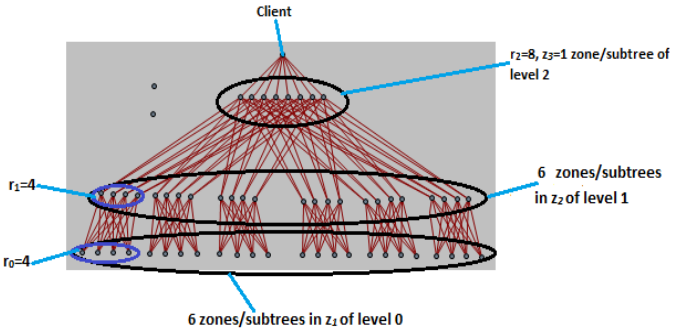

Fig. 1a: Labelling of notations. The Fig. 1a points out the positions of the notations used in describing the topologies. This example is for Single FT Ƶ (2;4,6;4,8,1,1), however, the same naming pattern/labeling is applicable to all other designs.

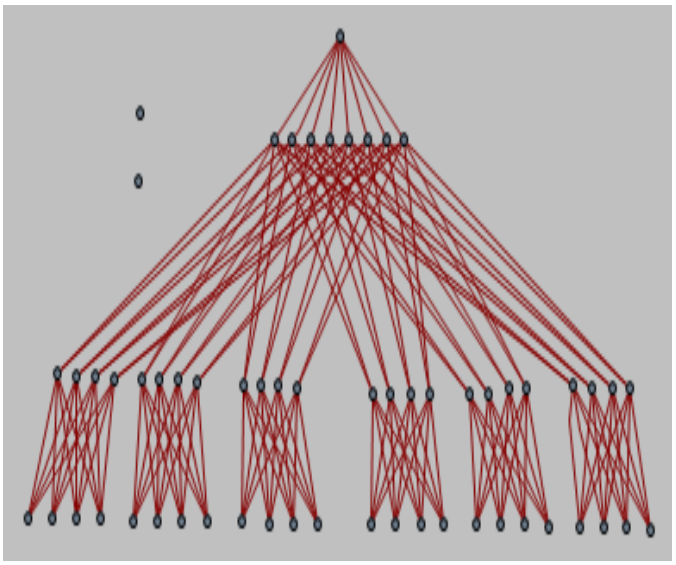

Fig. 2b: Ƶ (2;4,6;4,8,1,1): The sequence r_1=4 and r_2=8, refers to the roots of the zones z_2=6 and z_3=1; the sequence g_1=g_2=1 is the explicit degree of connectivity

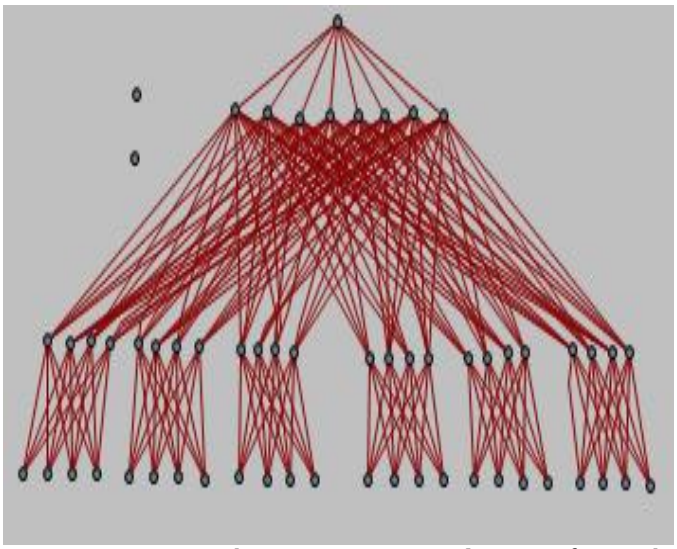

Fig. 3: Ƶ (2;4,6;4,8;1,2) *The sequence r_1=4 and r_2=8, refers to the roots of the zones z_2=6 and z_3=1; the sequence g_1=1 and g_2=2 is the explicit degree of connectivity*

