# Fault Tolerance Improvement for Cloud Data Center

Humphrey Emesowum, Athanasios Paraskelidis, and Mo Adda
School of Computing, University of Portsmouth PO1 3HE, United Kingdom
Email: {humphrey.emesowum, athanasios.paraskelidis, mo.adda}@port.ac.uk

***Abstract*** — One of the main research concerns of Data Center Network designs is the achievement of reasonable throughput during multiple failures. To design a fault tolerant data center network that will give operators the operational efficacy they needed to achieve such reasonable throughput is a hard nut to crack. In this paper, we proposed an improved version of fat-tree interconnection to address the issues of fault tolerance. Fat-Tree is a well-known architecture widely used in data center networks due to its congestion control and fault tolerance capabilities attributable to its availability of alternative paths from source to destination. Our focus is on cloud data center for client to server communications, thus Email and FTP applications were run separately on all the designs. Using the same number of servers and switches, we proved that our proposed Hybrid and Reversed Hybrid designs outperformed Fat-tree topology designs at multiple failures. The results show an improvement on fault tolerance capability, and cost-effectiveness for cloud data center.

***Index Terms***—Fat-Tree, fault tolerance, cloud data center, graceful performance degradation, reversed hybrid

## I. Introduction

In this era of cloud computing with a lot of internet-based services, as well as big data and internet of things being stored and retrieved from data center network; it is pertinent to examine the fault tolerance capabilities of cloud data center networks so that the right choice of topology can be made during deployment. It is also worthwhile to bear in mind that for performance, reliability, and availability 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 [1]. Liu et al. pointed out that one of the undoubtable features of data center is that they are prone to failures, which is because of many switches, servers, and links [2].

These observations agree with the assertion by [3]-[5] that since failure in data center infrastructure is inevitable, the network design must be in such a way that any common failure that occurs must be recovered from immediately, while adequate performance must also be maintained amidst such failure. Furthermore, in line with the need for communication growth and increase in traffics in cloud data center networks, new architectures need to be designed. And because these architectures contribute majorly in data center design - as a backbone, it is therefore needful to be very careful in the design consideration [6].

In view of this, and to work around abovementioned challenges, Fat Tree topology, which originates from fixed topology has been in use to design a data center network [2]. This is due to its ability to improve fault tolerance and congestion control because of its multi-paths from source to destination [7]-[9]. However, the conventional Fat tree has been known for lacking scalability, has a single point of failure and unmanageably high switch ports towards the root of the topology [10], [11]. Therefore, the extended generalized fat trees came into existence through Ohring et al., used for different performance requirements because it allows variable number of switch ports to be used at different levels of the network [10], [12]-[13]. Our work is derived from that of the authors in [14], called Zoned-Fat tree (Z-Fat tree,). An extension of Fat-Trees providing 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. Based on this, we have proposed bespoke hybrid designs of Fat-tree for the improvement of cloud data center networks that will tolerate fault, lessen congestion, and guarantee a graceful degradation of performance during multiple failures.

Subsequently, in Section 2, we succinctly looked at some related works to improve data center performance; Section 3 detailed the model design and description; Section 4 is where the simulation results for Email and FTP applications on different topologies are analyzed, compared, and contrasted. In the last Section, 5, we drew conclusion based the fault tolerance capability of each design, which is visible from the throughput performance under multiple failures.

## II. Related Works

Due to the economies of scale in the trend of cloud computing because of 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.

The authors in [1] acknowledged that for reliability and availability to increase in data center, fault tolerance

Manuscript received March 30, 2017; revised July 7, 2017.
Corresponding author email: humphrey.emesowum@port.ac.uk.
doi:10.12720/jcm.12.7.412-418

is an essential and inevitable requirement. They therefore proposed the use of fault tolerance mechanism in virtual data center hosted by physical data center to handle server failures. They achieved this by relocating the virtual machines that were hosted in the failed server to another server. By doing this they recovered all fault and server utilization by 90%. They also introduced a load balancing scheme to the network by using clustering to efficiently allocate the virtual data center on the physical data center. By so doing the impact of server failure in virtual data center was reduced by allocating virtual data center across the physical data center network. Although their work aimed at improving the fault tolerance of data center, but it is only based on server failure; also processing time for the relocation of the virtual machine during failure is time consuming and might disrupt the reliability sought for. Meanwhile our work is targeted at the commonest communication failure in data center, which are switches and links failures; and our designs improve the fault tolerance capability of the network in real time.

