Achieving a Fault Tolerant and Reliable Cloud Data Center Network

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*— The need for a robust data center that is fault tolerant can never be overemphasized, especially nowadays that the advent of big data traffic, internet of things and other on-demand internet applications are on the increase. The rate at which these data are transferred across the internet is worrisome, and a thing of concern to the data center developers. The emergence of ubiquitous computing has also aided to the increase in traffic across the internet, because computing occurs more now by use of any device, in any location, and in any format. These issues have compounded the management of Cloud Data Center used for storage, transfer, and analysis of data across the cloud; as a result, the data center network devices become prone to failures, which automatically impacts on its performance. Nevertheless, several researchers have come up with solutions, though not sufficient to mitigate these issues. Therefore, on our part, we realised that architectural design of data center network is the bedrock of having a fault tolerant, reliable, robust, and congestion free network. So, this paper, which is an extension of our previous works, based on an improved version of Fat Tree (called Z-node); we proposed a Hybrid fat tree design and compared it with Single fat tree design, for client to server communication pattern such as HTTP and EMAIL applications. The simulation results obtained with different device failures and traffic rate patterns, show that the Hybrid fat tree design performed better than the Single fat tree design, hence will be the best bet for the transfer and analysis of big data in cloud data center network.**

Keywords— Cloud Data Center Network; Fault-tolerance; Reliability; Congestion Control; Fat Tree (FT).

I. INTRODUCTION

Nowadays, the need to improve the fault tolerance of data center network by achieving robustness, availability, reliability, and resilience cannot be overstressed; be it Cloud, End-user-owned, or Scientific data center. Fault tolerance is the basic challenge to overcome for the abovementioned network performance metrics to be sustained in every data center network. Although it is very much critical in the case of cloud due to the emergence of ubiquitous computing, internet of things, and big data [1]. Therefore, in designing a data center network for the primary aim of achieving fault tolerant, the need to use the right choice of topology that will be able to provide sustainable performance amidst failures is inevitable. According to the authors of [2], the many switches, links, and servers used in data center network made it undeniably prone to failure. Over few decades, the challenges observed in data center network prompted design and redesigning of several network architectures such as Fat tree, DCell, BCube, VLE and so on [3, 4]. However, Fat tree, which originated from fixed topology, has been in use by several researchers to alleviate data center network fault tolerance. This is because of its multipath from source to destination [3, 5]. Fat tree has also been undergoing continuous stages of improvement by researchers to actualize the maximum use of its benefits in designing a robust fault tolerant data center network [3, 6, 7]. The authors in [2, 8] pointed out that the issue of scalability of the convectional Fat tree brought about the era of generalized Fat tree, which also has its challenges of having switches of the same radix and speed port across every level of the network. Nevertheless, Extended Generalized Fat tree emerged, allowing variable number of switch ports to be used at different level of network [18, 19, 20]. Having said that, our work evolved from a variant of Fat tree called Z-node that provides extra degree of connectivity to utilize the extra ports per switches that are in some cases not utilized by the architectural constraints of other variants of fat trees by Adda and Peratikou [9]. We have used this variant of Fat tree in some of our previous works [1, 10, to prove that our Fat tree hybrid design is a better option to use than single design when fault tolerance and graceful performance degradation are of optimum importance in cloud data center network. One of the proves was failing 70 links in each topology, then run series of simulations at different interarrival times [11]. Likewise, in this paper, with our Hybrid FT design that is twice the Single FT design; we proved this by simulating EMAIL and HTTP applications based on different traffic and failure conditions.

In the subsequent sections, e.g., II, related literatures are reviewed; then in III, detailed design description of the topologies are provided, these include mathematical relations for the switch connectivity and port mapping. In section IV,

we analyze the results of our simulations and discuss the trade-offs of both designs. Section V is where conclusion is drawn based on how gracefully each design was able to perform during failures. Finally, in section VI, we disclosed what our future work is to improvement and expansion of our proposed Hybrid data center network.

II. REVIEW OF RELATED WORKS

In a bid to solve problems associated with general performance challenges in cloud data center networks, there have been diverse research contributions. However, some of these related contributions are succinctly reviewed here, and they are: the use of traffic separation techniques; virtualization of data center based on server failure; leveraging traffic congestion using Fat tree-based InfiniBand architecture; and using optical interconnection in Helios architecture.