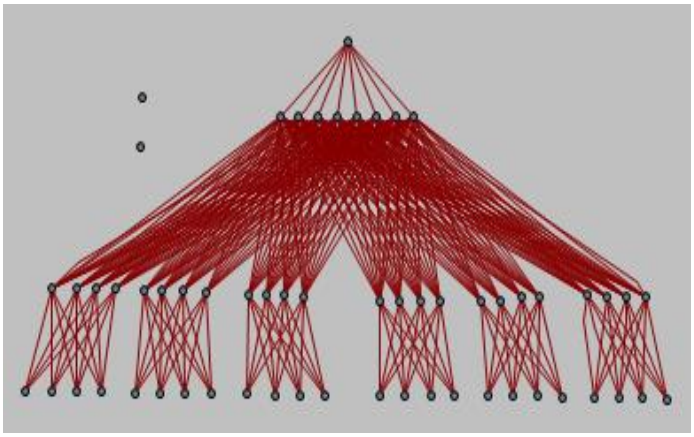

Fig. 4: Z (2;4,6;4,8;1,4): The sequence r_1=4 and r_2=8, refers to the roots of the zones z_2=6 and z_3=1; the sequence g_1=1 and g_2=4 is the explicit degree of connectivity

B. Designs of Hybrid FTs (H_2^+)

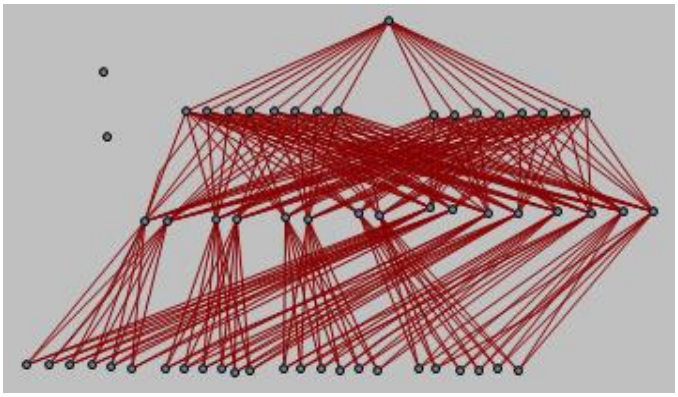

Fig. 5: H_2^+(2;6,4;2,8;1,1): The sequence r_1=2 and r_2=8, refers to the roots of the zones z_2=4 and z_3=1; the sequence g_1= and g_2=1 is the explicit degree of connectivity.

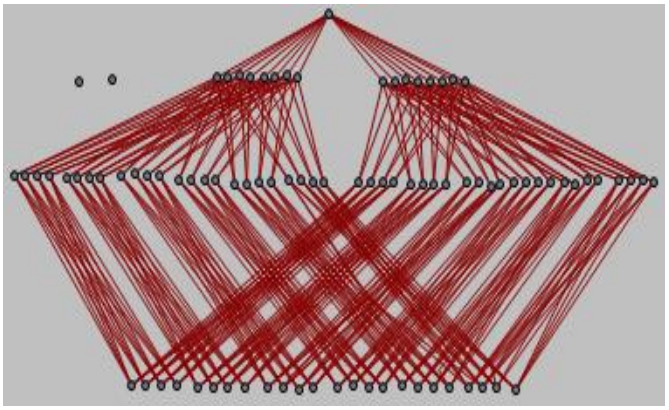

Fig. 6: H_2^+(2;4,6;4,8;1,1) The sequence r_1=4 and r_2=8, refers to the roots of the zones z_2=6 and z_3=1; the sequence g_1=g_2=1 is the explicit degree of connectivity.

