Slotted Optical Datacenter Networks with Sub-Wavelength Resource Allocation

K. Christodoulopoulos[1], K. Kontodimas[2], L. Dembeck[1], E. Varvarigos[2]
1: Nokia Bell Labs, Stuttgart, Germany
2: School of Electrical and Computer Engineering, National Technical University of Athens, Greece
konstantinos.1.christodoulopoulos@nokia-bell-labs.com

Abstract: We describe the dynamic resource allocation problem in a slotted Datacenter network and present fast scheduling solutions to solve it. We then review scalability related issues.
OCIS codes: 060.4250 Networks; 060.4251 Networks, assignment and routing algorithms; 060.1155 All-optical networks

1. Introduction

Today, the content-centric Internet and the omnipresent cloud applications are provided by hyper-scale Datacenters (DCs) located in core locations around the globe. As traffic within the DC (east-west) is much higher than incoming/outgoing traffic [1], and both are expected to continue to increase, the intra-DC network plays a crucial role to delivering the cloud services. Intra-DC traffic growth is outpacing progresses in network infrastructure, which generally follows Moore's law, threatening for a capacity crunch. State-of-the-art intra-DC networks are based on electronic switches connected in fat-tree or other variations of folded-clos topologies using fibers, with electro-opto-electrical transformation at each switch [2]. Fat-trees tend to underutilize resources, require a large number of cables and switches, suffer from poor scalability and upgradability (lack of transparency), and high energy consumption. The introduction of optical switching in intra-DC networks is a key for solving several of these shortcomings [3]-[10].

Optical switching is well established in core and metro telecom networks and is now being advocated for deployment not only between DCs, but also inside them, offering higher energy efficiency and transparency to bitrate and protocols. The introduction of optical switching in DCs has proven to be challenging due to the nature of optical switching technology and the resulting complication of controlling them. Optical switches exhibit a significant speed vs. radix trade-off. High radix (port-count) optical switches, like MEMS, offer msec reconfiguration speed, whereas nsec-speed optical switches, like PLZTs (lead zirconate titanate), are only available in small dimensions (e.g. 8×8).

Based on this tradeoff, two approaches are followed. The first is to build a hybrid network with high-radix and slow optical circuit switches (OCS), such as MEMS [3][4], and compensate for their low reconfiguration speed with a parallel fast electronic packet switched (EPS) network. Long lived elephant flows are served via the OCS while short and bursty flows over the EPS. This transfers the pressure to the control plane that, however, cannot be fast enough to identify the elephants and configure the OCS network to serve them. The second approach is to create a network of small-radix fast space switches (PLZ) and/or wavelength switches (WSS or tunable lasers & AWG) [3],[5]-[9]. Note that electronic switches also exhibit a speed vs radix tradeoff, but that is ameliorated with faster technology advances and also the ability to use buffering. In contrast, optical switches are bufferless. Building a network of bufferless optical switches (an *all-optical* network) shifts the buffering at the edges and needs a contention resolution mechanism, raising concerns regarding efficiency and the buffer sizes. The network is typically operated in slots and is synchronous. The size of the slot defined by the optical switches technology and typically set one or more orders of magnitude higher than the switches reconfiguration speed. Transmissions can follow a random access/opportunistic model with appropriate signaling when contentions occur to trigger retransmissions [9], or slots can be allocated in a predefined schedule (e.g. round robin RR [10]), or dynamically scheduled [5]-[8]. Interestingly, dynamic scheduling enables a lossless network with deterministic performance that is needed for certain, notably latency sensitive, applications [7]. In the next section we give a formal description of the dynamic resource allocation problem in a slotted DC network and fast scheduling solutions to solve it.

2. Slotted optical network operation and scheduling

We will consider the NEPHELE DC network [6] as a reference, but our analysis will be generic and applicable to several other networks [5],[7],[8]. We will assume a generic DC consisting of N racks of servers, and a Top of Rack (ToR) switch for each rack. We will assume that ToR switches are electro-optical and interconnected through a synchronous slotted network of (bufferless) optical switches, with each ToR having I ports facing that network. As discussed, such a network requires coordination to efficiently operate and avoid contention and we will focus on dynamic scheduling. This can be done at a single point (central controller) or distributedly. The tradeoff between speed vs. communication overhead of distributed controllers has not been fully studied. Most solutions focus on a single central controller [5]-[8], and we will follow this approach in the following. Fig.1a presents the studied architecture.