Traffic Separation Technique is a type of performance improvement in cloud data center proposed by the authors in [12]. Their aim is to have dedicated path for the big data traffic thereby separating the ordinary traffics from that of the big data. According to the authors big data traffic severely affects data center network. So, when separated to guarantee no coexistence of different data traffics on same path, it will lessen the overall transmission time of big data. However, this proposal focuses on how to reduce traffic congestion for big data only. Fault tolerance, which is the bedrock of reliability and availability in data center was also neglected of which achieving better performance improvement is questionable. In our work, not only that fault tolerance is achieved, reliability and congestion control are also taken care of.

Data center virtualization based on server failure is another contribution by Joshi and Sivalingam [13] to achieve fault tolerance in a data center network. Their work's strategy is based on the relocation of virtual machines hosted on the failed servers to healthy servers, which according to them, 90% of server utilization was achieved. In addition, they allocated virtual data center across the physical data center network using clustering to balance the network load and reduce the impact of server failure. But, this contribution to alleviate fault tolerance is only for server failures, which also by its process introduces delay to the network while relocating failed servers to the healthy servers. Contrarily, our contribution based on Hybrid design is mainly on the commonest failure regions in data center, which are the failure of communication links and the switches.

To help leverage traffic congestion in data center network, the authors of [14] proposed "A multiple LID Routing Scheme for Fat-Tree-Based InfiniBand Networks". InfiniBand architecture is used for its high bandwidth and low latency. However, when several processing nodes forward packets simultaneously to another node that is associated with one local identifier (LID), there will be traffic congestion. For this reason, these authors proposed that multiple LIDs be associated with one processing node based on the LID Mask Control (LMC) mechanism. So that each of the several processing nodes can make use of a LID each from the multi LIDs associated to the destination node to forward packets at the same time. This proposal is just for traffic congestion control only, again neglecting fault tolerance that is the bedrock of effective network performance. Based on their design, there is only one link from the subnet switch to the processing node of which if there is a switch failure, the processing node and the entire subnet will be disconnected from the network, resulting to a single point of failure.

Another contribution to performance improvement in data center that lacks fault tolerance capability in its design is the work of [15] called Helios architecture. It is a two-layer hybrid circuit-based data center network, with optical interconnection using wavelength division multiplex links; and uses optical circuit and electrical packet switches. The optical circuit switch encourages high bandwidth and long-lived communications, while the electrical packet switches are used for low latency, and fast all-to-all communication. Meanwhile the downside of this Helios architecture is lacking fault tolerance, scalability, and cost effectiveness [16]; thus, giving our Hybrid design a boost because it is scalable and tolerate fault.

III. MODEL DESCRIPTION

In our previous works [1, 10 11], we discussed how Fat tree topology was extensively explained and its notation defined. However, we also stated that the construction of the variant of improved version of fat tree we used in comparing the fault tolerance of cloud data center is derived from the mathematical equations of switch level relationship, switch connectivity, and port mappings. In our two-switch-level topologies, the level2 switches are connected to the clients with full connectivity; likewise, the servers at the base (level*0)* are connected to level1 switches in each zone/subtree using a full connectivity. The numbering of switches, their ports (at level1 and level2) and zone/subtree are from left to right starting from zero. We deploy the pattern used for Z-Fat tree by the authors in [12] in the connectivity; although, in this paper, we decided not to add extra links to the designs as we did in our previous works for simplicity sake. Nevertheless, the Z-Fat tree describes the number of root nodes per zone in its semantics and adds a degree of connectivity as 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 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 (figure1). Therefore, for the Single FT design figure 2: 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 our Hybrid FT design, the same semantics used is applicable. For example in figure 3: H_2^+(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. But at levels 1 and 2, r_1 and r_2 are doubled, thus the reason for hybrid. The sequence g_1=1 and g_2=1, indicates there are no extra connections.

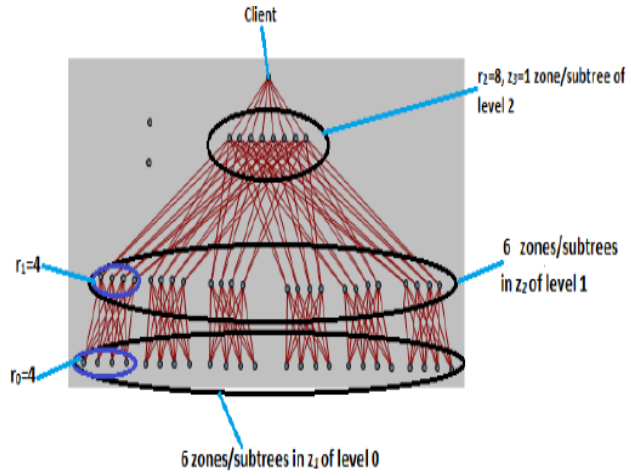

Figure 1: *Sample of Labelling showing positions of the notations used in describing the topologies [1]*