In [15], Suchara et al proposed a "traffic engineering and failure recovery mechanisms to help the evenly distribution of traffic, failure recovery, enable reliable data delivery in the presence of link failures and reduce operational costs of data centers. This architecture comprises: a precomputed multipath routing that ensures continuous connectivity in the event of failure; path-level failure detection that detects and recovers failures performed by the ingress router, though it does not learn a failed link only a path failure; and local adaptation to path failures for traffic rebalancing on the healthy paths upon path failure detection on the network. However, the shortcomings of this proposal give our work an edge over it; such as the inability to proactively detect a faulty device, even as the path detection is after failure has occurred. This will cause tremendous delay when transferring from the failed path to the pre-computed alternate paths, which may cause traffic disruptions in data center. Nevertheless, our hybrid designs are cost effective because we used same amount of resources of single topology for our hybrid topology and in some cases with fewer links, which helps reduce network complexity. The traffic disruption encountered in this proposal [15] caused by delay in transferring traffic from failed to healthy path is another downtime that our topologies do not experience, therefore making our work fault tolerant with a graceful performance degradation for data center communications.

In a bid to improve the performance of data center network, the authors in [16] came up with traffic separation techniques. With this mechanism, they separated big data network traffic from that of the ordinary data center traffic because they believed the network problem of big data centers is that big data traffic severely affects data center network. This guarantees that no coexistence of different traffics on the same path so that transmission journey of the big data traffic will be short and effective; thereby improving the traffic congestion in data center network. Well, the concept is good but they failed to realize that fault tolerance is the bedrock for reliability and availability in data center [1]. Therefore, with our robust designs that encompass fault tolerance, congestion control, and graceful performance degradation, the issue of traffic congestion is settled.

Furthermore, there are other interesting new trends in designing data center network with the use of optical interconnection based on wavelength division multiplex links. A typical example is called the Helios architecture [17], a 2-layer hybrid circuit-based data center network that uses optical and commodity switches. Its core switches are either optical circuit switches or electrical switches, while the top of the rack (ToR) switches are typical packet switches. For high bandwidth and long-lived communications, the optical circuit switches are used between the ToR switches, whereas the electrical packet switches are used for fast all-to-all communication between pod switches. However, [18] identified that although optical interconnection provides low latency, reduced power, and high capacity data center networks; but scalability, cost-effectiveness, and fault tolerance are its greatest challenges. Therefore, with these pros and cons of the optical interconnects, we could be proud of our fat tree-based data center designs because they can be efficiently scaled, cost effective, and fault tolerant.

In a nutshell, to the best of our knowledge, our proposed designs meet the three properties that make fat tree as a dominant choice in high performance interconnect, as stipulated in [19], which are: (i) deadlock freedom that makes it possible to route packets without using virtual channels; (ii) inherent fault-tolerance for easily handling of faults with the existence of multiple paths from source to destination; and (iii) full bisection bandwidth, where network sustains full speed communication between its two halves. And with our reversed hybrid that has an exact replica of the number of upward paths in the downward direction, full bisection, deadlock freedom, and fault tolerance are all achievable.

## III. Model Description

Fat trees, as discussed in [14], [19]-[22], with the notation $FT(h;m_1,m_2..,m_h;w_1,w_2..,w_h)$ was 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 a host and a switch at level$l-1$ and level1 has respectively. To construct our fat tree variants designed for comparing fault tolerance capability, it is pertinent to start from the basis, which are the mathematical equations for switch level relationship, switch connectivity and port mapping. By default we used full connectivity to connect the servers at level$l$-1 to level1 switches for each first zones, and the numbering of switches and its ports at every level are from left to right starting from zero. Where there is no added extra links, the switch to switch connection is done by connecting each lower level switch to the quotient of the divisor (the greatest common divisor (gcd) of $R_{n+1}$ and $R_n$,) and the dividend ($R_{n+1}$); which is $R_{n+1}/gcd_{(Rn+1,\ Rn)}$. Meanwhile where extra links are used in the connection, we introduced the pattern used for Z-Fat tree by the authors of [14], [20]. 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.

