Fault Tolerance and Graceful Performance Degradation in Cloud Data Center

Humphrey Emesowum*, Athanasios Paraskelidis, Mo Adda
School of Computing, University of Portsmouth, PO1 3HE, United Kingdom.

* Corresponding author. Tel: +44 023 9284 6460; email: humphrey.emesowum@port.ac.uk
Manuscript submitted October 15, 2017; accepted January 10, 2018.
doi: 10.17706/jcp.13.8.889-896

Abstract: In relation to network performance, graceful degradation amidst increase in the failure of network devices expects a reduction in performance in a way that a dramatic fall in throughput will not be noticed. To achieve a relevant graceful performance degradation especially in a cloud data center networks means the design must be fault tolerant. A fault tolerant 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 this is not the case because of the growth in the use of internet-based applications, big data, and internet of things; leading to several ongoing researches to find the best suitable design that could help alleviate the poor fault tolerance and graceful performance degradation in cloud data center. Fat trees (FT) interconnections have been the most popular topologies used in data centers due to their path diversities and good fault tolerance characteristics. In this paper, we propose a Reversed Hybrid architecture derived from the more generalized fat tree structure, Z-fat tree and compare it to a fat tree network with the same amount of resources for client server communication patterns such as HTTP and EMAIL application in a cloud data center. The results with faulty links show that our proposed Reversed Hybrid outperform the fat tree. We conclude based on the level of graceful performance degradation achieved that fault tolerance in data center cannot only be realized by adding extra hardware to the network, rather bespoke design plays a greater role.

Key words: Fault tolerance, graceful performance degradation, deadlock freedom, full bisection bandwidth.

1. Introduction

Data center is disposed to failures due to the large number of devices used for the interconnections and communications [1]. To recover from common failures with maintained adequate performance means that the design of the network must be firm [2], [3]; then as the servers, switches and links failure rates increase, such network must exhibit graceful performance degradation [4]. Since graceful performance degradation could be determined by the ability of the network to tolerate faults, then according to [5], fault tolerance is an essential and unavoidable requirement to maintain performance and reliability of network. The authors in [3], [5] suggest that for an achievable performance and reliable communication flow in a data center network, there must be a provision that will tolerate failures of devices. As a result, a manageable level of fault tolerance is attainable in an interconnection network by the creation of alternative paths between source and destination to achieve a graceful performance degradation during multiple failures.

In the quest to measure-up with the abovementioned technical challenges in data center networks, several influential network architectures were designed - Fat-tree, DCell, BCube and VLE [6], [7]. Fat tree (FT) is said to be widely used in designing data center networks [4], [8]. And over the years, Fat tree is undergoing several developmental stages due to its obvious contribution to the design of data center

network. From conventional fat tree that lacks scalability, to generalized fat tree with switches of the same radix and speed port in all network levels, to extended generalized fat tree (XGFT) that allows variable number of switch ports to be used at different level of the network [9]-[11]. However, in this paper, we based our work on an extension of fat tree called Z-node by [12]; to achieve fault tolerance and graceful performance degradation in a cloud data center.

In Section 2, we reviewed related works; and Section 3 is the design descriptions. Then in Section 4, we analyzed the simulation results; and finally, we drew conclusion based on the received packets, and ascertain the level of fault tolerance and graceful performance degradation of each design.

2. Related Works

Generally, across several network topologies of data center, there have been some contributions to work-around the challenges of faults tolerance and graceful performance degradation. However, some of these contributions have their strengths and weaknesses. The authors of [13] proposed Network Architecture for Joint Recovery and Traffic Engineering, to help split traffic between routers over several paths on a precomputed multipath during failure; so that reliability of traffic transmission and operational costs of data center can be actualized. With the precomputed multipath, the architecture can guarantee continuous connections during failures of some links. However, its path-level failure detecting and recovering mechanism is a reactive measure and therefore cannot detect a faulty device proactively, unless after the failure has occurred. The architecture also comprises local adaptation for path failure, for re-balancing of traffic from unhealthy paths to the healthy paths upon detecting failures. The downside is that it is time consuming to transfer traffic from unhealthy paths to the healthy ones, which will eventually cause disruption in data centers.

Another interesting new trend in designing data center that worth discussing is the wavelength division multiplex links used in optical interconnection. In designing Helios architecture, the authors in [14] proposed a circuit-based data center network with two-level hybrid consisting of either optical or electrical switches as core switches for high bandwidth; and a typical packet switch as the top of the rack switches for fast all-to-all communication between switches. But according to [15], optical interconnection used in data center only provides it with high capacity, low power, and low latency; the issues of cost effectiveness, scalability and fault tolerance are still a big challenge to this architecture. Based on this assertion in [15], our proposed reversed hybrid could be having an edge because it is cost effective, scalable, and able to tolerate fault.

