Improving the Efficiency of Server-centric Data Center Network Architectures

Ting Wang, Yu Xia, Dong Lin*, Mounir Hamdi
Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong
{twangah, rainsia, hamdi}@cse.ust.hk, dr.lindong@gmail.com*

***Abstract*—Data center network architecture is regarded as one of the most important determinants of network performance. As the most typical representatives of architecture design, the server-centric scheme stands out due to its good performance in various aspects. However, there still exist some critical shortcomings in these server-centric architectures. In order to provide an efficient solution to these shortcomings and improve the efficiency of server-centric architectures, in this paper, we propose a hardware based approach, named "Forwarding Unit". Furthermore, we put forward a traffic aware routing scheme for FlatNet to further evaluate the feasibility and efficiency of our approach. Both theoretical analysis and simulation experiments are conducted to measure its overall performance with respect to cost-effectiveness, fault-tolerance, system latency, packet loss ratio, aggregate bottleneck throughput, and average path length.**

***Keywords*—Data center network, Flat network, Server-centric**

I. Introduction

As a centralized repository clustering a massive number of servers, data centers are the home to essential large-scale computation, storage and Internet-based applications which provide various services, like searching, social networking, e-mails, gaming, cloud computing, and so on. The architecture of a data center network (DCN) plays a significant role in meeting the requirements of these services as well as the agility and dynamic reconfigurability of the infrastructure for changing application demands. With data availability and security at stake, the role of the data center is more critical than ever. As a result, many novel DCN architecture proposals are proposed, the most typical representatives of which are the server-centric architectures, such as DCell [4], BCube [3], FiConn [5], HyperBCube [7], and FlatNet [6].

In server-centric architectures, the servers use multiple NIC ports to get involved in the network infrastructure and take a part in forwarding packets. The server-centric network design stands out owing to its many superior features. Firstly, level-based and recursively defined architectures are more likely to achieve good scalability and more easily design a large-scale DCN (e.g. DCell scales double exponentially and FlatNet scales at O(n^3) with only two layers of network using n-port switches). Secondly, it achieves better performance in aggregate throughput and average packet delay for small sized networks [2]. Thirdly, the server-centric architectures are more cost-effective than other candidates (e.g. Fat Tree [1]) [9]. Fourthly, it has good fault-tolerance because of numerous node-disjointed paths [3][4][8]. Last but not least, it only uses low-end commodity switches without any modifications.

Although the server-centric scheme receives numerous highlights, there still exist many practical issues which are enumerated in Section III. In response to these issues, this paper aims to provide an efficient approach which can solve these shortcomings and improve the efficiency of server-centric architectures. The primary contributions of this paper can be summarized as follows:

1) Address the critical shortcomings which exist and need to be solved in server-centric data center networks.
2) Propose an efficient approach to achieve a logically flat DCN and solve the issues of server-centric architectures.
3) Put forward a traffic-aware adaptive routing algorithm.
4) Conduct extensive simulations to evaluate the feasibility and performance of the approach.

The rest of the paper is organized as follows. First we review the related research literature in Section II followed by Section III which addresses the existing drawbacks of server-centric architectures. Section IV demonstrates the approach to overcome these drawbacks and improve the server-centric architectures. Afterwards, Section V implements a fault-tolerant traffic-aware routing scheme based on FlatNet for further evaluating the proposed approach in Section VI. Finally, the paper is concluded in Section VII.

II. Related Work

Numerous server-centric architectures have been proposed, including DCell [4], BCube [3], FiConn [5], FlatNet [6], and so on. These architectures are usually level-based and recursively defined, and the servers should not only carry out computations, but also undertake forwarding tasks.

In DCell, the most basic element named DCell_0 consists of n servers and one n-port switch. Each server in a DCell_0 is connected to the switch in the same DCell_0. Generally, DCell_k is created by $t_{k-1}+1$ DCell_{k-1}s, where t_{k-1} is the number of servers in each DCell_{k-1}. Each server in a DCell_k has $k+1$ links, and the level-i link connects to a different DCell_{i-1} within the same DCell_i. In this way, DCell scales double exponentially.

