Report on Data Centers

Yongsen MA

I. Literature Review

According to the architecture and topology in these papers, Data Center Networks can be divided into static-electrical and flexible data centers. The static-electrical data centers can be classified into multi-layer tree (Fat-tree [1], VL2 [2]), server-centric (DCell [3], BCube [4]) and random graph (Jellyfish [5]). The flexible data centers have optical (c-Through [9], Helios [10], OSA [11]) and wireless (Wireless flyways [12], 3D beamforming [13], Cylindrical racks [14]) solutions. These papers mainly discussed the research issues including:

- Topology [1]–[5]
- Addressing and Routing [1]–[5], [16]
- Mapping [2], [15]
- Virtualization [2], [6], [7], [9], [16]
- Scheduling [1], [8], [9], [12]
- Failure Handling [15]–[17]
- Multi-tenant Resource Allocation [6]–[8]

A. *Static-electrical Data Centers*

The traditional data centers are generally designed in static connection by electrical switches. For the issues of topology, routing and scheduling in these data centers, there are numerous researches in this field including Fat-tree, VL2, DCell, BCube, Jellyfish, etc. These work can be classified into multi-layer tree, server-centric and random graph according to the differences in topology graph, which also brings different advantages and disadvantages.

1) Multi-layer Tree: The topology of fat-tree is multi-rooted tree which can provide full aggregate bandwidth with low cost and resolved the bottleneck in inter-node communication. But the scale of fat-tree is limited that 48-port switches can only support $48 \cdot \frac{48}{2} \cdot \frac{48}{2} = 27648$ hosts at most.

The conventional data center architecture has limited server-to-server capacity that the over-subscription ratio increases rapidly in upper layer switches. Also, servers in multi-layer tree are location related that nearby servers are responsible for demand peaks or failures resulting in resources wasting. According to the measurement and analysis of traffic patterns, VL2 adopts Virtual Layer 2 Networking to realize the dynamic resource allocation across large server pools. It can assign any server to any service which can address the fluctuating demands of individual services.

The multi-rooted tree topology has poor fault-tolerant features which make it hard to detect and maintain failures. Furthermore, multi-rooted tree can not support one-to-all and all-to-all communications effectively.

2) Server-centric: For server-centric data centers, all-to-all communication can be realized and the network can be incrementally expanded. DCell is a recursively defined structure which scales doubly exponentially as the node degree increases (ideal for large scale data centers). Furthermore, DCell is fault tolerant and can provide higher network capacity. But the bottleneck link locates in the low-level links and traffics are not balanced. Moreover, increment expansion **requires additional ports** which brings extra overhead and complexity.

BCube provides a similar server-centric solution for modular data centers, which has no performance bottlenecks and provides much higher network capacity. However, to connect certain number of servers, it needs more switches (k times) and wiring is more complex which will **increase the cabling cost**. For BCube, the aggregate bottleneck throughput decreases gracefully under both server and switch failures, which means poor performance when incomplete units exist.

In the multi-path routing and speedup forwarding of one-to-all communications, it needs **strict time synchronization**, and more importantly, brings the trade-off of reliability and capacity just as MIMO in wireless networks.

3) Random Graph: The architectures listed above have rigid structure that interferes with incremental expansion. Jellyfish adopts random graph topology to address this problem, and also provides great flexibility in oversubscription design. But it brings **challenging complexity** to routing, wiring and maintaining.

B. Flexible Data Centers

To improve the efficiency of performance/cost, it requires data centers be flexible which can respond to dynamic traffic loads. Currently, optical and wireless solutions are proposed to address this problem, and both methods can add flexibility to conventional data centers. Optical switching can provide higher bandwidth but suffers higher switching latency. Wireless flyway is easy to deploy but the physical layer interference and link layer protocol are challenging and need to be standardized.