Therefore, for our single topology, 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. But Fig. 2, Ƶ (2;4,6;4,8;1,4) shows the sequence $g_1$=1 and $g_2$=4, indicates there are extra connections at level 2.

For the Hybrid FT ($H_2^+$) designs, the same semantics used in Z-fat tree is applicable. For example for Fig. 3 $H_2^+$(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. In the same manner, the explanation for Fig. 3 is applicable to Fig. 4 H2+(2;4,6;4,8;1,1) with the sequence r1=4 and r2=8 referring to the number of roots nodes inside each zones z2 and z3 respectively.

For the Reversed Hybrid FT ($H_2^-$), Ƶ ($h$; $z_1$, $z_2$, …,$z_h$; $r_1$, $r_2$, …,$r_h$; $g_1$, $g_2$, …,$g_h$) 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 and level2. So that Fig. 5, $H_2^-$(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. Switch Levels Relationship*

$$R_{n+1} = R_1 + \Delta\ (n\text{-}1) \tag{1}$$

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

For example, Fig. 1 is simply: $R_{n+1} = R_1(4) + \Delta(4)\ (n(2)\text{-}1)$; this shows that the total number of level 2 switches is 8 since level 1 switch is 4 and a difference of 4.

## *B. Switch Connectivity*

$$X_{n+1} = (R_{n+1}((x_n \backslash R_n)\backslash Z_{n+1}) + (x_n \% R_n) * R_{n+1}/\mathrm{gcd}_{(Rn,Rn+1)} + k)\%\ R_{n+1}. \tag{2}$$

where ***k*** *represents* $\epsilon$ *{0, 1, ...,*$R_{n+1}$*/gcd(*$R_n$, $R_{n+1}$*)-1}.*[16] $\boldsymbol{X_{n+1}}$ represents the number of switch sought after at the upper level. $\boldsymbol{R_{n+1}}$ represents the total number of switches at the upper level. $\boldsymbol{x_n}$ 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 $\boldsymbol{R_{n+1}}$. $\boldsymbol{Z_{n+1}}$ represents the number of zones from upper level $\boldsymbol{n_{+1}}$. ***gcd*** is an acronym for Greatest Common Divisor used to get the exact number of $\boldsymbol{R_{n+1}}$ switches that $\boldsymbol{x_n}$ will connect to.

For example, connecting level 1 switch 0 to level 2 switches (in Fig. 1):

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

$\mathbf{X_{n+1}}$ = (0 + 0+k)%8;  = (0+k)%8.

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

If k = 0, it means that (0+k(0))%8; = 0%8 = 0.

If k = 1, it means that (0+k(1))%8; = 1%8 = 1. Therefore, switches to be connected to at level 2 are: 0 and 1.

## *C. Port Mapping*

$$X_{p+1} = (((X_n \backslash R_n)\% Z_{n+1}) * R_n/\mathrm{gcd}(R_n, R_{n+1}) + p) \tag{3}$$

where **p** represents $\epsilon$ {0, 1, …, $R_n$/gcd($R_n$ , $R_{n+1}$)-1}. [16], $\mathbf{X_{p+1}}$ represents switch ports to be mapped at upper level. **p** represents set of $R_{n+1}$ switch ports to be mapped with $\mathbf{X_n}$

To map the first switch's ports of first zone in level 1 to its corresponding level 2 switch ports. Since $Z_{n+1}$ represents number of zones from upper level $R_{n+1}$. Using Fig. 1 as an example, at level 1, there are 6 zones for $Z_2$ within zone $Z_3$, with $r_1$=4 in each. With 6 zones in level 1, implies that the down port of each level 2 switch is divided into 6 zones numbered from 0 to 5. Mapping level 1 switch 0 in the first zone of $Z_2$ to level 2 switches: $X_{p+1}$ =((( $X_n$\$R_n$) %$Z_{n+1}$)* $R_n$/gcd$_{(Rn\ ,\ Rn+1)}$+ p). where p $\epsilon$ {0, 1, …, $R_n$/gcd($R_n$ , $R_{n+1}$)-1} and $X_{p+1}$ = upper switch ports.

therefore,

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

$= ((0 \% 6) * 1 + p)$

$= 0 + p$ and $p \in \{0, 1, \ldots, R_n / \gcd_{(R_n, R_{n+1})} - 1\}$.

therefore $X_{p+1} = 0$.