In recognition of the fact that fault tolerance is an unavoidable necessity for reliability and availability to be achieved in data center network; Joshi and Sivalingam [5] proposed data center virtualization based on server failure to actualize fault tolerance. In their work, they relocated virtual machines hosted on the failed servers to healthy servers to achieve a 90% of server utilization. Also, they tried to reduce the impact of server failure in data center by allocating virtual data center across the physical data center network using clustering to balance the network load. Notwithstanding, this work is only aimed at providing fault tolerance only on server failure. Secondly, the relocation of data from failed servers to healthy ones can introduce much delay that will eventually hinder the reliability and performance sought for in the data center. On the contrary, our proposed design, which improves fault tolerance capability in real-time, is mainly on the failure of communication links, which is the commonest failure region in data center.

In conclusion, it is obvious that our proposed reversed hybrid will improve the fault tolerance and performance of cloud data center because it complies with the assertion in [16] that Fat tree is widely used for high performance interconnection network because of its: deadlock freedom, fault-tolerance capability, and full bisection bandwidth.

3. Model Description

Fat tree notation FT(h; m_1,m_2..,m_h;w_1,w_2..,w_h) has been described in several literatures e.g. in [9], [17]. Nevertheless, as explained in our previous work [18], in constructing our proposed fat tree variant; by default, we used full connectivity to connect the servers at level0 to level1 switches for each subtree/zone, and the numbering of switches and its ports at every zone and level are from left to right starting from zero. We introduced the pattern used by the authors of Z-Fat tree [12], which is defined by the number of root nodes per zone in its semantics and adds a degree of connectivity 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.

For the Z-Fat tree, Fig. 1 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=4 indicating that there are extra connections at level 2. For the Reversed Hybrid Fat tree: Fig. 2 H_2^-(2;6,4;2,8;1,1), the topology is divided into two parts- left and right. 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.

3.1. Single FT (Ƶ) and Reversed Hybrid FT (H_2^-)

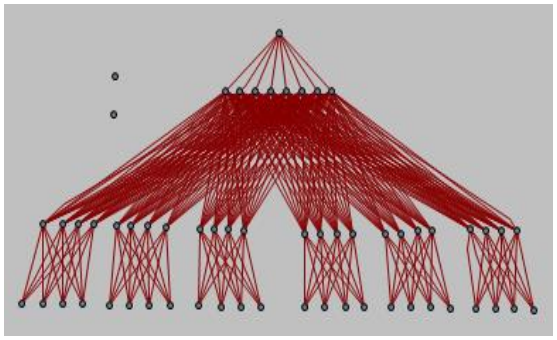


Fig. 1. Ƶ (2;4,6;4,8;1,4).

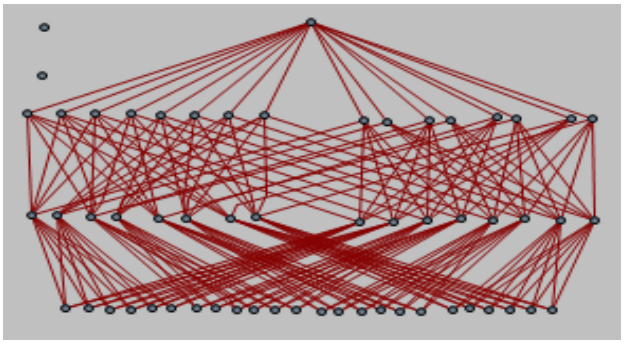


Fig. 2. H_2^-(2;6,4;2,8;1,1).

3.2. Switch Level Relationship

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

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.

3.3. Switch Connectivity

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

where k represents ϵ {0, 1, …, $R_{n+1}/gcd(R_n, R_{n+1})$-1}; (Down-top connection) [12]

$$X_n = (R_n\,((x_{n+1} \backslash R_{n+1})\ \backslash Z_n) + (x_{n+1} \% R_{n+1}) * R_n/gcd_{(Rn+1,\ Rn)} + k)\ \%\ R_{n.} \tag{2b}$$

where k represents ϵ {0, 1, …, $R_n/gcd(R_{n+1}, R_n)$-1} (Top-down connection) [12]

X_{n+1} *is* switch sought after at the upper level 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 level 2 to level 1 switches for top-down connection H_2^- (Fig. 2), using (2b)