BCube is specially designed for shipping container based modular data centers. Its most basic element BCube_0 is the same as DCell_0: n servers connect to one n-port switch. While constructing a BCube_1, n extra switches are used, connecting

978-1-4799-2003-7/14/$31.00 ©2014 IEEE

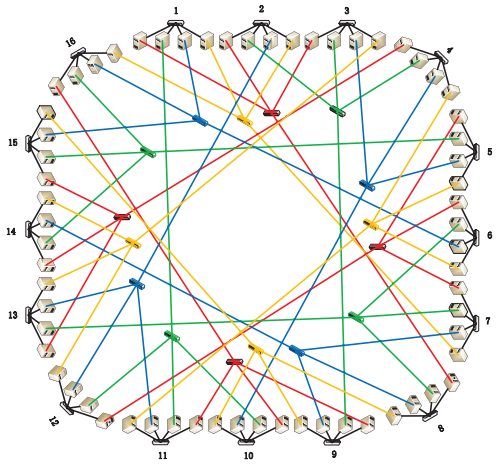

Fig. 1. A 64-server FlatNet architecture.

to exactly one server in each BCube_0. More generally, a BCube_k is derived from n BCube_{k-1} and n^k n-port COTS mini-switches. Different from DCell, the servers in BCube only connect to switches without connecting to other servers.

FiConn shares a similar design principle to DCell. The main difference is that the degree of each server in FiConn is always 2 while in DCell it is $k+1$. Likewise, a high-level FiConn is derived from a number of low-level FiConns. The routing algorithm in FiConn is traffic-aware which can help it utilize the link capacities referring to traffic states, resulting in a good aggregate throughput.

FlatNet is a 2-layer server-centric architecture, which scales at a speed of n^3. Briefly, the first-layer of FlatNet consists of n servers which connect to one n-port switch. And the second-layer of FlatNet is constructed by n^2 1-layer FlatNets. Different subsystems (1-layer Cells) are interconnected to each other using extra n^2 n-port switches. Fig.1 presents an example of a 64-server FlatNet architecture.

III. Issues of the Server-centric Scheme

The computational servers in server-centric architectures are not only running regular applications but also acting as routers to relay the traffic in the system. This brings numerous advantages and convenience, but it is also accompanied by many critical drawbacks:

1) The network becomes non-transparent to servers. Servers also cannot be independent of the network, and in some routing cases, server intervention is needed.

2) The Operating System kernel or the network protocol of a server has to be modified to implement the automatic fault-tolerant routing and flow-level load-balancing. Generally, the OS must be able to detect current network status (e.g., connectivity, variation of delay, etc.) and discover feasible alternative routing paths in real-time. Moreover, certain routing protocols (e.g., source routing scheme) must be implemented in order to allow a single flow transfer through multiple paths simultaneously. However, modifying the OS kernel could be very complicated and time-consuming in practice (e.g. DCell involves more than 13000 lines of C code [4]).

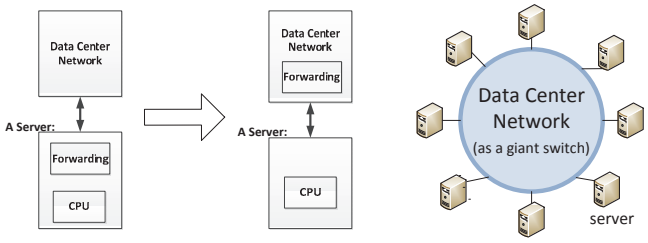

Fig. 2. Shifting forwarding task from a server to the network.

3) The server may become the bottleneck to the overall network performance and leads to additional packet delays, increased packet loss ratio and deceased aggregate bottleneck throughput, especially when suffering from insufficient software resources, for example CPU time and memory bandwidth [4], and servers cannot completely focus on computing and providing regular services.

