Management of fault tolerance and traffic congestion in 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 ubiquitous computing, coupled with the emergence of big data and internet of things, there have been constant changes in every aspect of cloud data center communications - its network connectivity, data storage, data transfer, and architectural design. As a result of this, the amount of data transferable, and the frequency of data transfer have tremendously increased; causing device failures and traffic congestions. To cope with these changes so that performance can be sustained amidst device failures and traffic congestion, the design of fault tolerant cloud data center is important. A fault tolerant cloud data center network should be able to provide alternative paths from source to destination during failures so that there will not be abrupt fall in performance. But still with the ongoing researches in this regard, there has not been a robust cloud data center design that can boast of being suitable for alleviating the poor fault tolerance of cloud data center. In this paper, we proposed the improved versions of fat-tree interconnection hybrid designs derived from the structure called Z-fat tree; to address the issues of fault tolerance. Then, we compared these designs with single fat tree architecture with the same amount of resources for client server communication pattern such as Email application in a cloud data center. The simulation results obtained based on failed switches and links, show that our proposed hybrid designs outperformed the single fat tree design as the inter arrival time of the packets reduces.

Keywords—Fault Tolerance; traffic congestion; Cloud Data Center; Ubiquitous Computing; Big Data; and Internet of Things.

I. INTRODUCTION

The increase in the use of cloud data center became inevitable because of the rapid growth of internet-based applications, internet of things, big data transfer and analytics. In line with this, there is an expected proportional increase in the size and deployment of interconnection networks to facilitate communications. Therefore, the performance of the cloud data center must be considered in terms of its fault tolerance, congestion control, low latency and reliability for the effectual data communication and storage. In the same vein, the authors of [1] stated that due to the economies of scale in the trend of cloud computing owing to the growth in internet communication, increase in traffics, emergence of internet of things (IoT) and big data transfer; many researchers now focus on robust ways to improve on the cloud data center networks for better performance in terms of congestion control, availability, fault tolerance and reliability. It is also noteworthy that for general performance to increase in data center network, fault tolerance is an essential and unavoidable requirement; so that even during failure there will still be available paths for packet transfer [2]. According to Liu et al. in [3], data centers are prone to failures because of many switches, servers, and links. In support of this assertion, [6] also observed the unavoidable failures in data center architecture, and suggest that network design should be in a way that common failures be immediately recovered while maintaining a graceful performance degradation amidst such failures.

To keep abreast with these challenges in data center network, several network architectures were designed -Fat-tree, DCell, BCube and VLE [4, 5]. Fat-tree has its origin from the fixed topology, a subset of tree-based topology used in designing data center networks [3]. Fat tree is said to be widely used in designing data center networks [6, 7]; and has been undergoing different stages of improvement by researchers because of its significance, which is due to its ability to improve fault tolerance and congestion control because of its multi-paths from source to destination [4, 8, 9]. There is issue of scalability with the convectional fat tree, which led to generalized fat tree that has switches of same radix and speed port across the entire network; then the emergence of extended generalized fat tree (XGFT) that allows variable number of switch ports to be used at different level of the network [4, 10, 11]. Having said that, our work is based on a variant of fat tree called Z-node [12], which we used in our previous projects to prove that our hybrid designs (for link failures) are better than the single fat tree designs in alleviating fault tolerance in cloud data center.

Section II shows a review of related works; section III is the description of models; section IV is the analysis of simulation results; and finally, we drew conclusion based on the received packets to ascertain the level of fault tolerance for each cloud data center design.

II. RELATED WORKS

In recent years, researchers have been working round the clock to mitigate the technical challenges facing computer communication and networking technology. In data center network topologies, there have been some contributions to work-around the challenges of faults tolerance and graceful performance degradation amidst failures. However, some of these contributions have their strengths and weaknesses. For instance, the use of separation techniques to improve data center network performance as proposed by [13], failed to consider fault tolerance, which is the bedrock for effective and reliable network performance. In their work, the authors made 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 one used to transmit the ordinary data traffic, thereby making the source to destination transmission paths less congested. On the contrary, this means that the failure of any path will lead to total disruption of service. Nevertheless, with our proposed hybrids, (H_2^+) and reversed hybrid (H_2^-) there exist alternative paths that encourages graceful performance degradation more than the single fat tree ($\mathbb{Z}$) during device failures in data center.