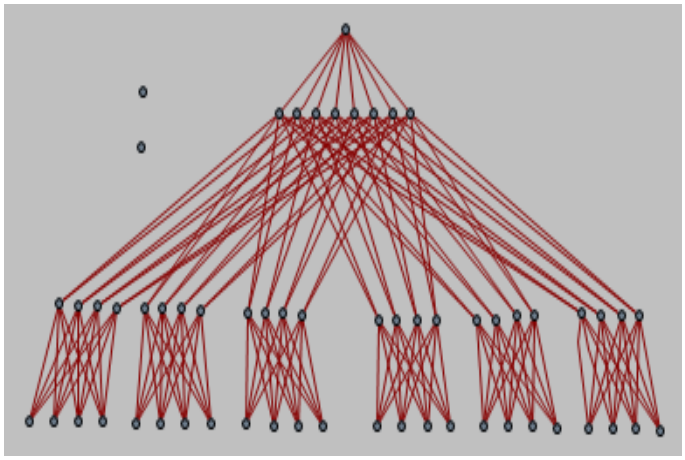

Figure 2: Z (2;4,6;4,8;1,1) The Single FT design topology

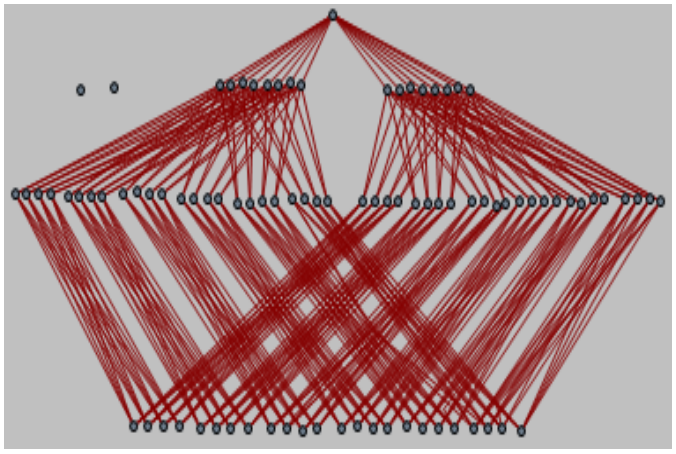

Figure 3: H_2^+(2;4,6;4,8;1,1) The Hybrid FT design topology

A. Mathematical Relations for designing the topologies

In this paper, we will not go into full details of showing the mathematical relations of the topologies but urge readers to see our previous works [1,10, 11] for better understanding.

a. Switch Level Relationship

$R_{n+1} = R_1 + \Delta(n-1)$ ***(I)***

R_{n+1} represents number of switches at the upper level. R_1 represents the number of switches at the first level, equal/greater than 2. Δ represents common difference between any two levels. *n* ***represents*** switch level.

b. Switch Connectivity

$X_{n+1} = (R_{n+1}((X_n \backslash R_n) \backslash Z_{n+1}) + (X_n \% R_n) * R_{n+1}/\gcd_{(Rn,\ Rn+1)} + k) \% R_{n+1}.$

where ***k*** *represents* ϵ *{0, 1, …, $R_{n+1}/gcd_{(Rn\ ,\ Rn+1)}$-1}.* *(II)[9]*

X_{n+1} *is* switch sought after at the upper level. R_{n+1} is the total number of switches at the upper level. X*n is l*evel *n* switch connecting to upper level switch at X_{n+1}. R_n is the total number of switches on level *n* connecting to upper level switches at R_{n+1}. Z_{n+1} is the number of subtrees/zones 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.

Hybrid FT design switch connectivity

In the following steps, the first zone/subtree switches (e.g. 0 and 1) at level1 are connected to their corresponding level2 switches for the left-hand side of the Hybrid FT design (Figure 3). However, the same procedure applies for connecting the right-hand side of design; and it is also applicable to the Single FT design

Step 1. To connect switch 0 at level 1, to its corresponding level2 switches, is as follows:

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

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

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

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

k = 0,1.

Therefore, corresponding switches to be connected to at level2 are:

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

=0 % 8 = 0

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

=1 % 8 = 1

Step 2. To connect switch 1 at level1 to its corresponding level2 switches, is as follows:

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

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

$= (2+k) \% 8.$

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

k= 0,1.