C. Design of Reversed Hybrid FT (H_2^-)

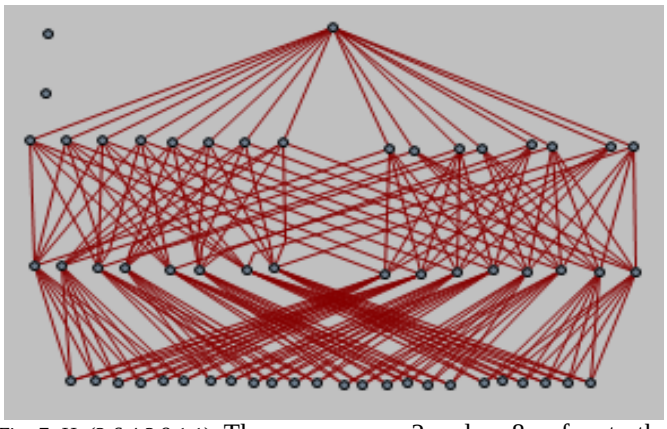

Fig. 7: H_2^-(2;6,4;2,8;1,1): The sequence r_1=2 and r_2=8, refers to the roots of the zones z_2=4 and z_3=1; the sequence g_1=g_2=1 is the explicit degree of connectivity.

D. Switch levels relationship

$$R_{n+1} = R_1 + \Delta\,(n-1) \qquad \textbf{(1)}\ [14]$$

R_{n+1} represents the sought after number of switches at the upper level of the network. R_1 represents the number of switches at the first level of the topology, which must be equal to or greater than 2 to avoid single point of failure. Δ represents common difference between any two levels. This must be constant across the topology. n represents switch level. This depends on the height of the topology, but for simplicity, in this paper we use 2-level FT topology.

E. Switch Connectivity

$$X_{n+1} = (R_{n+1}((x_n \backslash R_n)\backslash S_{n+1}) + (x_n \%R_n)* R_{n+1}/\gcd_{(Rn,Rn+1)} + k)\% R_{n+1} \qquad \textbf{(2)}\ [11]$$

where $\boldsymbol{k}$ *represents* ϵ *{0, 1, …,* $R_{n+1}/gcd(R_n, R_{n+1})$*-1}*

$\boldsymbol{X_{n+1}}$ represents the switch sought after at the upper level upper level to be connected to from the lower level switches.

R_{n+1} represents the total number of switches at the upper level

x_n represents the switch on level n connecting to upper level switch at $X_{n+1.}$ R_n represents the total number of switches on level n connecting to upper level switches at $R_{n+1.}$ Z_{n+1} represents the number of zones/subtrees from upper level $n_{+1.}$ *gcd* is an acronym for Greatest Common Divisor used to get the exact number of R_{n+1} switches that x_n will connect to.

The following examples showed how the first subtree switches at level1 are connected to their orresponding level2 switches in **Fig. 1**.

Example 1: Connecting level1 switch 0 to level2 switches

$$X_{n+1}= (8((0\backslash 4)\backslash 6) + (0\%4)*2+k)\%8$$

$$X_{n+1} = (0 + 0*2+k)\%8$$

$$= (0+k)\%8.$$

where k ϵ *{0, 1, …, $R_{n+1}/gcd_{(Rn, Rn+1)}$-1}* and k=0,1.

Therefore, switches to be connected to at level2 R_{n+1} are:

If k= 0 (0+ k(0)) %8
=0%8

Also if k= 1 (0+ k(1)) %8
=1%8 = 1

Example 2: Connecting level1 switch 1 to level2 switches

X_{n+1}= (8((1\4)\6) + (1%4)*2+k)%8

X_{n+1} = (0 + 1*2+k)%8

= (2+k)%8.

where k ϵ *{0, 1, …, Rn+1/gcd(Ri, Rn+1)-1}* and k=0,1.

Therefore, switches to be connected to at level2 R_{n+1} are:

If k=0 (2+ k(0)) %8
= 2%8 = 2

Also if k= 1 (2+ k(1)) %8
=3%8 = 3

Example 3: Connecting level1 switch2 to level2 switches

X_{n+1}= (8((2\4)\6) + (2%4)*2+k)%8

X_{n+1} = (0 + 2*2+k)%8

= (4+k)%8.

where k ϵ *{0, 1, …, Rn+1/gcd(Ri, Rn+1)-1}* and k=0,1.

