# CS540 - Paper Review Report # XI

Hailu Belay Kahsay - 20155624

## Title: VL2: A Scalable and Flexible Data Center Network

**Author:** Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, Sudipta Sengupta

Traditional data center topology and addressing mechanism has several drawbacks. The three major problems are limited server-to-server capacity, fragmentation of resources and poor reliability and utilization.The major cause of these drawbacks is the hierarchical topology, which brings along some problems like oversubscription and immobility. The major reason of oversubscription is that the higher the level, the more expensive the routers/switches are in that layer. Even though people use multiplex techniques like oversubscription to lower the cost as a means to overcome the problems, but the technique largely limits efficient bandwidth utilization and will bring interference between different services. Hence,VL2 a new paradigm for data center network that does only software modification by avoiding hardware changes to the existing data centers has been proposed.

The paper starts introducing VL2, a scalable data center architecture that provide uniform high capacity between servers(i.e. assign servers to a service without having to consider network topology), performance isolation between services(i.e. traffic of one service should not be affected by the traffic of any other service), Ethernet layer 2 semantics(i.e. the servers in each service should experience the network as if it were an Ethernet Local Area Network) in large data centers, and dynamic resource allocation across large server pools (aka agility).

VL2 uses a Clos network topology: each top-of-rack switch connects to two different aggregation switches, and each aggregation switch is connected to every core switch (called an “intermediate” switch).

Services use “application-specific IP addresses” (AAs), and each service has its own (virtual) IP subnet. VL2 automatically translates AAs into “location-specific IP addresses” (LAs), which describe the current location of the AA in the network topology. AAs remain unchanged as servers change network locations (e.g. due to VM migration). The translation is done by a directory service that runs on a separate set of end hosts; ARP requests made by end hosts are intercepted (by modifying the end hosts’ network stacks) to instead query the directory service. To ensure uniform performance, VL2 chooses a random path through the network for each flow. The shortest path between an end host and an intermediate switch is always three hops (top-of-rack switch, aggregate switch, intermediate switch). ECMP is used to choose among these equal-cost paths, which also makes switch failure tolerance easier.

- In the paper it is said that VLB never really causes any congestion by assignment of a large flow to a path that already has a large flow. This is because the flows never really get that big. In larger systems with a lot of large file transfer, this could cause a major problem. How to keep track of where the large flows are assigned and how to avoid assigning another large flow?
- They only consider a single application “highly utilized 1,500 node cluster in a data center that supports data mining on petabytes of data. It seems unreasonable to draw conclusions from this application only. Will the result remain if some range of data center applications are used?
- It seems that the reactive cache update proposal rely on modifying switch software. What will happen if the ToR pointed to by a stale LA crashes before the VL2 agents’ cache has expired?

**Title: Fastpass: A Centralized "Zero-Queue" Datacenter Network**

**Author:** Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Devavrat Shah, Hans Fugal

Traditional datacenter(DC) networks have been designed considering Internet's current design principles, where packet transmission and path selection decisions are distributed among the endpoints and routers.

This paper claims that an ideal datacenter(DC) network should provide several properties: 1) packets experience no queuing delays, 2) network has high throughput, 3) network should be able to support multiple resource allocation objectives between flows, applications and users. Such a network would be useful in many contexts especially in datacenters where queueing dominates end-to-end latencies, link rates are at technology's bleeding edge, and system operators have to contend with multiple users and a rich mix of workloads. DC that doesn't have these properties makes it very hard to meet complex service-level objectives and application specific goals.

Traditional networking protocols are essentially distributed protocols, packet transmission decision (congestion control) is done at the end-points whereas path selections among the network's switches(routing) are done in the middle at switches. This approach gives strong fault-tolerance and scalability, but at the cost of a loss of control over packet delays and paths taken.