4) A certain number of NICs are needed to be installed on each server to mete the future scaling out, which may be not very practical since most of current data center servers only have two built-in ports.

5) It is difficult to implement a Green Data Center since servers cannot be powered off even when they are idle for computing because they still undertake forwarding tasks.

All of these issues are mainly due to the servers being put onto the network side and enabling them to forward packets. Based on this careful observation, we propose a hardware based approach to address these issues in Section IV.

IV. Approach to Improve Server-centric Scheme

The forwarding tasks of servers in DCell [4] and BCube [3] are software implemented, meaning they use software for packet forwarding which relies too much on a CPU, and the CPU becomes the major bottleneck hindering the system from achieving all potential capacity gains according to their experimental results. As noted in [3] and [4], the researchers also expect to use a specially-designed network card in the server so that the server can execute the forwarding without the involvement of the CPU. To some extent this design can reduce the forwarding delay, however, it cannot make the data center network completely transparent to servers. In addition, in the cases of fault-tolerant routing, it still needs a server's intervention.

In order to totally overcome the drawbacks addressed above, we can shift the forwarding/routing tasks completely from a server to the network which makes servers independent from the network. In view of the above, we propose an efficient hardware-based approach by introducing a dedicated hardware named "Forwarding Unit". The basic idea under this approach is as illustrated in Fig.2. The forwarding unit completely isolates the server from the network and operates as a middleware. In other words, this forwarding unit is totally responsible for the forwarding tasks and the server no longer relays the traffic and thus no longer cares about the network status. After relieving the CPU from the forwarding work, the server only focuses on computing and its CPU resources can

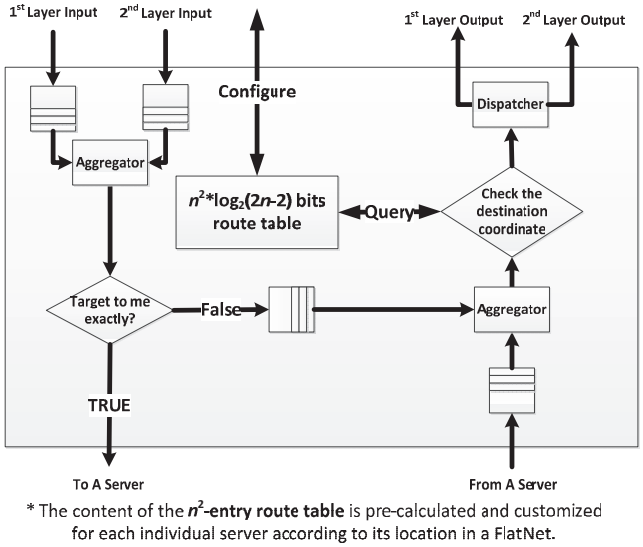

Fig. 3. The internal structure of a forwarding unit.

Subsystem-oriented routing cases: from source (S_2, S_1) to destination (D_2, D_1)
Case 1: (S_2, S_1) and (D_2, D_1) are in the same subsystem: (S_2, S_1)—①—>(D_2, D_1)
Case 2: (S_2, S_1) and (D_2, D_1) are in different subsystems, and there exists a 2-layer switch connects both subsystem S_2 and D_2: (S_2, S_1)—①—>(S_2, S_1')—②—>(D_2, S_1')—①—>(D_2, D_1)
Case 3: (S_2, S_1) and (D_2, D_1) are in different subsystems, and there is no direct connection between subsystem S_2 and D_2. An intermediate subsystem should be chosen for transition: (S_2, S_1)—②—>(S_2', S_1)—①—>(S_2', S_1')—②—>(D_2, S_1')—①—>(D_2, D_1)
Case 4: routing in a faulty environment when Case 1,2,3 fail: Choice 1: (S_2, S_1)—②—>(S_2', S_1)—Case1, 2, 3—>(D_2, D_1) Choice 2: (S_2, S_1)—①—>(S_2, S_1')—Case1, 2, 3—>(D_2, D_1) Choice 3: (S_2, S_1)—②—>(S_2', S_1)—①—>(S_2', S_1')—Case1, 2, 3—>(D_2, D_1)
*Notes: ①:the 1-layer switch; ②:the 2-layer switch

Fig. 4. The subsystem-oriented routing scheme.

be saved to achieve a better performance. Furthermore, from the users' perspective, the entire data center network can be simply regarded as a giant switch as shown in Fig.2 (right), and the architecture is then reduced to a logical single-layer data center network in a high level view.