Hence, level 1 switch 0 in the first zone of $Z_2$, will be mapped to ports 0 of level 2 switches where it is being connected to.

### D. *Single FTs (Ƶ)*

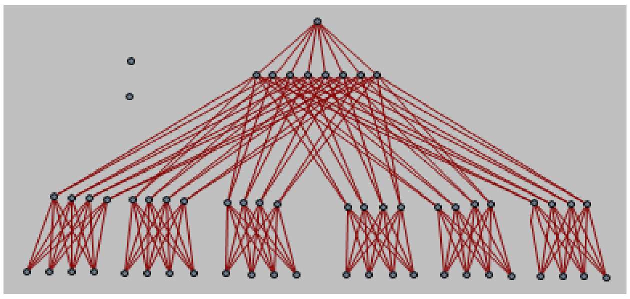

Fig. 1. Ƶ (2;4,6;4,8,1,1). The sequence $r_1$=4 and $r_2$=8, refers to the roots of the zones $z_2$=6 and $z_3$=1; the sequence $g_1$=$g_2$=1 is the explicit degree of connectivity.

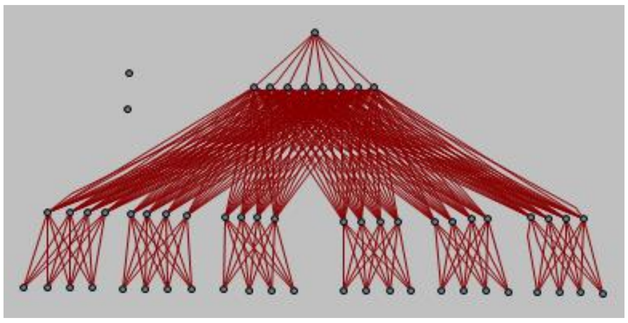

Fig. 2. Ƶ (2;4,6;4,8,1,4). The sequence $r_1$=4 and $r_2$=8, refers to the roots of the $_{\text{zones}}$ $z_2$=6 and $z_3$=1; the sequence $g_1$=1 and $g_2$=4 is the explicit degree of connectivity.

### E. *Hybrid FT ($H_2^+$)*

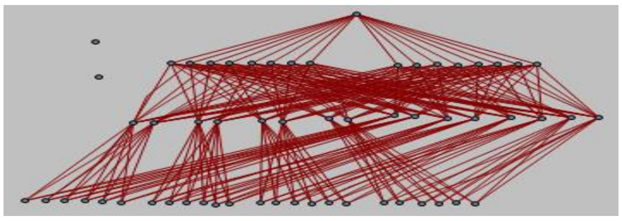

Fig. 3. **$H_2^+$(2;6,4;2,8;1,1).** The sequence $r_1$=2 and $r_2$=8, refers to the roots of the zones $z_2$=4 and $z_3$=1; the sequence $g_1$= and $g_2$=1 is the explicit degree of connectivity.

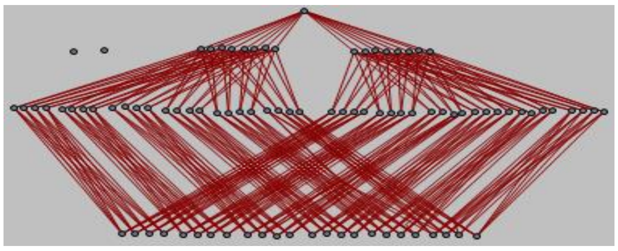

Fig. 4. **$H_2^+$(2;4,6;4,8;1,1)** The sequence r1=4 and r2=8, refers to the roots of the zones z2=6 and z3=1; the sequence g1=g2=1 is the explicit degree of connectivity.