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

where $k \in \{0, 1, \ldots, R_n/gcd(R_{n+1}, R_n)-1\}$. So, $k = 0, 1, 2, 3$. Therefore, If $k= 0$, $(0+ k(0))\ \%8 = 0\%8 = 0$; If $k= 1$, $(0+ k(1))\ \%8 =1\%8 = 1$; If $k= 2$, $(0+ k(2))\ \%8 = 2\%8 = 2$; and 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 $k \in \{0, 1, \ldots, R_n/gcd(R_n, R_{n+1})-1\}$. So, $k = 0$. Therefore, If $k= 0$, it implies $(0+ k(0))\ \%2 = 0$. Therefore, switch 0 at level 2 will connect to: 0, 1, 2, 3 level 1 switches at right-hand-side; and switch 0 at the left-hand-side.

3.4. Port Mapping

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

where p, set of switch ports to be mapped, *represents* $\epsilon \{0, 1, \ldots, R_n/gcd(R_n, R_{n+1})-1$; X_{p+1} represents switch ports to be mapped at upper level.

In Fig. 1, at level 1, there are 6 zones for z_2 within zone z_3, with r_1=4 in each. It implies that each level 2 switch has 24 down-port to be mapped. For example, we mapped level 1 switch 0 in the first zone of Z_2 to level 2 switches thus:

$X_{p+1} = ((0\backslash 4)\ \%6) * 4/4+ p; = (0\ \%6) * 1+ p = 0+p$; and $p \in \{0, 1, \ldots, R_n /gcd(R_n, R_{n+1})-1)$; p=0 and $X_{p+1} = 0$. Hence, level 1 switch 0 will be mapped to ports 0 of level 2 switches where it is being connected to.

3.5. IP Address Translation

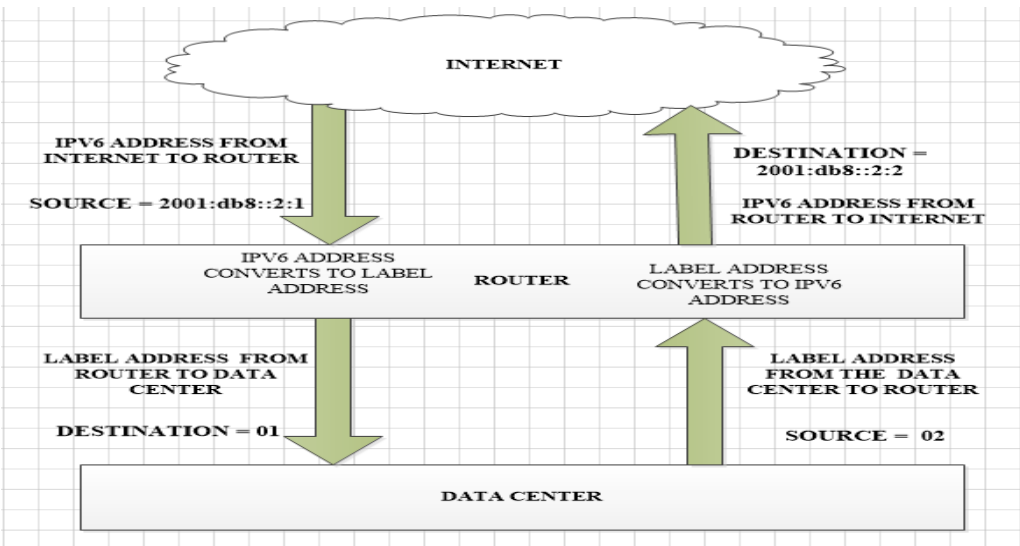


Fig. 3. Mapping internet IP address to data center labels. [18].

Fig. 3 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. For detailed explanation, be referred to our previous work in [18].

4. Analysis of Simulation Results

4.1. Table 1: Summary of Network Inventory

Topologies	Switches	Clients	Servers	Number of Links	10% of Failed Links	20% of Failed Links	30% of Failed links	40% of Failed Links
Reversed Hybrid $H_2^-(2;6,4;2,8;1,1)$	32	1	24	192	19	38	58	77
Single Fat-tree Ƶ(2;4,6;4,8;1,4)	32	1	24	296	30	59	89	118

Table 1 is the Summary of Network Inventory for the Simulation of Single Fat-tree (Ƶ) and Reversed Hybrid (H_2^-) carried out on Riverbed. Equal number of switches and servers were used in both topologies. In each design, we used a single workstation where the profile definition was deployed, used to model the behavior of a user, and acts as source traffic. Here, the workstation represents the users over the internet retrieving information from the servers(cloud).