1) Optical: Optical circuit switching technology has large bandwidth advantage over conventional packet switching, which provides suitable solution for data centers. But optical networking has **slow switching** that will increase the reconfiguration time, which remains as the main challenge for optical data centers. To take full advantage of high capacity of optical switching while ensure low latency, c-Through proposed a hybrid packet and circuit switched data center network architecture. Along with the hybrid architecture, it also makes demand estimation and traffic demultiplexing in end hosts. But it has high **configuration complexity** in separating the networks into packet/circuit-switch and the preconnection socket buffer. The socket buffers at end hosts also **increase the overhead** of traffic measurement and software programming, and maybe increase the **queue latency**. c-Through requires each host run a management daemon which further increases the system overhead. Finally, the authors use an Ethernet switch to emulate the optical switch and artificially restrict the communication and configuration, it will reduce the accuracy of evaluating the influence of optical's high latency.

Unlike c-Through, Helios implements traffic estimation and demultiplexing on switches and requires no modifications to end hosts. The topology of Helios is multi-rooted tree in which core switches can be either electrical packet switches or optical circuit switches. The optical parts **add flexibility** to data centers which can get optimal trade-off of cost, power consumption, complexity and performance for a given set of workloads. In the evaluation of Helios, the authors disable the function of debouncing and Electronic Dispersion Compensation to improve PHY throughput. But this will perhaps reduce the upper layer **Goodput** due to link insertion/removal and signal noise.

The hybrid structure in OSA is an abstract model that captures key aspects of both c-Through and Helios. In addition, OSA can connect one ToR to multiple ToRs simultaneously and multi-hop connection exists between any pair of remote ToRs. Compared to hybrid structures including c-Through and Helios, OSA is more flexible and can provide higher bisection bandwidth.

2) Wireless: Another solution for flexible data centers is 60GHz wireless links, but wireless data center is a new concept which should be explored further. Many issues such as interference, secluding, security, etc. should be **standardized**. On the other hand, the issues on cost, performance, energy-efficiency, reliability, etc. should be taken into consideration compared with electrical and optical data centers.

Apart from the physical topology, the conflict graph should be measured when wireless links are introduced. For N racks and K antenna orientations, the input table of conflict graph is very **large** with the size of $(NK)^2$. On the other hand, the propagation conditions are similar, which means the conflict table is **sparse**. In addition to wireless propagation, upper layer link state should also be designed specifically for data center networks. Apart from the concurrent links characterized in [13], the **efficient throughput** should be explored. For instance, although the concurrent links are more with larger ceiling height h, it can decrease the throughput according to the curves of RSS (or Data Rate) vs. distance.

C. Other issues

1) Multi-tenant: When data centers are shared in the cloud, i.e., multi-tenant data centers, the resource allocation is challenging due to the variability of tenants, application types and traffic patterns. In addition to the optimization within data center networking, the cloud providers should also conduct scheduling between different tenants.

Oktopus [7] adopts virtual network abstractions to make prediction and allocation. NetLord [6] also provides tenants with flexible network abstractions by fully and efficiently virtualization of he address at both layer 2 and 3. It requires only static configuration and local information, but it will increase the **reconfiguration complexity** when there is topology change which will occur frequently in flexible data centers. FairCloud [8] exposes the fundamental trade-offs in multi-tenant data centers and develops a set of resource allocation policies to best navigate these trade-offs. It allocates the bandwidth according to the number of VMs rather than the number of flows, sources, or source destination pairs. But the proportional allocation of link sharing is difficult in practical implementation.

2) Failure Handling: For large scale data centers, there is a fundamental problem of failure handling including detection, diagnosis and repair. NetPilot [17] presents an automated system to quickly mitigate failures before the diagnose and repair process. It takes advantage of **multi-level redundancy** to mitigate most types of failures by simple actions such as deactivation or restart.

DAC [15] presents fast device-to-logical mapping with low time complexity by leveraging the properties of topology graph, and it can also help to efficiently detect errors such as link failures and miswirings. Since DAC is generally designed for different data center structures, it can be further developed for certain topologies. For instance, the autoconfiguration of DCell and BCube can be faster due to the recursive topology, and similar for symmetric structures. Maybe the initial selection of vertex $\nu \in \pi_p^i$ also has influence on the efficiency for recursive or symmetric topology. On the other hand, DAC does not consider the connection differences in up and down links (directed graph), and in capacity matrix (weighted graph). The flexible data centers will also bring great challenges to DAC due to topology changes and traffic multiplexing.