This is a hybrid with 64 switches instead of 32 switches like other topologies

### F. *Reversed Hybrid FT ($H_2^-$)*

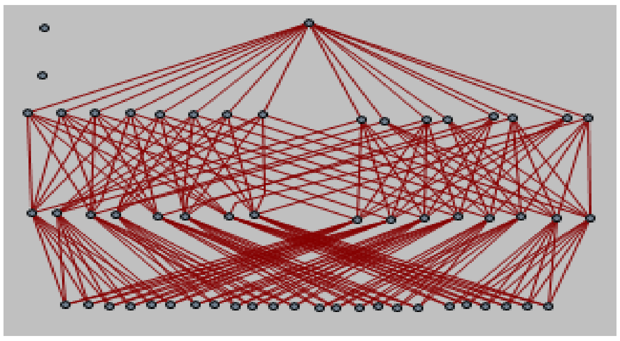

Fig. 5. $H_2^-$(2;6,4;2,8;1,1). The sequence $r_1$=2 and $r_2$=8, refers to the roots of the zones $z_2$=4 and $z_3$=1; the sequence $g_1$=$g_2$=1 is the explicit degree of connectivity.

Switch level connectivity and port mapping for the right-hand side of the topology, is done based on up-down pattern (from upper to lower level), whereas, the left-hand-side of the topology is connected and mapped like the given example that was done in down-up basis (lower to higher level, in subsection B and C).

### G. *IP Address Translation*

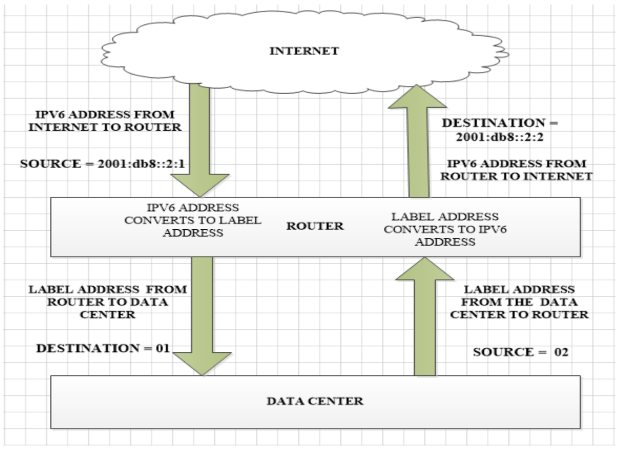
INTERNET
IPV6 ADDRESS FROM INTERNET TO ROUTER
SOURCE = 2001:db8::2:1
DESTINATION = 2001:db8::2:2
IPV6 ADDRESS FROM ROUTER TO INTERNET
IPV6 ADDRESS CONVERTS TO LABEL ADDRESS
ROUTER
LABEL ADDRESS CONVERTS TO IPV6 ADDRESS
LABEL ADDRESS FROM ROUTER TO DATA CENTER
DESTINATION = 01
LABEL ADDRESS FROM THE DATA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

Fig. 6. Mapping internet IP address to data center labels

Fig. 6 is a Network address translation setup that enables the servers of the data center to communicate with the clients on the internet; comprising internet, router, and data center. When the client from the internet wants to communicate with a server in the data center, it becomes impossible because the servers in the data center are using address label, which are logical numbers; while the client from the internet are using IP version 6 address. So, to make the communications possible, an IP translation router must map the incoming IPv6 address of arriving packets (e.g. source address = 2001:db8::2:1) to one of the servers' label address (e.g. destination label = 01). Meanwhile, as the server is returning the request to the client across the internet, the router also must map the label address of the server (e.g. source label = 02) to IPv6 address (e.g. destination address = 2001:db8::2:2) before it can be sent across the internet.