Therefore, switches to be connected to at level2 R_{n+1} are:

If k=0 (4+ k(0)) %8
= 4%8 = 4

Also if k= 1 (4+ k(1)) %8
=5%8 = 5

Example 4: Connecting level1 switch 3 to level2 switches

X_{n+1}= (8((3\4)\6) + (3%4)*2+k)%8

X_{n+1} = (0 + 3*2+k)%8

= (6+k)%8.

where k ϵ *{0, 1, …, Rn+1/gcd(Ri, Rn+1)-1}* and k= 0,1.

Therefore, switches to be connected to at level2 R_{n+1} are:

If k=0 (6+ k(0)) %8
= 6%8 = 6

Also if k= 1 (6+ k(1)) %8
= 7%8 = 7

F. Port Mapping

$$X_{p+1}=((X_n \backslash R_n)\%Z_{n+1})*R_n/gcd_{(R_n,R_{n+1})}+p \quad \textbf{(3)} \text{ [11]}$$

where p represents ϵ {0, 1, …, $R_n/gcd_{(R_n, R_{n+1})}$-1}

X_{p+1} represents switch ports to be mapped at upper level. *p* represents set of R_{n+1} switch ports to be mapped with X_n

In the following examples, we mapped every first switch of each zone/subtree of level1 to its corresponding level2 switch port. Since Z_{n+1} represents number of subtrees from upper level R_{n+1}, and level1 in Fig.1 shows 6 subtrees in total, this means that each level 2 switch has 6 different ports for a total of 6 subtrees to be mapped.

Example 1: Mapping level1 switch0 subtree0 to level2 switches is thus:

Xp+1 = ((0\4) %6) * 4/4+ p
= (0 %6) * 1+ p
= 0+p

And p ϵ {0, 1, …, $R_n/gcd_{(Rn, Rn+1)}$-1}

X_{p+1} = 0. Hence, level1 switch 0 will be mapped to ports 0 of level2 switches where it is connected to.

Example 2: Mapping level1 switch 4 subtree1 to level2 switches is thus:

Xp+1 = ((4\4) %6) * 4/4+ p
= (1 %6) * 1+ p
= 1+p

And p ϵ {0, 1, …, $R_n/gcd_{(Rn, Rn+1)}$-1}

X_{p+1} = 1+0 = 1. Hence, level1 switch 4 will be mapped to ports 1 of level2 switches where it is connected to.

Example 3: Mapping level1 switch 8 subtree2 to level2 switches is thus:

Xp+1 = ((8\4) %6) * 4/4+ p
= (2 %6) * 1+ p
= 2+p

And p ϵ {0, 1, …, $R_n/gcd_{(Rn, Rn+1)}$-1}

X_{p+1} = 2+0 = 2. Hence, level1 switch 8 will be mapped to subtree2 with ports 2 of level2 switches where it is connected to.

Example 4: mapping level1 switch 12 subtree3 to level2 switches is thus:

Xp+1 = ((12\4) %6) * 4/4+ p
= (3 %6) * 1+ p
= 3+p

And p ϵ {0, 1, …, $R_n/gcd_{(Rn, Rn+1)}$-1}

X_{p+1} = 3+0 = 3. Hence, level1 switch 12 will be mapped to subtree3 with ports 3 of level2 switches where it is connected to.

Example 5: mapping level1 switch 16 subtree4 to level2 switches is thus:

$$X_{p+1} = ((16\backslash 4)\ \%6) * 4/4 + p$$
$$= (4\ \%6) * 1 + p = 4+p$$

And $p \in \{0, 1, \ldots, R_n / \gcd_{(Rn, Rn+1)}-1\}$

$X_{p+1} = 4+0 = 4$. Hence, level1 switch 16 will be mapped to subtree4 with ports 4 of level2 switches where it is connected to.

Example 6: mapping level1 switch 20 subtree5 to level2 switches is thus:

$$X_{p+1} = ((20\backslash 4)\ \%6) * 4/4 + p$$
$$= (5\ \%6) * 1 + p$$
$$= 5+p$$

And $p \in \{0, 1, \ldots, R_n / \gcd_{(Rn, Rn+1)}-1\}$

$X_{p+1} = 5+0 = 5$. Hence, level1 switch 20 will be mapped to subtree5 with ports 5 of level2 switches where it is connected to.