In order to better illustrate the design of forwarding unit, without loss of generality we take FlatNet [6] as the representative of server-centric architectures. Fig.3 shows the internal structure of a forwarding unit which is specially designed for FlatNet. For a certain given node in FlatNet, there are at most n^2 possibilities of routing path (because each layer contains only n^2 subsystems), and each node has only up to $2(n-1)$ choices for one hop (because each node can reach $2(n-1)$ nodes in one hop), hence a route table the size of only $n^2 log(2n-2)$ can describe its routing behaviours. Therefore, for a FlatNet with one million servers where $n = 100$, the route table stored in the forwarding unit is just 9.3KB which implies a very small hardware cost. This provides the possibility that all the route tables can be pre-calculated and be stored in the forwarding units in advance, and then in the future be used directly at the complexity of $O(1)$. In this way, the system behaves more effectively without wasting time in calculating the route tables before determining a path and avoids the high network overhead brought by broadcasting path-probing packets. Under the circumstances of fault-tolerance or network virtualization, we can calculate the routing strategies offline, and then make appropriate corresponding configurations to the forwarding units.

Moreover, this forwarding unit based approach which shifts the control of routing/forwarding from servers to the network is also in line with the current trend of Software Defined Network. The system can then adopt OpenFlow-like technology to achieve a global configuration and management for the routing behaviours and gain a better flow scheduling in the data center network. It can also better realize the network virtualization. Besides, against the shortcomings that the routing paths in server-centric architectures are comparatively long and multiple additional forwarding actions are needed, the OpenFlow-like technology manages them globally much more easily which can further improve the system's flexibility.

Additionally, following this scheme the entire data center network can be designed as a whole, and its internal routing is simplified, for example, there is no need to consider the case of server failure. The complex cabling within the data center network can be directly implemented on the PCB or with the help of a dedicated physical port which simplifies the deployment. The system also becomes more reliable.

Concluding from the above analysis and discussions, on the one hand the forwarding unit is a hardware-based implementation approach which logically flattens the data center network and makes the internal network operate as a logical giant switch. On the other hand, it provides an efficient way to improve the performance of server-centric architectures. Additionally, it follows the core idea of Software Defined Network, but without changing the switch, and it also gives full consideration of the hardware's cost problem (more in Section VI).

V. Traffic-aware Routing Design

The original subsystem-oriented routing (SoR) scheme proposed in FlatNet [6] is a topology based routing scheme. However, this routing scheme is not adaptive and the routing decisions are made without regard to network state, where all flows between the same (*source, destination*) pairs will traverse the same path. Hence, this routing scheme eliminates the path diversity provided by the topology, which results in poor load balancing and poor link utilization. To address this issue, in this section, we present a traffic-aware routing (T-routing) taking subsystem-oriented routing as a basis.

The subsystem-oriented routing is briefly reviewed in Fig.4. The major modifications that T-routing made to SoR is that T-routing is adaptive to dynamic traffic states and always choose the outgoing link with the most available bandwidth to send the traffic when there are multiple equal paths. From another

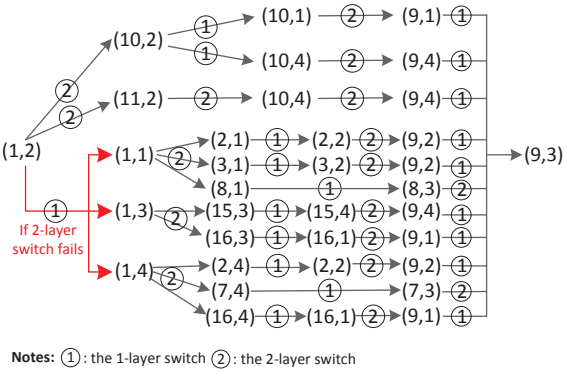

Fig. 5. Some of possible routing paths between server (1,2) and (9,3) only considering the faulty case of source server's 2-layer switch failure.

perspective, when port/link/switch faults occur, the available bandwidth of this link can be treated as zero, so the flow will always be routed by other links with maximal available bandwidth. Therefore, T-routing is also a fault tolerant routing scheme that can handle various failures.