TABLE I: SUMMARY OF NETWORK INVENTORY USED FOR SIMULATION ON RIVERBED

| NO | TOPOLOGIES | SWITCHES | CLIENTS | SERVERS | NUMBER OF LINKS | CONFIGURATION UTILITIES | SIMULATION TIME |
|---|---|---|---|---|---|---|---|
| 1 | SINGLE XGFT Ƶ (2;4,6;4,8;1,1) | 32 | 1 | 24 | 152 | 2 | 900SEC |
| 2 | SINGLE XGFT Ƶ(2;4,6;4,8;1,4) | 32 | 1 | 24 | 296 | 2 | 900SEC |
| 3 | HYBRID XGFT $H_2^+$(2;6,4;2,8;1,1) | 32 | 1 | 24 | 240 | 2 | 900SEC |
| 4 | HYBRID XGFT $H_2^+$(2;4,6;4,8;1,1) | 64 | 1 | 24 | 304 | 2 | 900SEC |
| 5 | REVERSED HYBRID XGFT $H_2^-$(2;6,4;2,8;1,1) | 32 | 1 | 24 | 192 | 2 | 900SEC |

As shown in Table I and represented on the graphs, 152 links represents the single FT without extra links Ƶ(2;4,6;4,8;1,1); 296 links represents the single FT with extra links: Ƶ(2;4,6;4,8;1,4); 240 links represents the hybrid FT without extra links: $H_2^+$(2;6,4;2,8;1,1); 304 links represents the hybrid without extra links but with double number switches $H_2^+$(2;4,6;4,8;1,1); and finally 192 links represents the reversed hybrid FT without extra links: $H_2^-$(2;6,4;2,8;1,1).

## IV. ANALYSIS OF SIMULATION RESULTS

### A. Email Results

The simulation for the three Fat-tree designs: Ƶ; $H_2^+$; and $H_2^-$ for EMAIL applications were run using simulation time of 900 seconds with packet size of 10,000000 bytes. At constant interarrival times of 0.025 seconds.

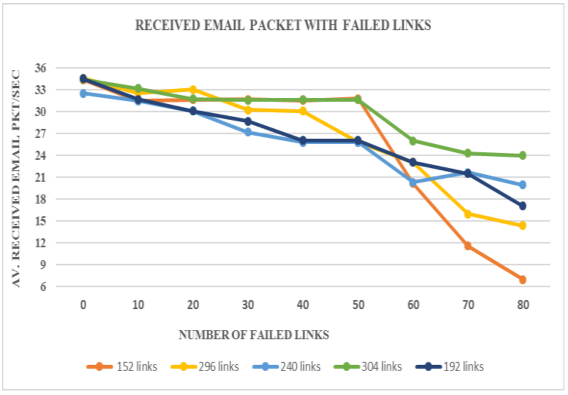

Fig. 7. Received email packets with failed links.

From the results of Fig. 7, it is clearly seen that the hybrid topologies outperformed the single topologies as the number of failed links increase. The graph shows that the single topology with 152 links was doing well up to the point where 50 links were failed, but as more links failed, there was a serious drop in performance. The same thing applies to the extra links Fat-tree topology with 296 links, which also shows a significant decrease in performance as more links failed. However, the hybrid topologies showed a graceful performance degradation as the number of failures increase.

Although the Single fat-tree topology with 156 links Ƶ(2;4,6;4,8;1,1) is half of the Hybrid topology with 304 links $H_2^+$(2;4,6;4,8;1,1**)**, but the average received packet of the former when 80 links were failed (6.97 pkt/sec) is not up to half of the average received packet of the latter when 80 links were failed (24 pkt/sec). Meanwhile, the other hybrid designs $H_2^+$(2;6,4;2,8;1,1), and $H_2^-$(2;6,4;2,8;1,1); with same number of switches and servers and fewer links, outperformed the single fat-tree designs. These results show that the hybrid designs can tolerate fault more than the single fat-tree designs and should be a better option.

### B. FTP Results

The simulation for the three Fat-tree designs: Ƶ; $H_2^+$; and $H_2^-$ for FTP Applications were run using simulation time of 900 seconds with packet size of 100,000 bytes. At constant inter- request time of 0.5 seconds. The graph shows the decline in the number of average received packets/secs from a zero failed link to 80 failed links.