II. SIMULATION RESULTS AND EVALUATION

The Table 1 is a summary of network inventory for the simulation carried out on Riverbed. The following results were collected for HTTP application: Received packet, Percentage of Packet Loss, and Page Response Time. 24 servers were used across all topologies, every other topology has 32 number of switches excluding the $H_2^+(2;4,6;4,8;1,1)$ that has double of the switches i.e. 64. All the networks were simulated using 2 configuration utilities: Application definition and Profile definition. [21] defined the Application definition as where the usage parameters like time, duration and repeatability are specified while the Profile definition is for describing the activity pattern of a user of the application over a period. Every network has been designed using a single workstation. Workstation is where the profile definition is deployed, used to model the behavior of a user, and acts as traffic source. In this paper, the workstation represents the users over the internet retrieving information from the servers(cloud). The simulation for the three distinct FT designs: Z; H_2^+; and H_2^- for HTTP Applications were run at simulation time of 900 seconds with packet size of 500,000 bytes. At constant Frame Inter-arrival times of 4.0 seconds.

Table 1: Summary of Network Inventory used for Simulation on Riverbed

Elements	Single FT $Z(2;4,6;4,8;1,1)$	Single FT $Z(2;4,6;4,8;1,2)$	Single FT $Z(2;4,6;4,8;1,4)$	Hybrid FT $H_2^+(2;6,4;2,8;1,1)$	Hybrid FT $H_2^+(2;4,6;4,8;1,1)$	Reversed Hybrid FT $H_2^-(2;6,4;2,8;1,1)$
Switches	32	32	32	32	64	32
Workstation/clients	1	1	1	1	1	1
Servers	24	24	24	24	24	24
Ethernet Physical links	152	200	296	240	304	192
Configuration Utilities	2	2	2	2	2	2

F. IP Address Translation

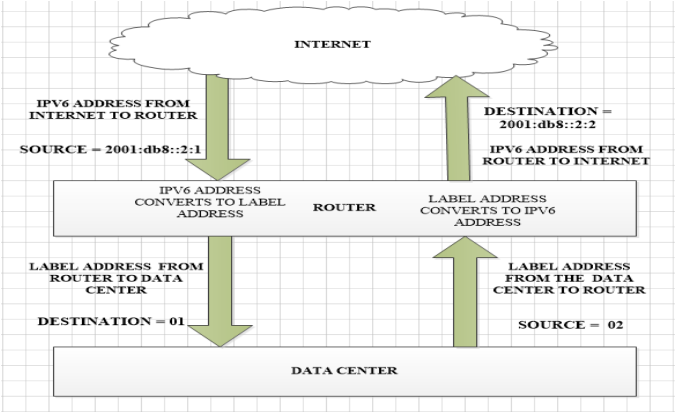

Fig. 7: Mapping Internet IP address to Data Center Labels. This is a network address translation setup that enables the servers of the data center to communicate with the clients across the internet. For detailed explanation, please be referred to our previous work on [14, 19].

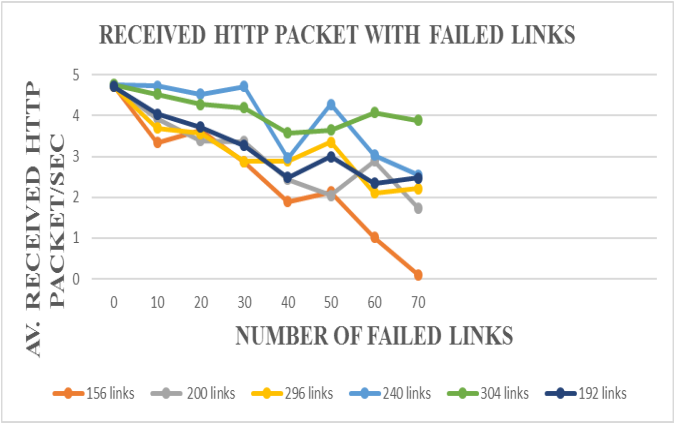

Fig. *8*: GRAPH OF RECEIVED HTTP PACKET WITH FAILED LINKS

The graph, Fig. 8 shows the average number of packets per second forwarded to the HTTP applications by the transport layers in the network. The simulation was run at diffrerent number of failed links on each design. The result shows that with healthy topologies (0 failed link), the average HTTP packet received per second remained almost the same for all designs. But as the number of failed links increase from 0 to 70, there is a noticeable decline in the average received packet/sec:

In general, it could be seen that the hybrid topologies perform better than the single topologies, depicting a better fault tolerance. Furthermore, the single topology with 24 servers, 156 links and 32 switches has an average received packet of 4.71/sec at no failed links. But as the number of links failure increased in multiples of 10 up to 70, the throughput dropped to 0.2 packet/sec. Therefore, comparing this topology with the Hybrid topology that has 24 servers, 304 links and 64 switches. The difference between these two topologies is that the hybrid is twice the single in terms of switches and number of links. But the hybrid has an average received packet per second at no failed link of 4.76 pkt/sec; and when 70 links were failed, the received packet was 3.89

pkt/sec. It is obvious that with such a great margin between 0.2 and 3.89 for the single and the hybrid respectively, the latter can tolerate much fault than the former, which almost has a zero throughput as the failure increased to 70 links. One could argue that since the hybrid topology is twice the single topology in terms of number of switches and links, that the received packet (throughput) should also be twice that of the single topology. This implies that the received packet should have been 0.2*2 =0.4pkt/sec. However, this is not the case here because of the design as the hybrid topology showed a greater fault tolerance capability.

Another design that worthy of comparing with the single designs e.g. Ƶ(2;4,6;4,8;1,2), is the reversed hybrid topology H_2^-(2;6,4;2,8;1,1). The former has the same numbers of switches and servers with the latter, but with the former having 200 links while the latter has 192 links. When there was no failed links, the average received packet per second for the single topology with 200 links Ƶ(2;4,6;4,8;1,2) was 4.71pkt/sec; while that of the reversed hybrid topology with 192 links (H_2^-(2;6,4;2,8;1,1)) was 4.71 pkt/sec. However, the graph of both designs kept declining as the number links failure increases, till the point when 70 links were failed the received packet for single FT design is 1.74 pkt/sec, while 2.47 pkt/sec was for the reversed hybrid. Even the single FT design with a very much higher number of links - 296 links, still showed a lower performance of 2.2 pkt/sec.

Fig. 9 shows graph of the percentage of packet loss with failed links. The graph shows the exact amount of packet that was lost in percentage during the simulation at different stage of failed links for each design. To calculate the packet loss, we use the mathematical relation below

$$\text{PERCENTAGE OF PACKET LOSS} = \left(\frac{\text{SENT PACKETS} - \text{RECEIVED PACKETS}}{\text{SENT PACKETS}}\right) * 100 \quad \textbf{(10)}$$

The graph shows that at 0 number of failed links, all the designs have 75% of packet loss. But as the numbers of failed links increase, the percentage of packet loss also increases. Nonetheless, the hybrid designs showed a better performance with topologies having 240 and 304 links respectively with 77% pkt loss, with a clear significant difference of 10% and above when compared with all the single designs having 87%, 82%, and 83% pkt loss.

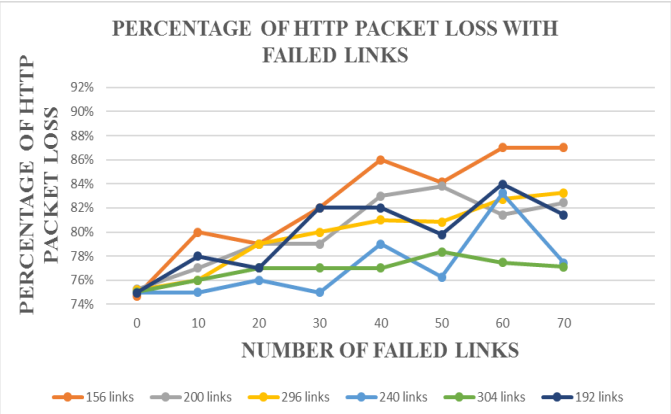

Fig. 9: GRAPH OF PERCENTAGE OF HTTP PACKET LOSS WITH FAILED LINKS

Table 2: Summary of Results for Received Packet and Percentage of Packet Loss for both Healthy and failed links.