4.2. Email Results

The simulation for Single FT (Ƶ) and Reversed Hybrid FT (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.

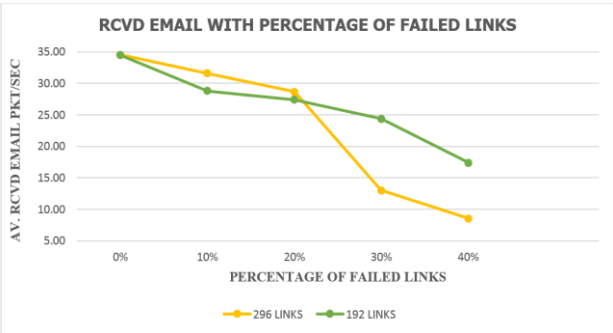


Fig. 4. Percentage of failed links.

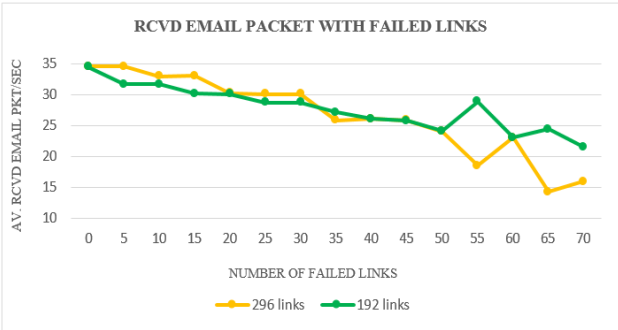


Fig. 5. Equal number of failed links.

The results of Fig. 4 for the received email packet, show a very big margin between our Reversed Hybrid of 192links and the Single FT of 296 links. The link failure is considered based on the percentage of the number of links each topology has as shown in Table 1. For example, when 10% of links are failed across both topologies, the number of links failed for the FT (Ƶ) is 30, while that of FT (H_2^-) is 19. However, at the same 0% of failed links, both topologies had average received packets/second of 34.58; but as the number of percentage of failed links increases, our proposed Reversed Hybrid FT (H_2^-) showed that it is far better than the Single FT (Ƶ). In Fig. 5, we confirmed our result by failing same number of links across the two topologies. In this scenario, we ran several simulations by failing links in multiple of 5 till 70. At 70 failed links, the throughputs received are 21.5 pkt/sec and 15.95pkt/sec for our reversed hybrid FT (H_2^-) and Single FT (Ƶ) respectively.

4.3. HTTP Results

The simulation for Single FT (Ƶ) and Reversed Hybrid FT (H_2^-) were run at simulation time of 900 seconds with packet size of 500,000 bytes. At constant Frame Inter-arrival times of 4.0 seconds.

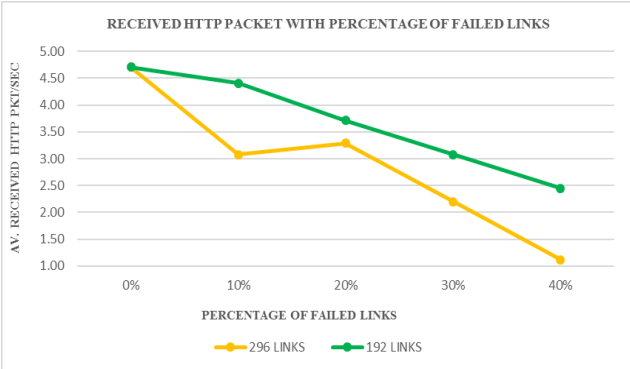


Fig. 6. Received percentage of failed links.

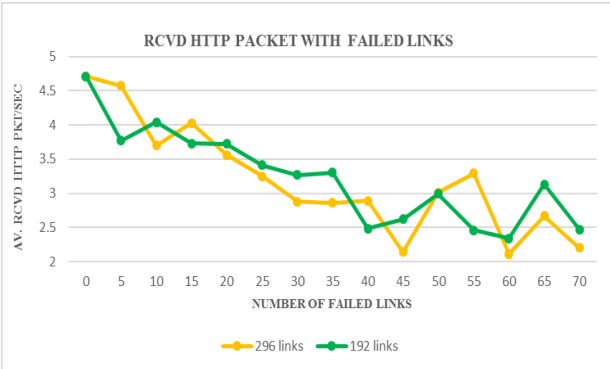


Fig. 7. Equal number of failed links.