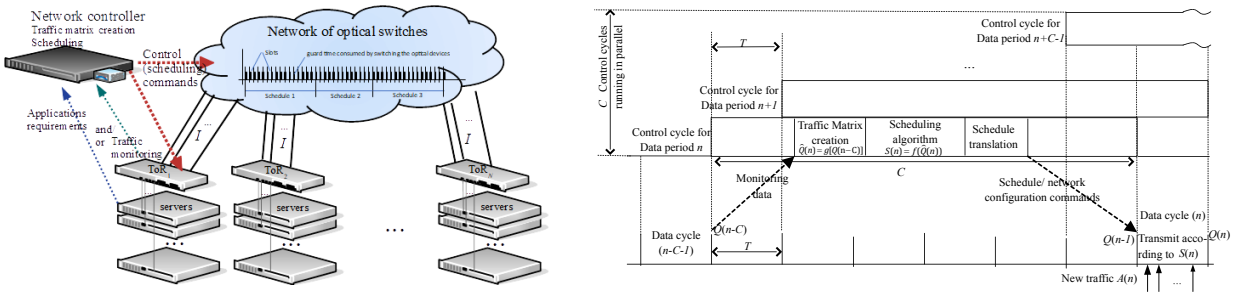

Fig. 1: a) Generic all-optical slotted network architecture, b) Data and Control cycles in a scheduled slotted all-optical network

We are considering a *closed control loop* or *cycle*. The resource allocation starts either with some feedback mechanism where the controller collects (monitors) data from the TOR queues or the applications communicate their requirements to the controller. Then the controller creates a traffic matrix and uses that to calculate the resource allocation and communicate that to the data plane/ optical switches. The Control cycle can be long and signaling is expensive and thus for an efficient DC network operation, the allocation is performed in cycles of T slots.

Under these assumptions, the network operates in *two cycles*: a) Data communication cycles of T slots (also referred to as *Data periods*), where actual data is transmitted, and b) Control cycles of duration C (in Data periods) where control information is exchanged. Fig.1b presents the Data and Control cycles. Let $\mathbf{Q}(n)$ denote the demand matrix for communication between ToRs at period n. Control cycle n corresponds to Data period n, and computes the schedule $\mathbf{S}(n)$ used during that Data period. Note, however, that the schedule is computed based on information that was available C periods earlier than the applied Data period: $\mathbf{S}(n) = f\,(g[\mathbf{Q}(n-C)])$, where g is the traffic matrix creation function that creates the *estimated demand matrix* $\widehat{\mathbf{Q}}(n)$ from $\mathbf{Q}(n-C)$, and f is the scheduling algorithm. When C>1 Data period (Control cycle duration is larger than the Data period), we can still have a new schedule each Data period: a new Control cycle starts every Data period and applies the schedule C periods later, for a total of C Control cycles running in *parallel*. The estimating function g can use predictions, filters, or any other method to improve performance. Moreover, it is possible for the scheduler to void fill the unallocated resources in $\mathbf{S}(n)$ to enable opportunistic and lossless transmissions. The scheduling time, apart from adding to the whole Control cycle, is quite important. A schedule calculated within one Data period enables pipelining: we can use a single scheduler for all C parallel Control cycles and make use of memory (previous calculations). In this case we can abstract the all-optical network as a single switch with C processing delay.

The scheduling algorithm provides the schedule $\mathbf{S}(n)$, which describes the ToR pairs that communicate in each of the I ports at each of the T slots at period n. Viewing the all-optical network as a single switch, we can take advantage of well-known scheduling solutions [11]. The scheduling algorithm takes the (estimated) demand matrix and decomposes it into a sum of permutation matrices. Any architecture constraint, such as wavelength contentions or propagation delay in the all-optical network can be accounted in the allowed permutation matrices forms (requiring small extensions to[11]). Each permutation is then translated into data-plane configurations.