Different Fat-tree- based Data Center Designs	Results of Healthy Links		Results of 70 Failed Links	
	Average Rcvd Packet (pkt/sec)	Percentage of pkt Loss (%)	Average Rcvd Packet (pkt/sec)	Percentage of pkt Loss (%)
152 links Ƶ (2;4,6;4,8;1,1)	4.73	75	0.2	87
200 links Ƶ (2;4,6;4,8;1,2)	4.71	75	1.74	82
296 links Ƶ (2;4,6;4,8;1,4)	4.71	75	2.2	83
240 links H_2^+(2;6,4;2,8;1,1)	4.75	75	2.54	77
304 links H_2^+(2;4,6;4,8;1,1)	4.76	75	3.89	77
192 links H_2^-(2;6,4;2,8;1,1)	4.71	75	2.47	81

The Fig. 10 shows the graph of HTTP Page Response Time, which is the time required to retrieve the entire page with all the contained inline objects [21]. It can be seen from the graph that there is a slight increase in the response time as number of failed links get to 70. The reason behind this is the fact that as the failure increases, there is increase in congestion and the time to retrieve the HTTP page will also increase. However, in the case of single FT design with 156links Ƶ(2;4,6;4,8;1,1), the page response time decreased to almost zero because at 70 failed links, the received HTTP packet was just 0.2 pkt/sec. Another thing to note from the graph is that the reversed hybrid topology has a slightly higher page response time than the other topologies. This means that it took a bit longer time for the page of the HTTP to be retrieved for the reversed hybrid.

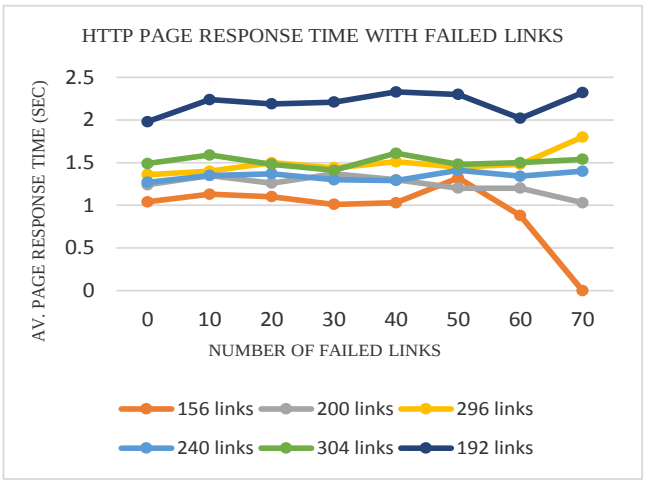

Fig. 10: GRAPH OF PAGE RESPONSE TIME

CONCLUSION

The results obtained from the simulations carried out show tremendous difference in the amount of throughput and packet loss, especially for the hybrid (H_2^+) with 304links, 64

switches; and the Single FT (Ƶ) with 152links, 32 switches. Meanwhile, to ascertain our earlier claim that fault tolerance and better performance can not only be realized by adding extra hardware rather bespoke design plays a greater role. The result of our proposed Reversed Hybrid (H_2^-) with 192 links, and that of the Single FT (Ƶ) with 296 links proved it all. Therefore, based on the number of failed links with respect to the Received Packets and Percentage of Packet Loss, it is certain that our proposed hybrids (H_2^+ and H_2^-) can bring about robust and reliable cloud data centers because of their fault tolerance capabilities. The fault tolerance capability exhibited by our proposed hybrid designs shows that faults in cloud data center could be managed in real time through bespoke design, till repair becomes available. Unlike some of the contributions as reviewed in the related works whereby the transfer of data from failed devices to healthy ones consumes more unnecessary time and causes delay.

REFERENCES

[1] P. Gill, N. Jain, & N. Nagappan, 2011. Understanding network failures in data centers: measurement, analysis, and implications. ACM SIGCOMM Computer Communication …. Available at: http://dl.acm.org/citation.cfm?id=2018477 [Accessed January 6, 2015].