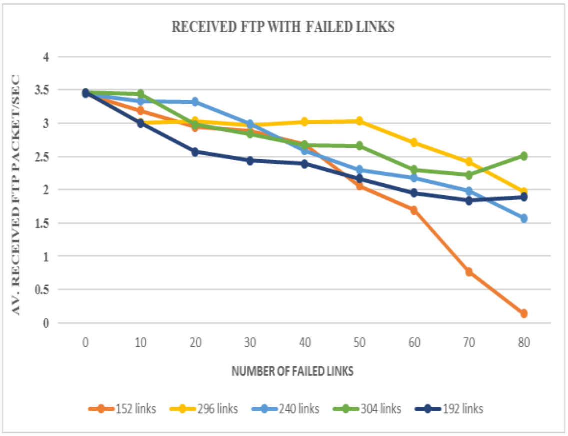

Fig. 8. Received ftp packets with failed links.

From the results shown in Fig. 8, it is undoubtable that the reversed hybrid with 192links performed better than the single fat-tree topology with 296links. The judgement is because the reversed hybrid has 41 percent of failed links and outputted 1.89pkt/sec at 80 failed links, while the single with 296links has 27 percent of failed links and has a throughput of 1.97 pkt/sec. Also, the former has less complexity than the latter.

Likewise, the single topology with 152links has a 52 percent of failed links and produced throughput of 0.14 pkt/sec, which is nowhere nearer to the received packet of the double hybrid $H_2^+$(2;4,6;4,8;1,1) of 26 percent failed links that has 2.57 pkt/sec. Although the double hybrid is twice the single fat tree, but the results from when 50 links were failed to 80 failed links are with large margin.

Fig. 9 shows the graph of the percentage of FTP packet loss with failed links. The graph tells us the exact percent of packet that was lost during the simulation at different stages of failed links for each of design. We calculate the packet loss using equation 10.

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

The graph of Fig. 9 shows that at zero failed links, the two single topologies have 75% packet loss. But as the number of failed links are increased to 80, the packet loss increased to 76%. Meanwhile the percentage of packet loss for the hybrids topologies at zero failed links and at 80 failed links are the same, 74% and 75% for hybrid and reversed hybrid respectively.

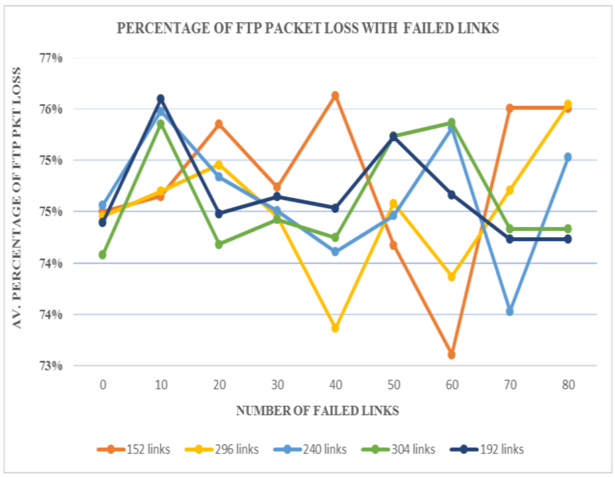

Fig. 9. Percentage of ftp packet loss with failed links

## V. Conclusion

Comparing the overall results of both EMAIL and FTP applications for the single fat-tree topology of 296 links Ƶ(2;4,6;4,8;1,4), with the reversed hybrid topology of 192 links $H_2^-$(2;6,4;2,8;1,1); shows that the latter has a better performance, tolerates faults, and will help reduce network complexity. Our reversed hybrid design shows that fault tolerance cannot only be achieved by adding more links and redundant devices, rather a bespoke design plays a greater role. The uniqueness of this design is that it has a replica of alternative paths for upward traffic forwarding to the client, on the downward traffic forwarding to the servers for the achievement of full bisection, deadlock freedom, and fault tolerance. This helps improves the challenges of 'nearest common ancestor switch' that is common in Fat-trees that made it prone to single point of failure. Therefore, with all these qualities found in our reversed hybrid topology, it proves that it is a better topology for cloud data center amongst other cloud data centers if fault tolerance is the prime target.

In like manner, the tremendous differences seen in the received packets between the hybrid topology $H_2^+$(2;4,6;4,8;1,1), and the single topology Ƶ(2;4,6;4,8;1,1); for both EMAIL and FTP applications show that the hybrid is the best bet if fault tolerant network is needed in a data center. The cost of the extra switches is the trade-off for the robust topology. These results speak volume of our proposed designs as they will aid the actualization of a fault tolerance and graceful performance degradation in a cloud data center.