Therefore, corresponding switches to be connected to at level2 are:

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

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

c. Port Mapping

$X_{p+1} = ((X_n \backslash R_n) \% Z_{n+1}) * R_n / gcd_{(Rn, Rn+1)} + p$ (III)[9]

where ***p,*** set of switch ports to be mapped, and represented as*:* $p \in \{0, 1..., Rn/gcd_{(Rn, Rn+1)} - 1;$

X_{p+1} ***r***epresents switch ports to be mapped at upper level.

Port mapping for Hybrid FT design **first zone/subtree switches**

In the following steps, the ports of the first zone/subtree switches (e.g. 0 and 1) at level1 are mapped to their corresponding level2 switches ports for the left-hand side of the Hybrid FT design (Figure. 3**)**. So, at level1, there are 6 zones for z_2, with r_1=4 in each and with one switch having 2 links. It means that each level2 switch has 6 down-ports to be mapped**.**

Step 1. Mapping level1 switch 0 in the first zone of z_2 to the ports of level2 switches is thus:
$X_{p+1} = ((0\backslash 4) \% 6) * 4/4 + p;$
$= (0 \% 6) * 1 + p,$
$= 0+p;$
and p $\in \{0, 1, \ldots, Rn/gcd_{(Rn, Rn+1)} - 1.$
$X_{p+1} = 0.$
Therefore, all the upgoing ports of switch 0, will be mapped to the first ports (port 0) of its corresponding level2 switches.

Step 2. Mapping level1 switch 1 in the first zone of z_2 to ports of level2 switches is thus:
$X_{p+1} = ((1\backslash 4) \% 6) * 4/4 + p;$
$= (0 \% 6) * 1 + p,$
$= 0+p;$
and p $\in \{0, 1, \ldots, Rn/gcd_{(Rn, Rn+1)} - 1.$
$X_{p+1} = 0.$
Therefore, all the upgoing ports of switch 1, will be mapped to the first ports (port 0) of its corresponding level2 switches.

d. IP Address Mapping

A network address mapping setup in figure 4, enables the servers of the data center to communicate with the clients that are across the internet. Because the servers are labelled with logical numbers, while the clients have IPv6 address; therefore, for there to be easy communication between them, there must be address mapping. This setup is made up of: data center (servers, switches, and routers) and the internet. As shown in figure 4, the router maps the incoming IPv6 address of arriving packets with source address **2001:db8::2:1** to one of the servers' label address with destination label **01**. The same thing happens when the server is sending back the request from the client, as the router also maps the logical address of the server e.g. **02** to IPv6 with destination address **2001:db8::2:2**, before it can be sent across the internet [1]