[2] S.C. Joshi, & K.M. Sivalingam, 2013. On fault tolerance in data center network virtualization architectures. 2013 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), pp.1–6. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6802837.

[3] Y. Liu, K.K. Muppala, & M. Veeraraghavan, 2014. A survey of data center network architectures., p.22pp.

[4] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in Proceedings of the ACM SIGCOMM 2008 conference on Data communication. ACM, 2008, pp. 63–74.

[5] M. Niranjan, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," ACM SIGCOMM Computer Communication Review, vol. 39, no. 4, pp. 39–50, 2009.

[6] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in Pro-ceedings of the 7th USENIX conference on Networked systems design and implementation. USENIX Association, 2010, p. 19.

[7] C. Minkenberg, R. P. Luijten, & G. Rodriguez, 2011. On the optimum switch radix in fat tree networks. 2011 IEEE 12th International Conference on High Performance Switching and Routing, HPSR 2011, pp.44–51.

[8] Y. Sun, J. Chen, Q. Liu, & W. Fang, 2014. Diamond: An improved fat-tree architecture for large-scale data centers. Journal of Communications, 9(1), 91–98.

[9] A. Peratikou, & M. Adda, 2014. Optimisation of Extended Generalised Fat Tree Topologies, 1(c), 1–10. Chapter January 2014 DOI: 10.1007/978-3-319-05209-0_7.

[10] E. Zahavi, 2010. Fat-Trees Routing and Node Allocation Providing Non-Blocking MPI Global Collectives for Exascale Jobs. Electrical Engineering, pp.1–8.

[11] M. Adda, and A. Peratikou. Routing and Fault Tolerance in Z-Fat Tree. IEEE Transactions on Parallel and Distributed Systems. Year: 2017, Volume: PP, Issue: 99 Pages: 1 - 1, DOI: 10.1109/TPDS.2017.2666807

[12] M. Bradonji, B. Labs, & M. Hill. Scaling of Capacity and Reliability in Data Center Networks Categories and Subject Descriptors, 2, pp.3–5.

[13.] K. Bilal, S. Khan, L. Zhang, and H Li, 2013. Quantitative comparisons of the state- of- the- art data center architectures. Concurrency and …, (December 2012), pp.1771–1783. Available at: http://onlinelibrary.wiley.com/doi/10.1002/cpe.2963/full [Accessed January 7, 2015].

[14] H. Emesowum, A. Paraskelidis, and M. Adda. "Fault tolerance improvement for fat-tree based cloud data centers", The 7th International Conference on Information Communication and Management, ICICM 2017, Moscow, Russia, in press.

[15] H. W. Park, I. Y. Yeo, J. R. Lee, & H. Jang, (2013). Study on Big Data Center Traffic Management Based on the Separation of Large-Scale Data Stream. 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 591–594. http://doi.org/10.1109/IMIS.2013.104

[16] M. Suchara, F. Park, D. Xu, J. Rexford, 2011. Network Architecture for Joint Failure Recovery and Traffic Engineering Categories and Subject Descriptors. ACM SIGMETRICS, pp.97–108

[17] C. Kachris, K. Konstantinos, I. Tomkos, 2013. Optical Interconnection Networks in Data Centers: Recent Trends and Future Challenges, (September), pp.39–45.

[18] X. Ye et al., "DOS: A Scalable Optical Switch for Data-centers," Proc. 6th ACM/IEEE Symp. Architectures for Networking and Commun. Sys., 2010, pp. 24:1–12.

[19] H. Emesowum, A. Paraskelidis, and M. Adda. "Fault Tolerance and Graceful Performance Degradation on Cloud Data Center", The 10th International Conference on Computer Science and Information Technology, ICCSIT 2017, Florence, Italy, in press.

[20] F.O. Sem-Jacobsen et al., 2011. Dynamic fault tolerance in fat trees. IEEE Transactions on Computers, 60(4), pp.508–525.

[21] Configuring Applications and Profiles: OPNET, Optimum Network Performance. Available at http://aetos.it.teithe.gr/~ziochr/network_lab/configuring_applications.pdf