In order to better understand the routing procedure, we use the 64-server FlatNet as shown in Figure 1 to illustrate how to route a flow from source (1,2) to destination (9,3). Here we use the same coordinate (C_2, C_1) of a server to label its forwarding unit, which indicates the C_1-th forwarding unit in the C_2-th subsystem. Originally, "(1,2)–>(9,3)" matches Case-3 which means the traffic should be first routed to an immediate subsystem which has direct connections to both subsystem 1 and 9. Then, as illustrated in Fig.5, (10,2) and (11,2) become the candidates of the next hop according to the SoR scheme as shown in Fig.4. Next, T-routing will select the route whose link has the most available bandwidth. If (10,2) is chosen here, T-routing will make an another decision to choose its next hop with the least workload. In case the 2-layer switch fails at the first step, then T-routing will route the traffic to a server (1,x) within the same subsystem as depicted in Fig.5, where "(1,2)–>(1,x)" has the most available bandwidth. Afterwards, the traffic will be routed from (1,x) to (9,3) in the same way as described above, and again in each hop the outgoing link with the most available bandwidth is always selected. As an adaptive routing scheme, T-routing makes each routing decision based on dynamic traffic states and real-time link utilization, which in return results in better link utilization and aggregate throughput.

VI. System Evaluation

In this section, extensive simulations are conducted to evaluate the performance of the forwarding unit based approach in the server-centric architecture FlatNet from various aspects under different network conditions using the T-routing scheme.

A. Network Traffic Pattern

All the simulation experiments in this section are carried out using all-to-all traffic pattern which simulates the most intensive network activities. This can evaluate the guaranteed performance of the system under the most rigorous network conditions. Besides, the uniform distribution flow mode is applied to determine the rate of generating new packets.

B. Performance Evaluation and Experimental Results

1) Average Path Length

TABLE I
THE AVERAGE PATH LENGTH FOR DIFFERENT SIZED FLATNET.

Switch Port	Number of Servers	Average Path Length
4	64	5.39
8	512	5.94
12	1728	6.14
16	4096	6.27
24	13824	6.39
48	110592	6.46

TABLE II
THE APL DISTRIBUTION AND FLOW STATISTICS

Path Distribution	Path Length	2	4	6	8
	Number of Paths	395	1424	1663	876
Flow Statistics (23472 flows in total)	12842 flows using level-0 links				
	10630 flows using level-1 links				

The path length, in this paper, is denoted by the number of links. The average path length (APL) is calculated according to $\frac{\sum PL_i * N_i}{\sum N_i}$, where N_i indicates the number of routing paths with the path length PL_i. Table I presents the experimental results of APL for different sized network in fault-free network conditions. It reveals that APL increases slightly with the increasing size of FlatNet. The APL for 64-server FlatNet is achieved as 5.39, and the peak value 6.46 appears for 110592-server FlatNet, where the diameter of FlatNet is 8. Table II gives the detailed APL distribution and flow statistics for a 64-sized FlatNet. It can be seen that the routing paths of lengths 4 and 6 are in the majority and the level-0 links carry more traffic.