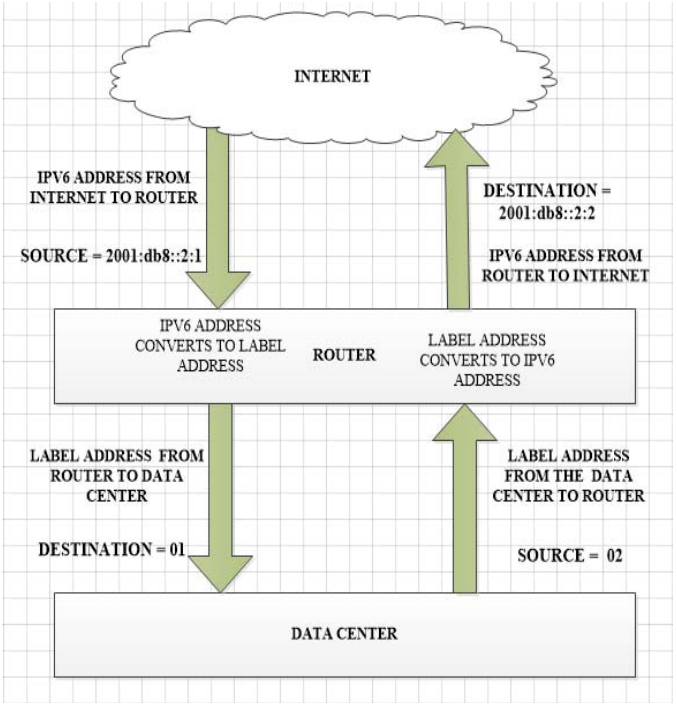

Figure 4: Mapping IP address to labels

IV. Results Analysis

A. Summary of Simulation Inventory

In this section, we will be analysing our simulation results based on two failure categories: a) percentage of failed links only and b) failed links and switches; for both HTTP and EMAIL applications respectively. These simulations were done on Riverbed (is a simulation tool that provides scalable simulation, detailed analysis of network performance and high-precision modelling of both wired and wireless networks) where the results of Received Packets were collected for EMAIL and HTTP applications based on client to server communication pattern. Generally, across every scenario of the Single and Hybrid designs (as shown in tables 1 and 4), 24 servers were used. However, the Single FT design has 32 switches and 152 links, while our proposed Hybrid FT design is twice the size of the Single, i.e. 64 switches and 304 links. The network simulation time for each scenario is 900 seconds, using 2 configuration utilities: Application definition where the usage parameters like time, duration and repeatability are specified; and Profile definition used for describing the activity pattern of a user of the application over a period. A workstation, which we referred to as client is where the profile definition is deployed for modelling the behaviour of a Client and acts as source of traffic. In this paper, the Clients represent the users over the internet retrieving information from the servers (cloud) [11].

From the first scenario (in Table 1) where links were failed based on percentages, we made sure that for the 10% failed links i.e. 15 links for the Single topology were all at level1 switches. Likewise, the 30-failed links of the Hybrid topology at 10% were all failed at its level1 switches. For the 20% failed links, which is 30 and 60 failed links for Single and Hybrid topologies respectively, were failed at level 1 and level2 switches equally. The same pattern was used to fail the other 30% and 40% links, which were carefully and selectively done to avoid failing all links connected to a switch lest the switch become isolated. In the second scenario (table 4), where both links and switches were failed. We applied the same procedure of carefully and selectively failing of links in both topologies. Example, for the Single topology, where 10 links were failed, we failed all 10 links at level1 switches. Similarly, where 20 links of Hybrid were failed, it was done at the level1 switches too. Then 5 and 10 switches of Single and Hybrid topologies respectively where also failed at level1, making sure that no failed link is connected to a failed switch. In all, we avoided random link failure to give a justified result and prevent having different effect. It is worth knowing that by not allowing of random link failure, as discussed above, helped to achieve a uniform pattern of failure across the two topologies for accurate result. Therefore, links were manually failed prior to starting the experiments.

a. Result analysis based on the percentage of failed links

Based on the percentage of failed links, for the Single and Hybrid topologies having 152 and 304 links respectively. Then, 10%, 20%, 30% and 40 % of each total amount of links were failed as shown in Table 1. So, with this failure comparison it is a bit fairer though 30% and 40% of the Hybrid have more link failures (see table 1). Notwithstanding, the results of Received HTTP packets as shown in figure 4 and table 2 confirm that the Hybrid tolerates more faults more than the Single. This is because at 0% failed links (healthy links), the Single design has a received output of 4.71 pkt/sec, while the Hybrid design has a received output of 4.76 pkt/sec. And as the percentage of failed links increases to 40% so that the number of failed links for Single design is 60, and that of the Hybrid is 121; they both produced outputs of 1.98 pkt/sec and 2.98 pkt/sec respectively. This shows that the Hybrid is more capable of tolerating faults, hence the reason for higher throughput.

In like manner, with EMAIL application as shown in figure 6 and table 3; at 0% failed links, both designs produced same output of 34.43 pkt/sec. However, as the number of failed links increase on percentage basis, one could see tremendous difference between the throughputs of both designs especially at 40% failed links where the Single topology has Received packets of 11.46 pkt/sec and the Hybrid has Received packets of 22.88 pkt/sec. This means that although the number of failed links of Hybrid topology is twice that of the Single topology, it still performs better than the Single topology.

Table 1: Summary of simulation inventory for the percentage of failed links

Topologies	No of switches	No of servers	No of clients	No of links	10% of failed links	20% of failed links	30% of failed links	40% of failed links	No of configuration utilities	Simulation time (seconds)
Single FT	32	24	1	152	15	30	45	60	2	900
Hybrid FT	64	24	1	304	30	60	91	121	2	900

