To sync or not to sync: why asynchronous traffic control is good enough for your data center

German Sviridov, Andrea Bianco, Paolo Giaccone
Dipartimento di Elettronica e Telecomunicazioni - Politecnico di Torino - Torino, Italy

***Abstract*—Recently proposed architectures for high-performance data centers advocate the adoption of a centralized control that coordinates the packet transfers within the data center network. In such architectures, centralized algorithms perform decisions regarding packet scheduling (i.e., when a packet is transferred from the server to the switches) and packet routing (i.e., the sequence of traversed switches to reach the destination server) with the aim of optimizing the overall performance. Notably, centralized control permits to reduce packet contention and to minimize the delay introduced by the data center network, but may rely on expensive mechanisms such as synchronous transmission of the packets from the servers.**

In our work we compare a generic *synchronous* architecture for the centralized control of a data center with a generic *asynchronous* architecture, that relaxes the strict packet-by-packet control required by the synchronous architecture and enables a simpler rate-based implementation at the servers. We show that the two architectures achieve near identical performance in terms of throughput, fairness and delays. We finally conclude that asynchronous architectures offer a better trade-off in terms of complexity and performance, with better scaling properties to very large sizes.

I. Introduction

Data centers (DC) are constituted by a large concentration of servers, providing computing and storage resources, typically to run cloud computing services and real-time applications. Servers are connected through a Data Center Network (DCN) that interconnects them to the rest of the data center and provides access to the Internet, as shown in Fig. 1. Interestingly, most of the data traffic within a DCN is local, mainly due to the high exploitation of parallel processing, of the redundancy in the data storage and of the internal control mechanisms. Thus, performance perceived by the users heavily depends on the performance of the data transfers within the data center.

In recent years, the demand for low latency and high bandwidth in the DCN has grown dramatically, partially compensated by a scalable design of data centers, exploiting multi-layer Clos-based topologies [1]. The memories internal to the switches have instead kept growing slowly. Consequently the ratio of RTT over bandwidth for the intra-data center communications have kept shrinking. As a consequence, under such conditions, congestion control schemes in standard TCP protocols, which were originally tailored to WAN/LAN with high/medium RTT and medium/low bandwidth, are not able to converge fast enough, ultimately leading to poor performance. This is also exacerbated by the small size of most of the traffic flows. Indeed, the majority of flows do not last long enough to trigger congestion control mechanisms, which were originally designed for long-lasting flows. This fact has motivated the networking community to devise ad-hoc transport schemes, exclusively designed for DCNs, able to minimize the latency within the data center by exploiting a centralized traffic control. Thanks to the limited physical extension and small propagation delays, centralized control provides the possibility to globally monitor the network state in real time and to optimize the packet transmissions. Note that these ad-hoc solutions coexist with the legacy transport protocols adopted in the servers.

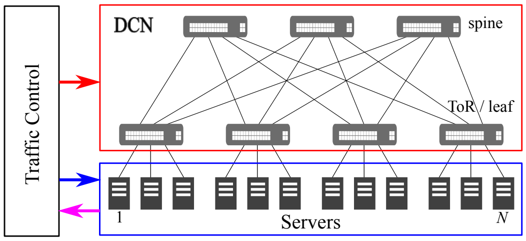


Fig. 1: Architecture of a multilayer data center with a central traffic control for packet scheduling and routing

Recently, Fastpass [2] has been proposed as a centralized scheduler for DC to coordinate the data transfer between servers. The main idea is to abstract the DCN as a logical "big switch" in which each server is connected to a single port: Fastpass ensures that at any time at most one packet is transferred to and from each server. This guarantees almost no congestion within the DCN switches and very low, close to ideal, DCN crossing latencies, accounting only for the store-and-forward delays and for the link propagation delays. However, this property comes at a cost: mainly the requirement of DC-wide synchronization among servers. Indeed, all servers must have a common time reference to trigger the transmission of each individual packet at a predefined time instant, chosen by the packet scheduler. Hence, Fastpass mimics a synchronous TDM-based network. The downside of minimizing the DCN latency is that most of the delay is now experienced at the servers, as later shown in the numerical results in Sec. III-G, which investigate the queuing spreading across the various levels of the data center hierarchy, from servers to the higher layer switches. Nevertheless, the predictability of DCN crossing delays permits to better control the overall performance of data transfers, providing a nice solution in the design of high-performance data centers.