The authors of [2] accepted the fact that fault tolerance is essential for reliability and availability to increase in data center [14], they therefore in their work, hosted a virtual data center on a physical data center to handle the server failures. They accomplished this by relocating the virtual machines that were hosted in the failed server to another server, then recovered fault and server utilization by 90%. Similarly, with the introduction of load balancing scheme in the network using clustering to allocate the virtual data center on its physical host; they reduced the impact of server failures. Insomuch that this work is aimed at improving fault tolerance in data center, it is rather time consuming to relocate the virtual machines during server failure and might cause poor performance due to delay. However, in our work, our primary concern is the commonest communication devices e.g. switches and links. Our previous works, which are based on only link failures [1, 14, 15], and this current work that is based on both switches and links failures, show that our proposed designs can improve the fault tolerance and congestion of a cloud data center network in real time.

The wavelength division multiplex links used in optical interconnection is another interesting new trend in designing data center. The Helios architecture as proposed by [16], is a circuit-based data center network with two-level hybrid consisting of either optical or electrical switches as core switches for high bandwidth used between the top of rack (ToR) switches; whereas the electrical packet switches are used for fast all-to-all communication between pod switches. Nevertheless, [17] identified the strength and weakness of using optical interconnection in the design of data center networks. The strength is that it provides high capacity, low power, and low latency; while the weakness is the issues of cost-effectiveness, scalability and fault tolerance that still pose a threat to the architecture. Based on this claim by [16], our proposed hybrid designs could be said to have an edge because of its cost-effectiveness, scalability, minimal network complexity, and fault tolerance.

In conclusion, based on the outcome of the survey carried out on fault tolerance properties by the authors of "On Performance Evaluation of Fault-Tolerant Multistage Interconnection Networks" as detailed in [18], shows that adding more hardware to interconnection networks for its improvement will not produce a better performance when compared to the original network. To this end, our hybrid designs could be appreciated because even with lesser number of links, fault tolerance was exhibited by the amount of received data when compared to the single fat tree with higher number of links. Correspondingly, full bisection bandwidth, deadlock freedom, and fault tolerance; which make fat tree a dominant choice for the design of data center are all inherent in our proposed designs.

III. MODEL DESCRIPTION

The explanation of Fat tree topology are in several literatures, e.g. [12, 19, 20, 21, 22] by the notation $FT(h;m_1,m_2..,m_h;w_1,w_2..,w_h)$, fat tree is being defined thus: h represents switch levels of the tree numbered from 0 at the bottom. The sequence m_1, m_2 represent the number of children each switch at level1 and level2 has respectively; while w_1, w_2 represent the number of parent-switches a host and a switch at level*0* and level1 has respectively [14]. However, as explained in our previous works [1, 14, 15], the construction of the variants of improved version of fat tree we use in comparing the fault tolerance of cloud data center is derived from the mathematical equations of switch level relationship, switch connectivity, and port mappings (section III, subsections A, B, and C). The level2 switches are connected to the clients with full connectivity, likewise, the servers at level*0* are connected to level1 switches in each zone/subtree using a full connectivity. The numbering of switches and its ports at every level and zone are from left to right starting from zero. To connect switch to switch in the case where there are no extra links, is by connecting each lower level switch to the quotient of the divisor (the greatest common divisor (gcd) of Rn+1 and Rn,) and the dividend (Rn+1); which is $Rn+1/gcd_{(Rn+1,\ Rn)}$. But, where extra links are used in the connection, we deploy the pattern used for Z-Fat tree by the authors of [12], (see Fig 1 to 7). The Z-Fat tree describes the number of root nodes per zone in its semantics and adds a degree of connectivity as $\mathbb{Z}$ (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 [Fig. 1].