2) System Latency

The network latency consists precisely of the queuing delay at each hop, transmission delay and propagation delay. In order to actually evaluate the overall packet delay of the network, we use the global packet lifetime (the time from packet's generation to the arrival at its destination) to measure the system latency. Fig.6 illustrates the performance of the average global packet lifetime (in milliseconds) in different sized FlatNet networks. In this experiment, the buffer size of forwarding unit is set to 10 MTU (1 MTU = 1500 bytes) by default. It can clearly be seen that 13%–20% reduction in the network latency is feasible when applying forwarding units, and larger sized networks achieve a larger latency reduction. This reveals that the forwarding unit based approach can contribute much to the network latency.

3) Aggregate Bottleneck Throughput

The aggregate bottleneck throughput (ABT) is used to evaluate the maximum sustainable throughput over the entire network under the all-to-all traffic pattern. For each server, among all of its flows to other servers, the flows with the

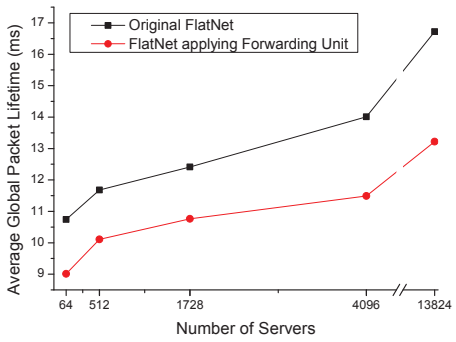

Fig. 6. The comparison of the system performance in network latency.

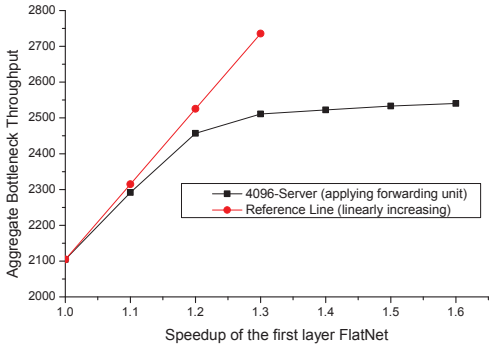

Fig. 7. The performance in ABT with different first layer speedup factors.

smallest throughput are named as the bottleneck flows. The definition of ABT is given as: *the number of flows times the throughput of the bottleneck flow* [3].

ABT is very sensitive to APL. Assuming the bandwidth of each link in a one-way communication is 1 and all links are capable of two-way communication, then the overall network capacity C_{nw} is $2 * N_{links}$, where N_{links} denotes the total number of physical links (there are actually $2*N_{links}$ virtual communication links). If we define the proportion of network capacity that ABT can reach as $P_{ABT} = \frac{ABT}{C_{nw}}$, then we can derive that $P_{ABT} = \frac{1}{APL}$. A simple proof is given as:

$$NF_{link} = \frac{N_{flows} * APL}{2 * N_{links}} \quad (1)$$

$$ABT = N_{flows} * \frac{1}{NF_{link}} \quad (2)$$

where N_{flows} is the total number of flows in the network and NF_{link} indicates the number of flows carried in one link. Combining Equation (1) and (2), there is

$$ABT = \frac{2 * N_{links}}{APL} \quad (3)$$

$$P_{ABT} = \frac{ABT}{C_{nw}} = \frac{1}{APL} \quad (4)$$

which concludes the proof.

TABLE III
THE PERFORMANCE OF ABT IN A 1000-SERVER FLATNET

1000-server FlatNet, APL = 6.063		
	Original FlatNet	FlatNet applying Forwarding Unit
ABT	484	576
P_{ABT}	$\frac{484}{2*2000} = 12.1\%$	$\frac{576}{2*2000} = 14.4\%$
$\frac{1}{APL}$	16.5%	16.5%

The evaluation of ABT is conducted on a 1000-server FlatNet using 10-port switches, which has 2000 physical links. Table III gives the simulation results of ABT and APL. Given the APL = 6.063, the FlatNet applying forwarding unit achieves ABT of 576, which reaches as high as 14.4% of the overall network capacity and is actually very close to the theoretical limitation (1/6.063 = 16.5%). Clearly, the FlatNet applying forwarding unit achieves a much better performance in ABT than the original server-centric FlatNet.