II. Research Issues

A. Topology Measurement

It is challenging to measure such a large scale topology graph which usually requires all-to-all communications. It is even more hard due to the uncertain connections for flexible data centers. Since both traffic matrix [2], [12] and topology graph [15] are sparse, we can utilize Compress Sensing to reduce overhead, just as the famous example that using only 3 out of 10 words can reconstruct the whole account password. In this way, we can use small number of samples to get the topology so that all-to-all communications are not necessary.

The requirements and reasons for Compress Sensing are:

1) topology graphs are **large**, and so comes the requirement to reduce measuring and mapping overhead
2) wireless propagation and traffic demand have **dynamic** furthers, so the measurement is challenging
3) topology graphs are **sparse**, which is the prior condition for compress sensing
4) topology graphs only have **local changes** (errors, on-off or flexible switching) in real-time operating

If the adjacency matrix of a given topology graph $G = (V, E)$ is $A^{N \times N}$, it can be represented by N vertex A_i. Then the estimated matrix $y \in \mathbf{R}^n$ can be calculated as follows:

$$y = \Phi A_i \tag{1}$$

where $\Phi \in \mathbf{R}^{n \times N} (n \ll N)$ is the measure matrix. The dimension of y is reduced to only n which requires less samples. If the measured data is recovered through $\hat{A}_i = f(y) = \Psi y = \Psi \Phi A_i$, the estimated results can be obtained by optimization

$$\min \parallel A_i - f(y) \parallel_{l_x} \tag{2}$$

$$s.t. \quad \Phi A_i = y \tag{3}$$

where $l_x (x = 0, 1, 2)$ are the norms of a vector, among which l_1 is usually used representing the number of non-zero coordinates. Then y can be easily measured and A can be reconstructed through y if the measure and reconstruct matrix (Φ and Ψ) are suitably designed. Therefore, all-to-all communications are not required which can help to reduce the measurement overhead.

The topology graph can be estimated by Compress Sensing due to its large-scale and sparse feature. Furthermore, the propagation table and traffic matrix can be measured in this way according to the measurement results in VL2 [2] and wireless flyways [12].

B. Demand Response

According to the traffic measurement and analysis in VL2 [2] and wireless flyways [12], traffic patterns and loads are dynamic and unpredictable. So demand response can be done by Lyapunov optimization for it only needs the information of current and previous state. Let $Q[t]$ represent the queue backlog at the beginning of slot t, and $D[t]$ denote the size of incoming data, i.e., **demand matrix**, then the queue backlog evolving over time is

$$Q[t+1] = Q[t] - \mu[t] + D[t], \tag{4}$$

where $\mu[t]$ is the amount of data transferred during slot t, which can be modeled by the function of **transmit state, topology graph, capacity matrix**, etc. Then the queue $Q[t]$ is defined stable if

$$\overline{Q} = \limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} E\{Q[\tau]\} < \infty. \tag{5}$$

Under the stability constraint, we can design the path selection algorithm according to the measurement results of topology graph, capacity matrix and demand matrix.

C. Resource Allocation

Multi-path provides extra selection for routing, packet forwarding and capacity scheduling, which can serves as **link redundancy** [17] or **duplicately packet delivery** [4]. So there is a fundamental **trade-off between reliability and capacity** (link redundancy and duplicately packet delivery) for multi-path routing. This is more challenging (perfect matching or edge coloring) for flexible and on-demand data centers.

If packets (flows) are separately delivered through multiple paths, the physical throughput can be improved. But it needs more strict time synchronization to ensure accurate encoding results, since the length and state of concurrent paths are different. Moreover, the number of concurrent links should be optimized to improve the overall efficiency of performance/cost.

On the other hand, flexible data centers provide multiplexing mechanism for path selection. Unlike multi-path delivery, flexible multiplexing seeks high link utilization and bandwidth efficiency. The trade-off generated by concurrent links still exists, and may be more complex due to topology changes. Consideration the switching and queue latency,

III. Problems

[9] c-Through

1) It estimates traffic demands according to the per-connection socket buffer limit **at end hosts**, which brings extra overhead and increase the complexity of programming.
2) It adopts Edmonds' algorithm for perfect matching which is a central component, so why still use distributed hosts to make multiplexing management? Moreover, Edmonds is designed for single-path matching but not for **multi-path scheduling**.
3) It uses packet-switch to artificially emulate optical-switch, which will reduce the evaluation accuracy.