## References

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

[2] Y. Liu, J. K. Muppala, and M. Veeraraghavan, "A survey of data center network architectures," 2014, p. 22.

[3] C. Guo, *et al*., "BCube: A high performance, server-centric network architecture for modular data centers," in *Proc. ACM SIGCOMM Conference on Data Communication*, 2009, pp. 63–74.

[4] G. Zarza, *et al*., "FT-DRB: A method for tolerating dynamic faults in high-speed interconnection networks," in *Proc. 18th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, 2010, pp. 77–84.

[5] L. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[6] Y. Wu, *et al*., *Scalable Computing and Communications: Past, Present, and Future*, 2012, pp. 1–6.

[7] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM 2008 Conference on Data Communication*, 2008, pp. 63–74.

[8] R. N. Mysore, A. Pamboris, N. Farrington, *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. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. 7th USENIX Conference on Networked Systems Design and Implementation*, 2010, p. 19.

[10] 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.

[11] Y. Sun, J. Chen, Q. Liu, and W. Fang, "Diamond: An improved fat-tree architecture for large-scale data centers," *Journal of Communications*, vol. 9, no. 1, pp. 91–98, 2014.

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

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

[14] "Routing and fault tolerance in Z-Fat tree, adda Mo: Peratikou adamantini," *IEEE Transactions on Parallel and Distributed Systems*, no. 99, p. 1, 2017.

[15] M. Suchara, *et al*., "Network architecture for joint failure recovery and traffic engineering categories and subject descriptors," *ACM SIGMETRICS*, pp. 97–108, 2011.

[16] H. W. Park, I. Y. Yeo, J. R. Lee, and H. Jang, "Study on big data center traffic management based on the separation of large-scale data stream," in *Proc. Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2013, pp. 591–594.

[17] N. Farrington, *et al.*, "Helios: A hybrid electrical/opticalswitch architecture for modular data centers," in *Proc. ACM SIGCOMM '10*, 2010, pp. 339–50.
[18] E. For, Optical Interconnection Networks in Data Centers: Recent Trends and Future Challenges, Sept. 2013, pp. 39–45.
[19] B. Bogdanski, "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, "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. Ohring, *et al*., "On generalized fat trees," in *IPPS, Santa Barbara*, CA, USA, pp. 37–44.
[22] E. Zahavi, I. Keslassy, and A. Kolodny, "Quasi fat trees for HPC clouds and their fault-resilient closed-form routing," in *Proc. 22nd Annual Symposium on High-Performance Interconnects*, 2014, pp. 41–48.

**Humphrey C. Emesowum,** PhD research student at the School of Computing, University of Portsmouth. He is researching on the Improvement of Fault Tolerance Capability on Cloud Data Center Networks. He Obtained his BSc in Network Engineering and Telecommunication Studies from Nottingham Trent University in 2012. In 2014, he obtained his Master's degree in Information Technology Management. He is a member of The Institution of Engineering and Technology IET.

**Dr. Athanasios Paraskelidis,** A senior Lecturer, Deputy Admissions Tutor for Undergraduate Degrees University of Portsmouth. He is currently lecturing Computer Systems Architecture and Networks; Network Fundamentals; and Interaction in computer Systems. He obtained a PhD in Wireless Network Segregation Utilising Modulo in Industrial Environments from the University of Portsmouth. He has acted as member of the Technical Program Committee panel for some international conferences and currently a Co-Investigator on Future Technologies for Construction and Asset Information Management; and a member of the Member of the Institute of Electrical and Electronics Engineers (IEEE).

**Dr. Mo Adda,** Principal Lecturer at the University of Portsmouth since 2002. He obtained a PhD in distributed systems and parallel processing from the University of Surrey. As a Senior Lecturer, He taught programming, computer architecture and networking for 10 years at the University of Richmond. From 1999-2002, He worked as a senior software engineer developing software and managing projects on simulation and modelling. He has been researching parallel and distributed systems since 1987. His research interests include multithreaded architectures, mobile networks and business process modelling, parallel and distributed processing, wireless networks and sensor networks, network security, embedded systems, simulation and modelling, mobile intelligent agent technology