4) Network Speedup

As mentioned above, in FlatNet the traffic burden carried by different layers differs. According to the experimental results shown in Table II, the level-0 links undertake higher network traffic loads resulting in a higher link utilization rate. This careful observation implies that the first layer is more likely to become the bottleneck and will negatively affect the aggregate bottleneck throughput. Fig.7 exhibits the improvements on the aggregate bottleneck throughput with different first layer speedup factors for a 4096-server sized FlatNet. Based on our findings derived from simulations, the most beneficial speedup factor to the first layer network is suggested as 1.2, which leads to a more cost-effective network.

TABLE IV
HARDWARE COST AND SYSTEM PERFORMANCE

Switch Port-Count	Forwarding Units	Route Table Size (Bytes)	Buffer Size (MTU)	Avg. Packet Loss Ratio
4	64	5.17	10	0.0212%
			20	0.0204%
			30	0.0190%
8	512	30.46	10	0.0250%
			20	0.0239%
			30	0.0198%
12	1728	80.27	10	0.0253%
			20	0.0241%
			30	0.0203%
16	4096	157.02	10	0.0257%
			20	0.0244%
			30	0.0207%
24	13824	397.70	10	0.0260%
			20	0.0249%
			30	0.0212%

5) Hardware Cost & Average Packet Loss Ratio

Table IV presents the cost of hardware implementation and system's performance in the average packet loss ratio. As illustrated in the table, the hardware cost is very low, for instance, for a 13824-server data center the size of the route table stored in each forwarding unit is only 397.7 bytes. The system also demonstrates a favourable performance in controlling traffic congestion and packet loss. The average packet loss ratio is kept at a very low level of no more than 0.026%, and there is a certain decrease when the buffer size increases.

6) Fault Tolerance

The fault tolerance mainly depends on the network intercon-

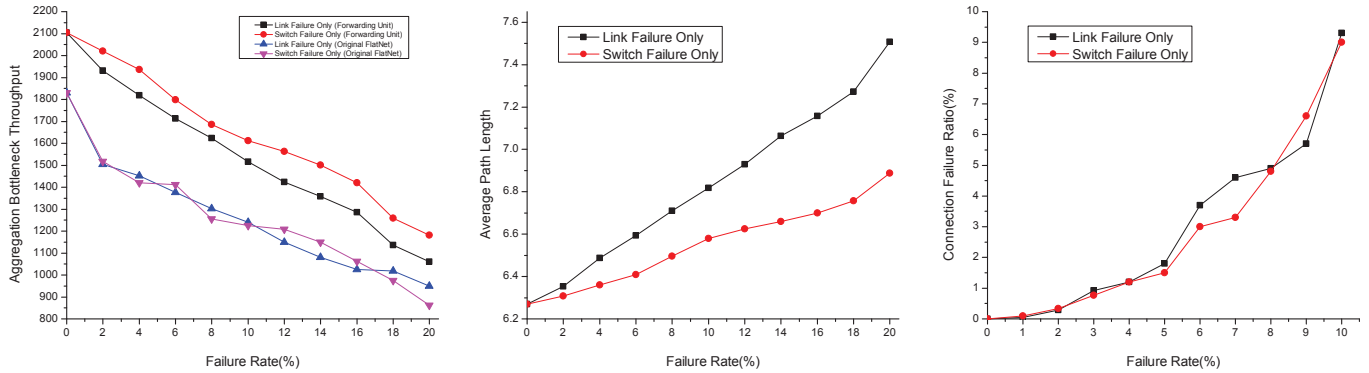

Fig. 8. The performance of ABT, fault tolerance and APL under faulty conditions in a 4096-server FlatNet running with forwarding units.

nection architecture and the specially designed fault tolerant routing schemes. The network devices (e.g. switch, router) and links that occur failures greatly degrades the network performance (such as ABT, APL, routing connection success ratio). In the server-centric architectures, the server failures will also greatly affect the network performance since servers are also considered as routers in the system. However, the forwarding unit based approach which isolates the servers from the network has absolutely no longer needs to consider the case of server failures any more when designing the fault tolerant routing scheme.