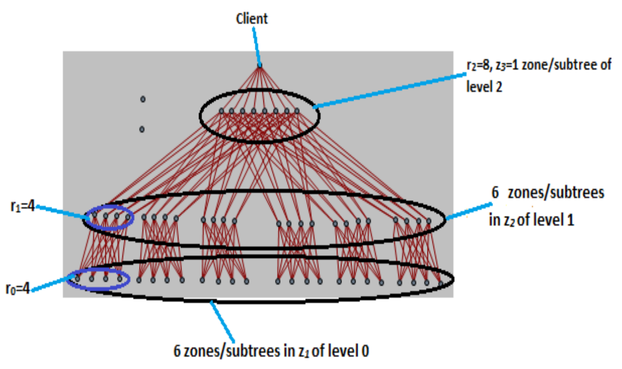

Client
r_2=8, z_3=1 zone/subtree of level 2
6 zones/subtrees in z_2 of level 1
r_1=4
r_0=4
6 zones/subtrees in z_1 of level 0

Fig. 1: Sample of Labelling showing positions of the notations used in describing the topologies [14]

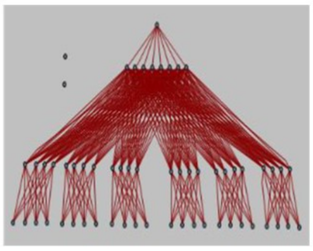


Fig. 2: Ƶ (2;4,6;4,8,1,4) one client

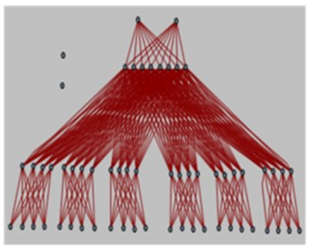


Fig. 3: Ƶ (2;4,6;4,8,1,4) two clients

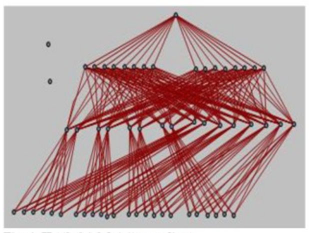


Fig. 4: H_2^+(2;6,4;2,8;1,1) one client

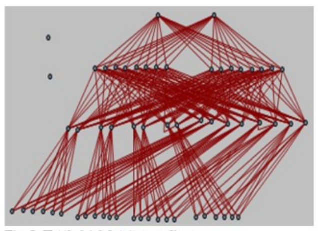


Fig. 5: H_2^+(2;6,4;2,8;1,1) two clients

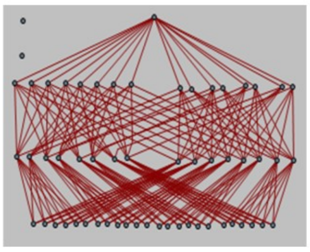


Fig. 6: H_2^- (2;6,4;2,8;1,1) one client

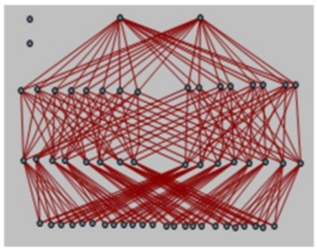


Fig. 7: H_2^- (2;6,4;2,8;1,1) two clients

Therefore, for our single topology, Fig. 2 and 3, Ƶ(2;4,6;4,8,1,4), 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 is the explicit degree of connectivity, which indicates that there are extra connections at level 2. For the Hybrid Fat tree: Fig. 4 and 5, 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 8links. It implies that each level2 switch has 24 down-ports to be mapped**.** Therefore, mapping level1 switch 0 in the first zone of Z_2 to level 2 switches is thus:

$$X_{p+1}= ((0\backslash 4)\ \%6) * 4/4+ p;$$
$$= (0\ \%6) * 1+ p,$$
$$= 0+p;$$

And $p \in \{0, 1, \ldots, R_n/\gcd(R_n, R_{n+1})-1\}$.