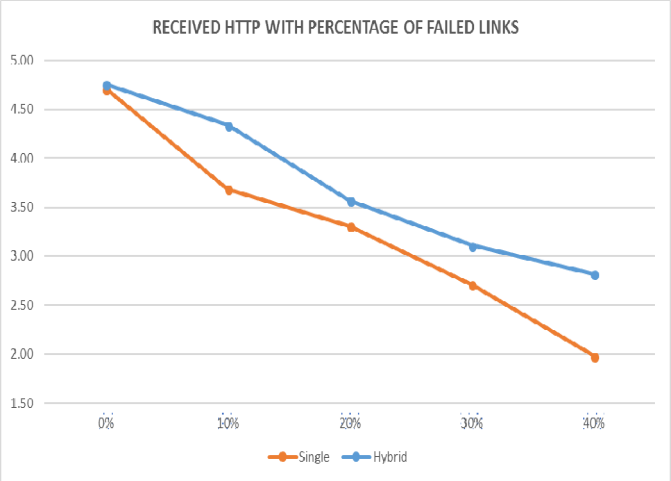

Figure 5: Graph of the Results of Received HTTP Packets based on percentage of failed links simulated at constant Inter Arrival Time of 4.0 seconds and with a constant frame size of 500,000 bytes.

Table 2: Summary of the results of HTTP packets based on percentage of failed links

Topologies	PERCENTAGE (%) OF FAILED LINKS				
	0%	10%	20%	30%	40%
Single FT Received packets	4.71	3.69	3.42	2.71	1.98
Hybrid FT Received packets	4.76	4.34	3.57	3.11	2.82

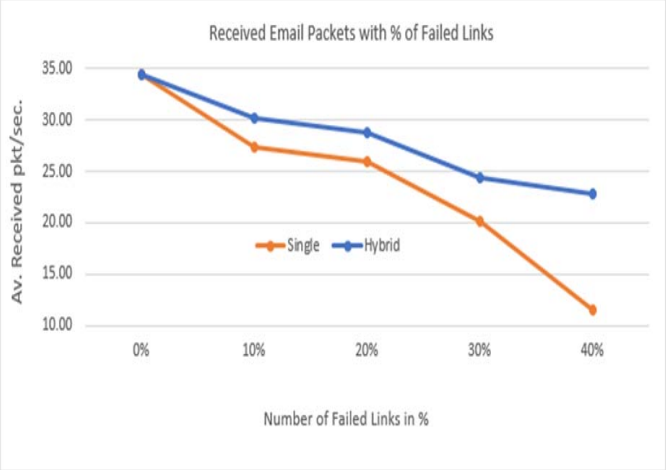

Figure 6: Graph of Results of Received Email Packets based on percentage of failed links simulated at constant Inter Arrival Time of 0.025 seconds and constant EMAIL size of 10,000,000 bytes.

Table 3: Summary of the results of received EMAIL packets based on percentage of failed links

Topologies	PERCENTAGE (%) OF FAILED LINKS				
	0%	10%	20%	30%	40%
Single FT Received packets	34.43	27.44	26.02	20.21	11.46
Hybrid FT Received packets	34.43	30.11	28.77	24.45	22.88

b. Result analysis based on failed links and switches

To evaluate our result in this regard by adding more failures into the network, we had to fail both links and switches before running the simulation. At this time, as shown in table 4, and because the number of links and switches of the Single design is twice that of the Hybrid design, we decided to use a different comparison yardstick. Thus, we had to multiply by 2 both the links and switches failed in the Single design for the Hybrid design. Therefore, for HTTP application as shown in figure 7 and table 5, the simulations were run at different interarrival times (secs) of 2, 4, 8, 10, and 12. As the inter arrival time is the time between each packet arrival and the next, it means that the smaller the interarrival time, the more packets are generated and vice versa. As a result, at an interarrival time of 12.0 sec, the Single design has an output of 0.62 pkt/sec, while the Hybrid design has an output of 1.01 pkt/sec. But as the interarrival time is being reduced to 2.0 sec, it means that more traffic or packets will be generated, and this will determine how fault tolerant and reliable a network could be to handle the huge traffic amidst failures. Based on this, Single design has a received output of 6.7 pkt/sec while the Hybrid design has 9.4 pkt/sec.; confirming that the Hybrid design is more fault tolerant than the Single design.

The same failure trend is also applicable to the EMAIL application based on the parameters shown in Table 5. So, the results of the simulation as shown in Figure 8 and Table 6 also confirms and validate the previous results. At an interarrival time of 0.1 sec, the amount of received packets for both Single and Hybrid designs are 6.78 and 8.6 respectively, with a difference of 1.82 pkt/sec. Meanwhile as the interarrival time was reduced to 0.00625 sec, both the Single and the Hybrid produced 111.48 pkt/sec and 141.32 pkt/sec respectively; having a difference of 29.84 pkt/sec. This difference in throughput shows the level of fault tolerance and graceful performance degradation the Hybrid design can help actualize in a cloud data center during huge traffic generation amidst failures.