Fig.8 exhibits the performance of ABT, APL and connection failure ratios in the case of link failures and switch failures in a 4096-server FlatNet applying forwarding units. Although the system yields a gradually decayed network performance as the failure rate increases, the measured results still largely outweigh those of the original FlatNet. For example, the ABT in the case of 20% switch failure rate is achieved as 1181.3, while the original FlatNet only obtained 862.3 ABT for the same case, which indicates a much better improvement in ABT after introducing forwarding units. The improvement on APL is not obvious, yet it can be kept within 8 under faulty network conditions. Moreover, the results shown in Figure 8 also reveal that the system holds an upper bound of connection failure ratio, that is, if the link/switch failure rate is f then the connection failure ratio of the whole system will also not exceed f. All of these results reflect a good performance in fault tolerance.

VII. Conclusion

In this paper, based on careful investigations we addressed some critical shortcomings existing in the server-centric data center architectures caused by a dependency on servers to forward packets. In order to efficiently solve these drawbacks, we proposed a hardware based approach which shifts forwarding tasks from servers to the network aiming to achieve a logically flat data center network. The forwarding unit totally isolates the servers from the network, and makes the data center network completely transparent to servers which brings abundant advantages. In order to evaluate the feasibility and good performance of this approach, we presented a traffic-aware routing scheme based on FlatNet. Extensive simulations have been conducted to evaluate the overall performance of the system, including average path length, system latency, aggregate bottleneck throughput, hardware cost, packet loss ratio and fault tolerance. The experimental results of the efficiency and feasibility of the forwarding unit based approach are convincing. Moreover, based on some careful observations we suggested a proper speedup factor to the first layer network to achieve a more cost-effective network.

Acknowledgment

This research has been supported by a grant from the Research Grants Council (RGC) with project number 612912 and Huawei Technologies Co. Ltd with project number HUAW10B15Z002.09/10PN. Besides, the authors would like to express their thanks and gratitudes to the anonymous reviewers for their valuable feedback and advice.

References

[1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 63–74. ACM, 2008.

[2] Kashif Bilal, Samee U Khan, Limin Zhang, Hongxiang Li, Khizar Hayat, Sajjad A Madani, Nasro Min-Allah, Lizhe Wang, Dan Chen, Majid Iqbal, et al. Quantitative comparisons of the state-of-the-art data center architectures. *Concurrency and Computation: Practice and Experience*, 2012.

[3] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. Bcube: a high performance, server-centric network architecture for modular data centers. *ACM SIGCOMM Computer Communication Review*, 39(4):63–74, 2009.

[4] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. Dcell: a scalable and fault-tolerant network structure for data centers. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 75–86. ACM, 2008.

[5] Dan Li, Chuanxiong Guo, Haitao Wu, Kun Tan, Yongguang Zhang, and Songwu Lu. Ficonn: Using backup port for server interconnection in data centers. In *INFOCOM 2009, IEEE*, pages 2276–2285. IEEE, 2009.

[6] Dong Lin, Yang Liu, Mounir Hamdi, and Jogesh Muppala. Flatnet: Towards a flatter data center network. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 2499–2504. IEEE, 2012.

[7] Dong Lin, Yang Liu, Mounir Hamdi, and Jogesh Muppala. Hyper-bcube: A scalable data center network. In *Communications (ICC), 2012 IEEE International Conference on*, pages 2918–2923. IEEE, 2012.

[8] Yang Liu, Jogesh Muppla, Malathi Veeraraghavan, Dong Lin, and Mounir Hamdi. Data center networks. http://www.amazon.ca/Data-Center-Networks-Yang-Liu/dp/3319019481, 2013.

[9] Lucian Popa, Sylvia Ratnasamy, Gianluca Iannaccone, Arvind Krishnamurthy, and Ion Stoica. A cost comparison of datacenter network architectures. In *Proceedings of the 6th International COnference*, page 16. ACM, 2010.