It implies that p=0 and X_{p+1} = 0. Therefore, level1 switch0 will be mapped to ports 0 of each level2 switches.

A. IP Address Translation

A network address translation setup that enables the servers of the data center to communicate with the clients across the internet. Detailed explanation is in our previous work [14].

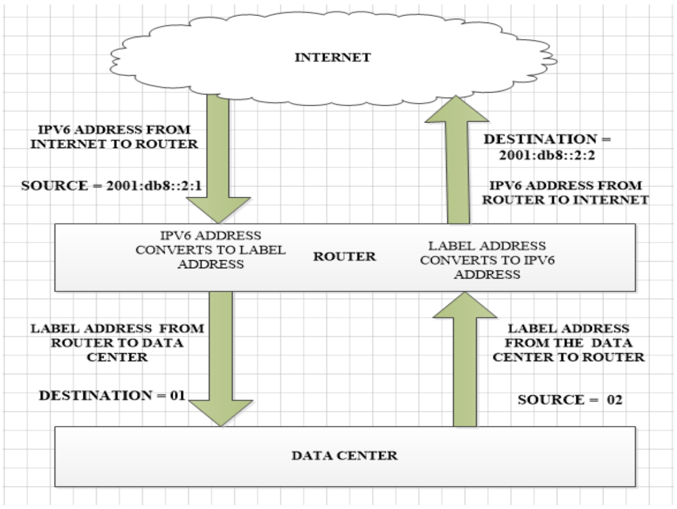

INTERNET
IPV6 ADDRESS FROM INTERNET TO ROUTER
SOURCE = 2001:db8::2:1
DESTINATION = 2001:db8::2:2
IPV6 ADDRESS FROM ROUTER TO INTERNET
IPV6 ADDRESS CONVERTS TO LABEL ADDRESS
ROUTER
LABEL ADDRESS CONVERTS TO IPV6 ADDRESS
LABEL ADDRESS FROM ROUTER TO DATA CENTER
DESTINATION = 01
LABEL ADDRESS FROM THE DATA CENTER TO ROUTER
SOURCE = 02
DATA CENTER

Fig. 8: Mapping Internet Address to Data Center Labels. [14]

IV. Analysis of Simulation Results

A. The Network Inventory of and Definitions

Table 1 is a summary of network inventory for the simulation of Email application carried out on Riverbed, where results of Received Packets were collected. A total of 24 servers and 32 switches were used across all topologies in all scenarios. All the networks were simulated using 2 configuration utilities: Application Definition and Profile Definition. [23] 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. In this paper, the client is where the profile definition is deployed, for modelling the behaviour of a user, and acts as traffic source. It represents the users over the internet retrieving information from the servers(cloud). A constant simulation time of 15 minutes with packet size of 10,000000 bytes were used across each design of all the three scenarios.

In scenario one, one client was used, and 5 switches were failed; while for scenario two, two clients were used for the simulation, with the same number of failed switches as in scenario one. However, in scenario three, we used two clients as was in scenario two but failed 5 switches and 6 links for the simulations.

because it is a hybrid. The sequence g_1=1 and g_2=1, indicates there are no extra connections. For the Reversed Hybrid Fat tree: Fig. 6 and 7, H_2^-(2;6,4;2,8;1,1), the topology is divided into two parts-left and right (mirror image). So, 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 that there are no extra connections. These sequences stand for each side of the topology in reversed form, thus it is called a reversed hybrid.

B. Switch Level Relationship

$$Rn+1 = R1+ \Delta(n-1)$$

(I) [14]

R_{n+1} representsnumber 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.