The results of Fig. 6 for the received http packet, also show big margin between our Reversed Hybrid of 192links and the Single FT of 296links. As at same 0% of failed links, both topologies performed equally; but as the number of percentage of failed links increases, our proposed Reversed Hybrid FT (H_2^-) showed a better fault tolerance and graceful degradation than Single FT (Ƶ). In Fig. 7, we also confirmed our result by failing same number of links across the two topologies as done in Fig. 5. Our Reversed Hybrid FT (H_2^-) also proved to be better than the Single FT (Ƶ) as more links are failed.

5. Conclusion

Our results for the graceful performance degradation and fault tolerance was carried out using two different applications on different links failures parameters to ascertain that our proposed Reversed Hybrid FT(H_2^-) is far better than the Single FT(Ƶ). The uniqueness of our Reversed Hybrid FT (H_2^-) design is that the right-hand-side has a replica of alternative paths for upward and downward traffic forwarding to and from the server; at the left-hand-side of the topology. In the conventional fat tree with the server to server communications, there is nearest common ancestor switch that helps forward a packet to its destination through a unique path. But with our Reversed Hybrid FT (H_2^-) there is alternate path to reach the packet destination from the nearest common ancestor switch at any level, thereby eliminating the issue of single point of failure and guaranteeing fault tolerance. Therefore, with these levels of fault tolerance and graceful performance degradation exhibited by our Reversed Hybrid, means that a robust cloud data center that can withstand growth in internet applications is achievable.

References

[1] Liu, Y., & Muppala, J. (2013). Fault-tolerance characteristics of data center network topologies using fault regions. *Proceedings of the International Conference on Dependable Systems and Networks.*

[2] Ramos, R. M., Martinello, M., & Rothenberg, C. E. (2013). Data center fault-tolerant routing and forwarding: An approach based on encoded paths. *Proceedings of 2013 Sixth Latin-American Symposium on Dependable Computing* (pp. 104-113).

[3] Gill, P., Jain, N., & Nagappan, N. (2011). Understanding network failures in data centers: Measurement, analysis, and implications. *ACM SIGCOMM Computer Communication.*

[4] Guo, C., *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).

Retrieved form http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.149.9669

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

[6] Al-Fares, M., Loukissas, A., & Vahdat, A. (2008). A scalable, commodity data center network architecture. *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication — SIGCOMM '08.*

[7] Bradonji, M., Labs, B., & Hill, M. (2014). *Scaling of Capacity and Reliability in Data Center Networks Categories and Subject Descriptors, 2,* 3-5.

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

[9] Liu, Y., Muppala, J. K., & Veeraraghavan, M. (2014). *A Survey of Data Center Network Architectures.*

[10] Al-Fares, M., Loukissas, A., & Vahdat, A. (2008). A scalable, commodity data center network architecture. *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication. ACM* (pp. 63-74).

[11] Niranjan, R. M., *et al.* (2009). Portland: A scalable fault-tolerant layer 2 data center network fabric. *ACM SIGCOMM Computer Communication Review,* 3*9(4)*, 39-50.

[12] Adda, M., & Peratikou, A. (2017). Routing and fault tolerance in Z-fat tree. *IEEE Transactions on Parallel and Distributed Systems, 99.*

[13] Suchara, M., Park, F., Xu, D., & Rexford, J. (2011). Network architecture for joint failure recovery and traffic engineering categories and subject descriptors. *ACM SIGMETRICS*, 97-108.

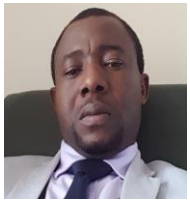
[14] Farrington, N., *et al.* (2010). Helios: A hybrid electrical/opticalswitch architecture for modular data centers. *Proceedings of ACM SIGCOMM '10* (pp. 50–339).

[15] For, E. (2013). *Optical Interconnection Networks in Data Centers: Recent Trends and Future Challenges*.

[16] Bogdanski, B. (2014). *Optimized Routing for Fat-Tree Topologies.* Ph.D thesis, Department of Informatics Faculty of Mathematics and Natural Sciences University of Oslo.

[17] Sem-Jacobsen, F. O., Skeie, T., Lynse, O., & José, D. (2011). Dynamic fault tolerance in fat trees. *IEEE Transactions on Computers, 60(4),* 508-525.

[18] Emesowum, H., Paraskelidis, A., & Adda, M. (2017). Fault tolerance improvement for fat-tree based cloud data centers. *Journal of Communications, 12(7),* 412-418.



Humphrey Emesowum received the Ph.D 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 B.Sc 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) and a student member of IEEE.

Athanasios Paraskelidis is 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 Ph.D in wireless network segregation utilizing 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).

Mo Adda is a principal lecturer at the University of Portsmouth since 2002. He obtained a Ph.D 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.