In this paper, the authors propose a centralized protocol, called arbiter, to handle congestion and path selection. With a centralized arbiter, they lose the scalability and fault-tolerance of the traditional approach but gained several key features such as consistently low latency, no persistent congestion and packets will never be dropped due to buffer overflow. This paper describes the design, implementation, and evaluation of Fastpass, a system that shows how centralized arbitration of the network's usage allows endpoints to burst at wire-speed while eliminating congestion at switches. This approach also provides latency isolation: interactive, time-critical flows don't have to suffer queueing delays caused by bulk flows in other parts of the fabric. The idea we pursue is analogous to a hypothetical road traffic control system in which a central entity tells every vehicle when to depart and which path to take. Then, instead of waiting in traffic, cars can zoom by all the way to their destinations. The authors have also implemented Fastpass in the Linux kernel using high-precision timers (hrtimers) to time transmitted packets and their results show that compared to the baseline network, the throughput penalty is small but queueing delays reduce dramatically, flows share resources more fairly and converge quickly, and the software arbiter implementation scales to multiple cores and handles an aggregate data rate of 2.21 Terabits/s.

- Is Fastpass really "zero queue"? Because there is still queueing at end-hosts: *when a host wants to transmit data, it must queue those packets and send a message to the arbiter to get a slot, it must then wait for a response, and then wait for a slot*.
- Is there no queue for the arbiter?
- The authors never measure or demonstrate the impact of this extra queuing at high load (or at any load) on the end-to-end latency.
- In the paper it is said that if the primary arbiter fails, the secondary arbiter waits for the queues to flush then take over within a few milliseconds. Loss during this time?

**Title: ProjecToR: Agile Reconfigurable Data Center Interconnect**

**Author:** Monia Ghobadi, Ratul Mahajan, Amar Phanishayee, Nikhil Devanur, Janardhan Kulkarni, Gireeja Ranade, Pierre-Alexandre Blanche, Houman Rastegarfar, Madeleine Glick, Daniel Kilper

The design of traditional datacenter(DC) networks is composed of electrical packet switches interconnecting racks of servers in multi-tier cable topology. The switches are cabled together using optical fiber cable. The fundamental shortcoming of this interconnect though is it is static: there is a static capacity between top-of-rack (ToR) switches. That is we basically can't change the topology unless someone is send to the field and actually re-cabled it. Because the interconnect is static, the ideal demand matrix is to have a uniform and static between all rack pairs in a way that demand does not exceed the capacity. The designer must decide in advance how much capacity to provision

between ToR switches. Depending on the provisioned capacity, the interconnect is either expensive (e.g., with full-bisection bandwidth) or it limits application performance when demand between two ToRs exceeds capacity.

It is shown in the paper that many researchers have recognized these shortcoming and proposed reconfigurable interconnects, using technologies that are able to dynamically change capacity between pairs of ToRs. But, as per the analysis of traffic from four diverse production clusters, the authors showed that current approaches lack at least two of three desirable properties for reconfigurable interconnects: 1) Seamlessness: few limits on how much network capacity can be dynamically added between ToRs; 2) High fan-out: direct communication from a rack to many others; and 3) Agility: low reconfiguration time. Hence, they propose ProjecToR, a novel free-space optics based approach for building data center interconnects. It uses a digital micromirror device (DMD) and mirror assembly combination as a transmitter and a photodetector on top of the rack as a receiver. The approach enables all pairs of racks to establish direct links that can be reconfigured within 12 μs. To carry traffic from a source to a destination rack, transmitters and receivers in our interconnect can be dynamically linked in millions of ways.

The authors have developed topology construction and routing methods to exploit ProjecToR's flexibility, including a flow scheduling algorithm that is a constant factor approximation to the offline optimal solution. They show to effectively use such a flexible interconnect and developed an online flow scheduling algorithm that has provable guarantees. The experiment and analysis indicated that compared to the conventional folded-Clos interconnect, this approach can improve mean flow completion time by 30-95% and reduce cost by 25-40%.

- How feasible is the ProjecToR interconnect?
    - What will be the impact of environmental conditions such as vibration and dust?
- Performance of packet routing algorithm?
- While calculating the cost, they use a cost estimation based on cost breakdown of each component. What about other costs that need to be considered?