Table 4: Summary of simulation inventory for the links and switches failures

Topologies	No of switches	No of links	No of clients	No of servers	No of failed links	No of failed switches	No of configuration utilities	Simulation time (seconds)
Single FT	32	152	1	24	10	5	2	900
Hybrid FT	64	304	1	24	20	10	2	900

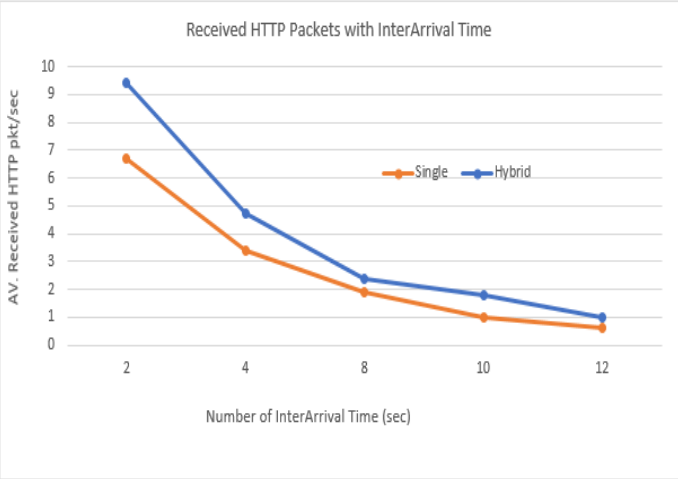

Figure 7: Graph of the Results of Received HTTP Packets based on failed links and switches simulated at different Inter Arrival Time and with a constant frame size of 500,000 bytes.

Table 5: Summary of the Results of Received HTTP Packets based on failed links and switches

Topologies	INTER ARRIVAL TIME (SECONDS)				
	2.0	4.0	***8.0***	***10.0***	12.0
Single FT Received packets	6.7	3.42	1.92	1.02	0.62
Hybrid FT Received packets	9.4	4.72	2.36	1.8	1.01

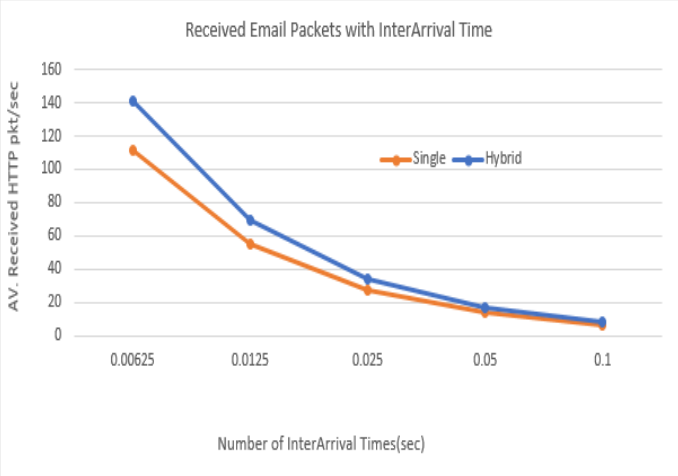

Figure 8: Graph of the Results of Received Email Packets based on failed links and switches simulated at different Inter Arrival Time and constant Email size of 10,000,000 bytes.

Table 6: Summary of the Results of Received Email Packets based on failed links and switches

Topologies	INTER ARRIVAL TIME (SECONDS)				
	0.00625	***0.0125***	***0.025***	***0.05***	0.1
Single FT Received packets	111.48	55.3	27.26	13.64	6.78
Hybrid FT Received packets	141.32	69.24	34.38	17.19	8.6

*B. **Trade-offs***

To balance the trade-offs between complexity and performance and choose the right topology at different levels of failures for both topologies will be based on the following:

Cost: Although, both topologies are built with same number of servers, but the cost of building the Single network is cheaper than the Hybrid because of less number of switches - the number of switches of Single topology is 32, while that of the Hybrid is 64. Apart from the cost of building these, topologies, putting into consideration the cost of their maintenance, it is understandable that the Single topology is easier to be maintained. But looking at this from a different perspective, one could deduce that since the output of the Single topology is not equal to the output of the Hybrid topology in relation to the number of failures; therefore, even if there are two different Single topologies, their output put together will not be equal to the Hybrid topology. It is expected that there is at least approximately same amount of output for both topologies looking at the number of failed links and switches, but that is not the case here. Also, it could be acceptable to say that maintaining our Hybrid topology will be cheaper than maintaining a double Single topology.