Calculating the schedule can be done from *scratch* with *offline* algorithms that range from optimal [11] (e.g. using Birkhoff-Von Neumann theorem and bipartite graph matching) to heuristics [12]. The size of the problem is $O(N^2 \cdot I \cdot T)$, for N^2 communicating pairs (ToR), I ports per ToR and T slots. The schedule (output) can be written in a concise representation $O(N \cdot I \cdot T)$, defining what each ToR transmits on each port at each slot. This is also the data that needs to be transferred from the scheduler to data-plane devices. Targeting medium to large DC, optimal offline algorithms have high execution time, in the order of tens of secs in our implementations. A parallel heuristic algorithm was implemented in a FPGA exhibiting execution time in tens of μsec scale for $N \approx 1$K ToRs and medium load and density traffic $|\delta(\widehat{\mathbf{Q}})| \leq 0.25$, where $\delta(\cdot)$ denotes matrix density [13]. To further improve the speed, we can apply an incremental scheduling approach, where we start with the previous schedule and make changes over it. We need to make $\boldsymbol{D} = \widehat{\mathbf{Q}}(n-1) - \widehat{\mathbf{Q}}(n)$ changes in the schedule, and scheduling complexity is reduced to $O(\delta(|\boldsymbol{D}|))$. Such approach will be efficient if traffic is locality persistent, that is, if ToR-to-ToR traffic does not change substantially between periods [12]. Considering the parallel algorithm on the FPGA, we can expect μsec execution for medium dynamicity traffic $\delta(|\boldsymbol{D}|) \leq 0.25$. Note that the incremental approach reduces also the signaling overhead. Table 1 outlines the scheduling approaches and provides reference execution times in our implementations.

With Control cycle parallelization a new schedule is applied at each period and is lacking behind by C, the Control cycle duration. Fig.2b shows the effect of the Control cycle duration C on the execution time of 3 MapReduce jobs running simultaneously (simulated in OMNET++ using TCP/IP stuck, assuming a DC of N=32 racks, hosting 2 servers each connected to ToR with 10Gbps, I=2 optical network ports per ToR, periods of T=16 slots, a slot of 50 μsec; each MapReduce job ran for 10 iterations, with 16 map-, 8 reduce- and 8 storage-servers). We observe that the dynamic

resource allocation is slightly slower than fat-tree (even for C=0, the schedule is applied the next period and traffic waits 1 period on average) and much faster than round-robin (RR) allocation for low C. If control is extremely fast we can reduce the period or eliminate it to match fat-tree performance. The improvement over RR reduces as C increases, since the control plane falls more and more behind, and after a point it cannot follow the traffic and RR becomes better.

Scheduling algorithm and implementation platform	Traffic affecting parameters	Through put	Magnitude of exec. time (1K ToR)
Offline optimal (Matlab)	Load, Density	100%	>10 sec
Offline heuristic (Matlab)	Load, Density	90%	1 sec
Incremental heuristic (Matlab)	Dynamicity	80%	100 msec
Offline parallel heuristic (FPGA)	Load, Density	90%	100 μsec

Table 1. Implemented scheduling algorithms throughput and execution times

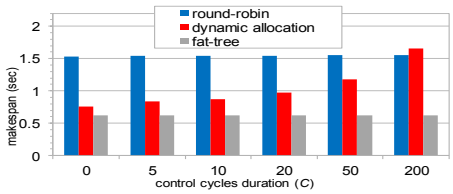

Fig 2. MapReduce makespan vs Control cycle duration (C)

3. Scalability issues and future directions

Our target is to dynamically allocate the resources to follow the arriving traffic. With Control cycle parallelization we produce a new schedule every period and the techniques discussed in Section 2 (parallel FPGA heuristic, incremental scheduling) make scheduling calculations quite fast. Still the Control cycle involves several modules (monitoring agents or application-aware interface, traffic matrix calculation, scheduler, translation of schedule) along with the modules communication and the signaling between the controller and the devices. Considering this and the effect on application performance (Fig. 2), we can conclude that efficient hyperscale DC network operation will be hard to achieve with dynamic closed loop slot scheduling.