C. Switch Connectivity

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

where ***k*** *represents* $\epsilon \{0, 1, ..., R_{n+1}/gcd(R_n , R_{n+1})-1\}$;

(II) (Down-top connection) [12]

$$X_n = (R_n ((x_{n+1} \backslash R_{n+1}) \backslash Z_n) + (x_{n+1} \% R_{n+1}) * R_n/gcd(R_{n+1}, R_n) + k) \% R_n.$$

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

(III) (Top-down connection) [12]

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 level* 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. For example, connecting switch 0 at level2 to level1 switches for top-down connection H_2^- (Fig.6&7), using (III):

$$X_n = (8((0\backslash 2) \backslash 4) + (0\%2) *4+k) \%8= (0+0+k) \%8$$

where k $\epsilon\{0, 1, …, R_n/gcd(R_{n+1}, R_n)-1\}$.

So that, k = 0,1,2,3.

Therefore, If k= 0, (0+ k(0)) %8 = 0%8 = 0;

If k= 1, (0+ k(1)) %8 =1%8 = 1;

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

If k= 3, (0+ k(3)) %8 = 3%8 = 3.

Also, switch0 inter-connecting the left-hand-side:

$$X_n= (2((0\backslash 2) \backslash 4) + (0\%2) *1+k) \%2$$
$$= (0+0+k) \%2$$

where k $\epsilon\{0, 1, …, R_n/gcd(R_n, R_{n+1})-1\}$.

So that, k=0.

If k= 0, it implies (0+ k(0)) %2 = 0. Therefore, switch 0 at level2 will connect to: 0,1,2,3 level1 switches at right-hand-side; and switch 0 at the left-hand-side [15].

D. Port Mapping

$$X_{p+1} = ((X_n \backslash R_n) \%Z_{n+1}) *R_n/gcd(R_n, R_{n+1})+p$$

(IV) [13]

where ***p,*** set of switch ports to be mapped, and represented as*:* *p* ϵ *{0, 1, ...,* $R_n/gcd(R_n, R_{n+1})-1$*;* X_{p+1} ***r***epresents switch ports to be mapped at upper level. In Fig.2&3, at level 1, there are 6 zones for z_2, with r_1=4 in each and with one switch having

Table 1: Summary of Network Inventory

Scenario One: Topologies with one Client	No. of Switches	No. of Clients	No. of failed Switches	No. of Servers	No. of failed Links	No. of Configuration Utilities	Simulation Time in Minutes
Reversed Hybrid FT: 192 Links	32	**1**	**5**	24	**0**	2	15
Hybrid FT: 240 Links	32	**1**	**5**	24	**0**	2	15
Single FT: 296 Links	32	**1**	**5**	24	**0**	2	15
Scenario Two: Topologies with two Clients							
Reversed Hybrid FT: 208 Links	32	**2**	**5**	24	**0**	2	15
Hybrid FT: 256 Links	32	**2**	**5**	24	**0**	2	15
Single FT: 304 Links	32	**2**	**5**	24	**0**	2	15
Scenario Three: Topologies with two Clients							
Reversed Hybrid FT: 208 Links	32	**2**	**5**	24	**6**	2	15
Hybrid FT: 256 Links	32	**2**	**5**	24	**6**	2	15
Single FT: 304 Links	32	**2**	**5**	24	**6**	2	15

E. Comparing the Simulation Results

Table 2, 3 and 4 are the summary of the results of the Received Email Packets as shown on Fig. 9, 10 and 11 respectively. Across all the networks in all scenarios, we failed 5 switches for each design during simulation; and extra 6 links in each design of scenario three were failed. There are different Inter Arrival Time of the packets in milliseconds for each simulation as shown in the abovementioned tables. In scenario one, when the Inter Arrival Time was set at 25 milliseconds, the average received Email packet for the Reversed Hybrid FT with 192 links was 34.47 pkt/millisecond. At same 25 milliseconds, the Hybrid FT with 240 links was 34.61; while the Single FT with 296 links was 31.68 pkt/millisecond. Although the difference was not much, but as the inter arrival time decreases, the difference in the received packets became clearer between our proposed Hybrid FT designs and the Single FT design. Take for instance at 0.195 milliseconds, the Hybrid with 240 links has the highest average received packet of 4420.36 pkt/millisecond, followed by 4408.16 pkt/millisecond for the Reversed Hybrid with 192 links, then 4040.68 pkt/millisecond for the single design (see table 2). The difference in received packet per millisecond between Hybrid FT design and Single design is 379.32, while that of Reversed Hybrid FT design and the same Single FT design is 367.48. This shows that our proposed hybrid designs can tolerate fault more than the single design at high traffic.