Response time: The download and upload response times for the Single topology are a bit lower than that of the Hybrid topology, though very negligible and remained constant for the period of simulation. This is expected since the Hybrid has greater number of links, which might cause a delay in routing a packet.

Reliability: The effect of the Single topology's low fault tolerance in both scenarios (based on the results obtained) is pronounced on its performance when compared with the Hybrid topology. Therefore, the compromise is that the Single topology is less complex, cheaper but less *fault tolerant and less reliable*. Meanwhile, the Hybrid topology's performance is better than the Single topology in all scenarios. So, the cost

of extra switches and the network complexity are the trade-offs for its reliability and sustained fault tolerance for graceful performance degradation.

V. CONCLUSION

As the backbone infrastructure for future information technology, cloud data center is meant to be designed in such a way that it could serve diverse computing needs; especially in this era of big data and everyday computing. To this end, we focused on achieving fault tolerant network believing that it is the bedrock for effective network performance. Therefore, our simulation results, which are based on a) percentage of failed links, for EMAIL and HTTP applications and b) failing of links and switches also for EMAIL and HTTP applications; together proved our Hybrid FT design to be fault tolerant and reliable. Hence, one should be able to attest to the role a bespoke design of data center network could play in improving its fault tolerance and reliability.

VI. FUTURE WORK

Although in our experiments we used 24 servers each for both Single and Hybrid Topologies, with 32 and 64 switches respectively. However, in our future work, we are working at experimenting with 100s to 1000s of switches and servers. So that our proposed cloud data center can actually be industrially useful. Having said that, our Hybrid design is something to reckon with because of its ability to prove fault tolerance and reliability in a cloud data center.

Reference

[1] H. Emesowum, A. Paraskelidis, and M. Adda. "Fault tolerance improvement for cloud data center," Journal of Communications, vol. 12, no. 7, pp. 412-418, 2017.

[2] Y. Liu, J.K. Muppala, & M. Veeraraghavan. A survey of data center network architectures., 2004, p.22pp.

[3] 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.

[4] M. Bradonji, B. Labs, & M. Hill. Scaling of Capacity and Reliability in data center networks categories and subject descriptors, 2, 3–5. Bilal, K., Khan, S., Zhang, L., & Li, H. (2013).

[5] 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.

[6] C. Guo et al., BCube: A high performance, server-centric network architecture for modular data centers. Proceedings of the ACM SIGCOMM 2009 conference on Data communication, pp.63–74.

[7] A. Akella, T. Benson, B. Chandrasekaran, C. Huang, B. Maggs, & D. Maltz. A universal approach to data center network design. Proc. of 16th Int. Conf. on Distributed Computing and Networking (ICDCN), 2015.

[8] R.M. Niranjan et al. "Portland: A scalable fault-tolerant layer 2 data center network fabric," ACM SIGCOMM Computer Communication Review, vol. 39, no. 4, pp. 39–50, 2009.

[9] M. Adda; 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

[10] 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.

[11] H. Emesowum, A. Paraskelidis, and M. Adda. "Fault tolerance capability of cloud data centers", 2017 IEEE 13th International Conference on Intelligent Computer Communication and Processing (ICCP 2017) pp.495–502.

[12] Park, H. W., Yeo, I. Y., Lee, J. R., & Jang, H. (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.

[13] Joshi, S.C. & Sivalingam, K.M. On fault tolerance in data center network virtualization architectures. 2013 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), pp.1–6.

[14] Lin, X.-Y. L. X.-Y., Chung, Y.-C. C. Y.-C., & Huang, T.-Y. H. T.-Y. A multiple LID routing scheme for fat-tree-based InfiniBand networks. 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.

[15] N. Farrington et al., "Helios: A Hybrid Electrical/OpticalSwitch Architecture for Modular Data Centers," Proc. ACM SIGCOMM '10, 2010, pp. 339–50

[16] For, E. Optical Interconnection Networks in Data Centers: Recent Trends and Future Challenges, (Sept. 2013), pp. 39–45.

[17] Configuring Applications and Profiles: OPNET, Optimum Network Performance.

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

[19] A. Peratikou and M. Adda, "Optimisation of extended generalised fat tree topologies," Chapter 1–10, January 2014.

[20] E. Zahavi, "Fat-Trees routing and node allocation providing non-blocking MPI global collectives for exascale jobs," *Electrical Engineering*, pp. 1–8, 2010.