Let us review the parameters that affect scalability. Assuming that DC applications use clusters of VMs that collaborate, the scheduled slotted optical network can easily allocate a fixed (could be full) capacity to them. The first time this allocation is performed, it will have to go through the Control cycle (this can overlap with VM setup and booting), but afterwards the schedule can be repeated periodically with incremental scheduling doing that with no overhead. It is only when we change the allocated capacity that we pay the Control cycle delay over again. Thus, it is applications dynamicity and the granularity at which we want to follow the traffic that affects performance. Allowing for certain oversubscription or relaxing the desired granularity result in much lower complexity.

Moreover, not all traffic needs to be scheduled. Typically, a DC would run different applications that are mapped into traffic classes with varying requirements. For example, latency sensitive applications, have strict requirements that can only be met through scheduling, which enables lossless communication and deterministic latency and even jitter [7][15]. Serving all traffic through an open-loop control [10] or opportunistically [9] where there is no or limited control, cannot support such applications. So, we need a *hybrid control*, where traffic with strict QoS requirements is scheduled while the rest is served through an open loop or an opportunistic mechanism.

Finally, most proposals (apart from CBOSS [7]) target hyperscale DC. Hypescale size requires complicated and expensive optical network architectures, while scheduling and control turn to be the bottlenecks. Emerging applications such as 5G C-RAN, IoT, VR, autonomous driving, etc. require low latency, posing a strict constraint on distance (tens of km), which cannot be met by the centralized DCs model. So a distributed Edge Cloud model, where several small DCs are close to the users and collaborate over the metro network is expected to appear in the near future [14]. Optical switches could very well target small to medium size DCs, where scheduling and control complexity is substantially reduced. Note that Edge Cloud is vastly related to latency sensitive applications. So, scheduling will definitely be required, and the techniques described in section 2 and a hybrid control plane are relevant. Finally, in the Edge Cloud era, resource allocation has to extend the DC limits and be performed end-to-end over the whole Edge Cloud [15].

4. References

[1] Cisco, “Cisco Global Cloud Index: Forecast and Methodology, 2016-2021”, white paper.
[2] A. Singh, et. al. "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google’s Datacenter Network", SIGCOMM 2015.
[3] C. Kachris, I. Tomkos, “A Survey on Optical Interconnects for Data Centers”, IEEE Communications Surveys & Tutorials, 2012.
[4] N. Farrington, et. al., “Helios: a Hybrid Electrical/Optical Switch Architecture for Nodular Data Centers”, ACM SIGCOMM, 2010.
[5] G. Porter et al., “Integrating Microsecond Circuit Switching into the Data Center”, ACM SIGCOMM, 2013.
[6] P. Bakopoulos, et. al. “NEPHELE: an End-to-end Scalable and Dynamically Reconfigurable Optical Architecture for Application-aware SDN Cloud Datacenters”, IEEE Communications Magazine, 2018.
[7] N. Benzaoui, et. al., “CBOSS: Bringing Traffic Engineering Inside Data Center Networks”, JOCN, 2018.
[8] G. Saridis, et. al., “Lightness: a Virtualizable Software Defined Data Center Network with All-Optical Circuit/Packet Switching", JLT 2016.
[9] F. Yan, W. Miao, O. Raz, N. Calabretta, “OPSquare: a flat DCN architecture based on flow-controlled optical packet switches”, JOCN, 2017.
[10] W. M. Mellette, et. al. “RotorNet: A Scalable, Low-complexity, Optical Datacenter Network”, ACM SIGCOMM 2017.
[11] C. Chang, W. Chen, H. Huang, “Enhanced Birkhoff-von Neumann Decomposition Algorithm for Input Queued Switches”, Infocom 2000.
[12] K. Christodoulopoulos, et. al., “Efficient Bandwidth Allocation in the NEPHELE Optical/Electrical Datacenter Interconnect”, JOCN, 2017.
[13] I. Patronas, et. al., “Scheduler Accelerator for TDMA Data Centers”, PDP 2018.
[14] Y.-C. Hu, et. al, “Mobile Edge Computing: A Key Technology Towards 5G”, ETSI Whitepaper, 2015.
[15] N. Benzaoui, et. al., “DDN: Deterministic Dynamic Networks”, ECOC 2018.