[10] Helios

1) Disabling debouncing and EDC can improve the PHY throughput, but the **upper layer goodput** will be lower due to link insertion/removal and signal noise.
2) The primary bottlenecks are the stages of measuring traffic matrix and changing topology, so it is important to reduce overhead of topology measurement and on-demand response.

[11] OSA

1) In the problem formulation of demand response, the sum of w_{ijk} is constrained to 1, whether it can be extended so that **multi-path (concurrent) forwarding** can be implemented.
2) The inter-rack traffic is constrained by outgoing-incoming, but it is a strict constraint which reflects the capacity efficiency? On the other hand, the traffic matrix should be included in the out-equal-in (actual transit not capacity matrix) constraint, and in **certain time slot**?
3) The link capacity should be larger than traffic demand? Moreover, the capacity should be constrained to a basic threshold and the remaining parts are satisfied by **flexible switching**?

[15] DAC

1) The initial selection of vertex v.
2) The selection of SPLD and orbit flittering can be designed for recursive topology and symmetric structures.
3) How to further examine link capacity and multiple paths.

[1] Fat tree

1) the scale is limited.
2) redundant switch elements at the leaves are the only way to tolerate failures at lower-level switch.
3) one-to-all and all-to-all communication is limited.

References

[1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. ACM SIGCOMM '08, 2008.

[2] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. VL2: a scalable and flexible data center network. ACM SIGCOMM '09, 2009.

[3] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. DCell: a scalable and fault-tolerant network structure for data centers. ACM SIGCOMM '08, 2008.

[4] 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 '09, 2009.

[5] A. Singla, C.Y. Hong, L. Popa, and P.B. Godfrey. Jellyfish: Networking data centers randomly. USENIX NSDI '12, 2012.

[6] Jayaram Mudigonda, Praveen Yalagandula, Jeff Mogul, Bryan Stiekes, and Yanick Pouffary. NetLord: a scalable multi-tenant network architecture for virtualized datacenters. ACM SIGCOMM '11, 2011.

[7] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. Towards predictable datacenter networks. ACM SIGCOMM '11, 2011.

[8] Lucian Popa, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. FairCloud: sharing the network in cloud computing. ACM SIGCOMM '12, 2012.

[9] Guohui Wang, David G. Andersen, Michael Kaminsky, Konstantina Papagiannaki, T.S. Eugene Ng, Michael Kozuch, and Michael Ryan. c-Through: part-time optics in data centers. ACM SIGCOMM '10, 2010.

[10] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. Helios: a hybrid electrical/optical switch architecture for modular data centers. ACM SIGCOMM '10, 2010.

[11] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen. OSA: An optical switching architecture for data center networks with unprecedented flexibility. USENIX NSDI '12, 2012.

[12] Daniel Halperin, Srikanth Kandula, Jitendra Padhye, Paramvir Bahl, and David Wetherall. Augmenting data center networks with multi-gigabit wireless links. ACM SIGCOMM '11, 2011.

[13] Xia Zhou, Zengbin Zhang, Yibo Zhu, Yubo Li, Saipriya Kumar, Amin Vahdat, Ben Y. Zhao, and Haitao Zheng. Mirror mirror on the ceiling: flexible wireless links for data centers. ACM SIGCOMM '12, 2012.

[14] Ji-Yong Shin, Emin Gün Sirer, Hakim Weatherspoon, and Darko Kirovski. On the feasibility of completely wireless datacenters. ACM ANCS '12, 2012.

[15] Kai Chen, Chuanxiong Guo, Haitao Wu, Jing Yuan, Zhenqian Feng, Yan Chen, Songwu Lu, and Wenfei Wu. Generic and automatic address configuration for data center networks. ACM SIGCOMM '10, 2010.

[16] Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. PortLand: a scalable fault-tolerant layer 2 data center network fabric. ACM SIGCOMM '09, 2009.

[17] Xin Wu, Daniel Turner, Chao-Chih Chen, David A. Maltz, Xiaowei Yang, Lihua Yuan, and Ming Zhang. NetPilot: automating datacenter network failure mitigation. ACM SIGCOMM '12, 2012.

[18] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010.