To confirm our results, we increased the number of clients from one to two with the same number of failed switches, so that there will be high volume of packets transmitted. In a like manner as with the first scenario, at 25 milliseconds, the results of the second scenario across the three designs are almost the same value too, with just a slight difference.

However, as the time between each arrival of packet and the next (inter arrival time) tends towards zero, our proposed hybrid designs show significant margin of received packet than the single design. Therefore, at inter arrival time of 0.951 milliseconds, the received Email packets are 8850.12 pkt/millisecond, 8821.02 pkt/millisecond, and 8091.22 pkt/milliseconds for Hybrid, Reversed Hybrid, and Single designs respectively (as shown in table 3).

To further prove that our hybrid designs outperform the single fat tree design, we had to fail more 6 links in each design of scenario three, thereby making it a total of 5 switches and 6 links failed. Based on this, the summary of the results from fig. 11 as shown in table 4 shows that our proposed hybrid designs exhibit fault tolerance and graceful performance degradation. As the inter arrival time decreases from 25 milliseconds towards 0.195 milliseconds, a very clear difference in received email packets are noticed among the three different designs. At the inter arrival time of 25 milliseconds, the Single FT has a received email packets of 52.85 pkt/milliseconds, the Reversed Hybrid FT has 63.13 pkt/milliseconds, and the Hybrid FT has 66.09 pkt/milliseconds. However, as the inter arrival time is being decreased to the point of 0.195 milliseconds, the Single FT has 6799.44 pkt/milliseconds, the Reversed Hybrid FT has 8064.89 pkt/milliseconds, and the Hybrid FT has 8472.01 pkt/milliseconds. When these three scenarios are properly looked at, especially the second and third scenarios with two clients; one can deduce that our proposed designs are able to tolerate failures.

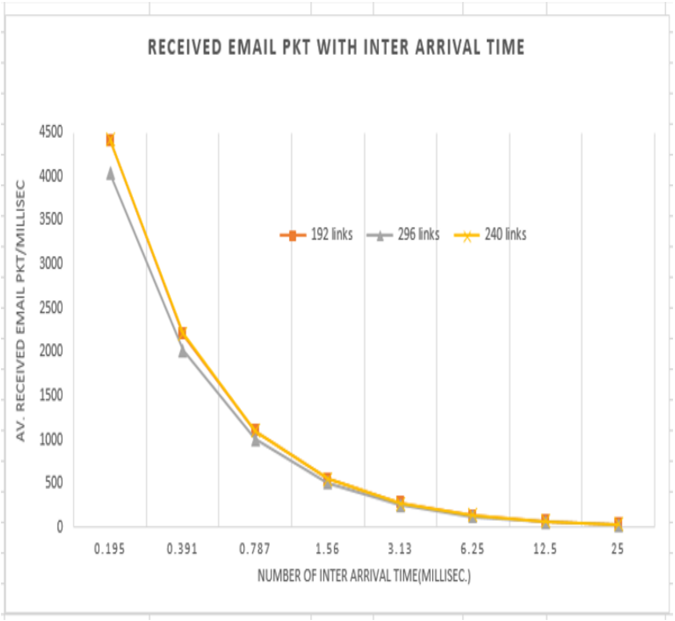


Fig. 9: Received Email Packet for two Clients, and 5 failed switches.

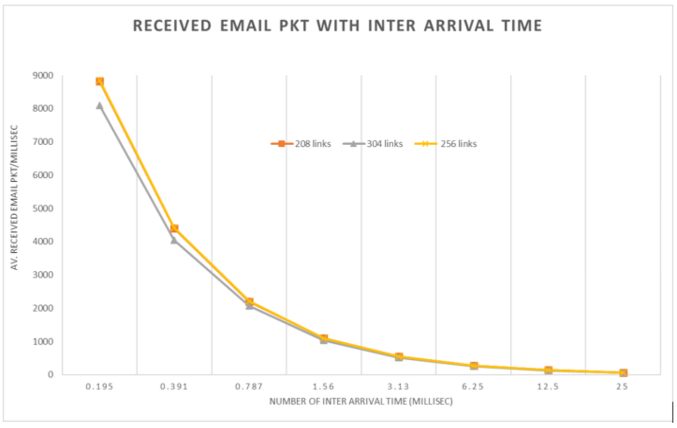


Fig. 10: Received Email Packet for two Clients, and 5 failed switches.

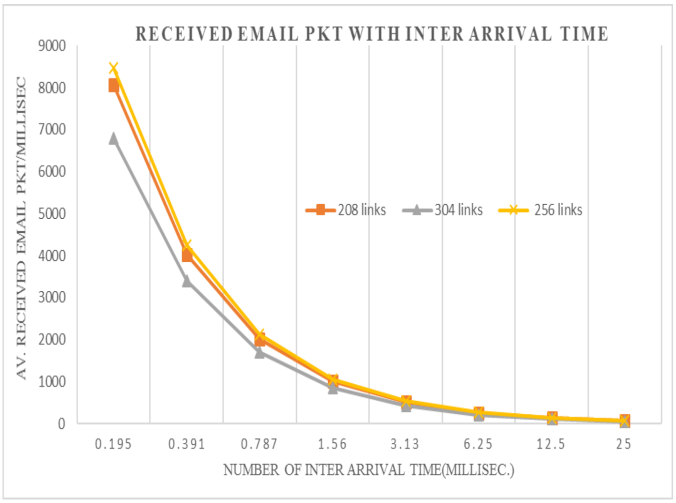


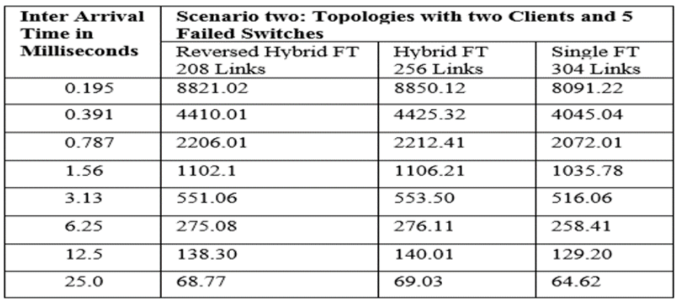
Fig. 11: Received Email Packet for two Clients, 5 failed switches and 6 failed links.

F. Tabular Summary of the Results of Received Email Packets for all scenarios

Table 2: Summary of the Results of Received Email Packets for Scenarios one.

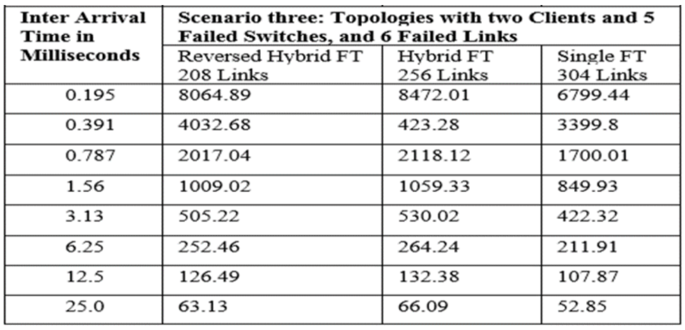
Inter Arrival Time in Milliseconds	**Scenario one: Topologies with one Client and 5 Failed Switches**		
	Reversed Hybrid FT 192 Links	Hybrid FT 240 Links	Single FT 296 Links
0.195	4408.16	4420.36	4040.68
0.391	2206.01	2210.19	2021.02
0.787	1102.08	1105.10	1010.34
1.56	551.58	553.96	505.13
3.13	278.08	277.01	252.80
6.25	137.89	138.27	126.24
12.5	68.95	69.01	63.22
25.0	34.47	34.61	31.68

Table 3: Summary of the Results of Received Email Packets for Scenarios two.



Inter Arrival Time in Milliseconds	**Scenario two: Topologies with two Clients and 5 Failed Switches**		
	Reversed Hybrid FT 208 Links	Hybrid FT 256 Links	Single FT 304 Links
0.195	8821.02	8850.12	8091.22
0.391	4410.01	4425.32	4045.04
0.787	2206.01	2212.41	2072.01
1.56	1102.1	1106.21	1035.78
3.13	551.06	553.50	516.06
6.25	275.08	276.11	258.41
12.5	138.30	140.01	129.20
25.0	68.77	69.03	64.62

Table 4: Summary of the Results of Received Email Packets for Scenarios two.



Inter Arrival Time in Milliseconds	**Scenario three: Topologies with two Clients and 5 Failed Switches, and 6 Failed Links**		
	Reversed Hybrid FT 208 Links	Hybrid FT 256 Links	Single FT 304 Links
0.195	8064.89	8472.01	6799.44
0.391	4032.68	423.28	3399.8
0.787	2017.04	2118.12	1700.01
1.56	1009.02	1059.33	849.93
3.13	505.22	530.02	422.32
6.25	252.46	264.24	211.91
12.5	126.49	132.38	107.87
25.0	63.13	66.09	52.85

CONCLUSION

The differences in the received packets between our proposed hybrid FT designs (H_2^+ and H_2^-), and the single design (Ƶ) for the Email application show that our proposed hybrids are a better choice to sustain fault tolerance in cloud data center amidst high volume of data transmission, and ranges of device failures. The fault tolerance capability exhibited by our proposed hybrid designs shows that switch and link failures in cloud data center could be managed in real time through bespoke design while waiting for repair or replacement. At healthy network, for all the scenarios, the received email packet per millisecond are almost the same value. But because more traffics are injected as the inter arrival time decreases, the network became more congested. The traffic congestion amidst failure of 5 switches and 6 links became a litmus test to ascertain the fault tolerance and robustness of these designs. Having said that, our proposed hybrid designs have been able to prove this through their sustained and better graceful performance degradation. This is also a confirmation of our previous work in [14] where similar experiments were carried out with a total failure of 80 links. We therefore look forward to real-life implementation of our proposal by industries.

REFERENCES

[1] 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), pages 495 – 502.

[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.

[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. (2008). A scalable, commodity data center network architecture. Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication - SIGCOMM ’08, 63.

[5] M. Bradonji, B. Labs, & M. Hill. (2014). 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).

[6] C. Guo et al., 2009. 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, (2015). A universal approach to data center network design. Proc. of 16th Int. Conf. on Distributed Computing and Networking (ICDCN).

[8] R. Niranjan Mysore, 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.

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

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

[11] 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.

[12] 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

[13] 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.

[14] 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.

[15] 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.

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

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

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

[19] Bogdanski, B., 2014. Optimized Routing for Fat-Tree Topologies. PhD Thesis submitted for the degree of Philosophy Doctor Department of Informatics Faculty of Mathematics and Natural Sciences University of Oslo January 2014.

[20] A. Peratikou, 2014. An optimized and generalized node for fat tree classes. PhD Thesis submitted for the degree of Doctor of Philosophy, Department of Computer Science, Faculty of Technology. University of Portsmouth, UK. April 2014.

[21] S.R. Ohring, M. Ibel, S.K. Das, and M.J. Kumar, 2002 “On generalized fat trees,” in IPPS, Santa Barbara, CA, USA, pp. 37–44. ISBN: 0-8186-7074-6.

[22] E. Zahavi, I. Keslassy, & A. Kolodny, (2014). Quasi fat trees for HPC clouds and their fault-resilient closed-form routing. Proceedings - 2014 IEEE 22nd Annual Symposium on High-Performance Interconnects, 41–48.

[23] OPNET Configuring Applications and Profiles: Optimum Network Performance. Available online at: Network lab configuring applications