In our work, we address the following question: Is it possible to achieve performance similar to an almost ideal synchronous architecture by relaxing the constraint on *synchronization*, thus reducing its cost and complexity? We show that the answer is positive. Indeed, fully *asynchronous* data transfers can achieve performance similar to Fastpass without any strict synchronization among the servers. The asynchronous solution is still based on a centralized controller, that now only orchestrates the actual rates at which each server injects packets in the DCN, instead of controlling the exact time when each single packet is transmitted by the servers. Consequently, we claim that architectures based on rate allocation are able to achieve a better trade off in terms of performance and complexity than synchronous architectures.

For a fair comparison, we provide a general framework to compare synchronous vs asynchronous traffic control of the packet transmissions from the servers. Finally, to motivate our main claim, we conduct an extensive simulation campaign, with a detailed simulation model of the DC, to assess the performance of the two control approaches.

The paper is organized as follows. Sec. II describes the considered synchronous and asynchronous traffic control models in DCN. In Sec. III we describe the adopted simulation model and discusses the main numerical results. In Sec. IV we discuss some previously proposed centralized asynchronous traffic control schemes. Finally, in Sec. V we draw our conclusions.

II. Scheduling and routing in DCN

As shown in Fig. 1, we assume a data center with N servers connected by a multilayer DCN, in which each server is connected to a ToR switch. ToR switches are interconnected through a multilayer Clos-based topology, e.g., leaf-spine in the case of a two-layers topology. For simplicity, we assume that the bisection bandwidth of the DCN is maximum, thus no over-subscription is present[1]. We also assume an homogeneous scenario with all the links have the same capacity. For inhomogeneous scenarios with link rates, at some layer, f times faster than at another layer, it is possible to construct an equivalent topology in which the faster links are split into f parallel disjoint slower links, replicating f times the switches with the higher rate ports. The process can be iterated across all the layers until all the links in the DCN equivalent topology have the same capacity.

Traffic flows are transferred between pairs of servers. The packet transmission from the servers to the ToR switches and the corresponding routing path are coordinated by a central traffic control. Transmission queues at the servers are organized as per-server destination, to avoid the throughput degradation due to the well-known head-of-line problem. Thus, a maximum of N (logical) queues are managed by each server.

A. Synchronous (SYN) architecture

This architecture is based on Fastpass and the implementation issues are discussed in [2]. We assume that all server linecards are synchronized; time is slotted according to a fixed packet transmission time. At a generic timeslot t, a synchronous controller runs a sequence of three phases, as depicted in Fig. 2:

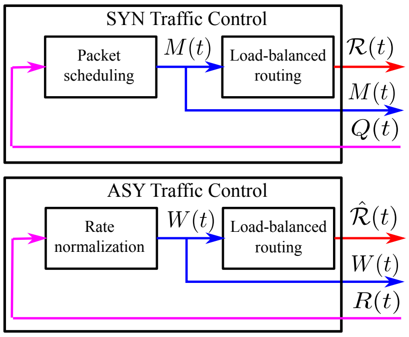


Fig. 2: Synchronous (SYN) vs Asynchronous (ASY) architecture

1) *Queue state collection*. The state of all the transmission queues at the servers is retrieved (e.g., by interacting with proper socket monitoring tools installed in the server kernel [2]). Let $Q(t) = [q_{ij}(t)]$ be a $N \times N$ matrix such that $q_{ij}(t)$ denotes the priority to transmit a packet from server i to server j, at timeslot t. As an example, $q_{ij}(t)$ can be the queueing delay of the packet at the head-of-line of the corresponding transmission queue. Thus, packets with higher queueing delay have higher transmission priority.
2) *Packet scheduling*. Based on $Q(t)$, a packet scheduler chooses a set of source-destination server pairs for transmission during timeslot t, such that at there is at most one concurrent transmission to and from each server. Thus, the scheduler computes a *matching* between the source servers and the destination servers. The matching is described by a binary $N \times N$ matrix $M(t) = [m_{ij}(t)]$, such that $m_{ij}(t) = 1$ if server i transmits the head-of-line packet directed to server j during timeslot t.
3) *Load-balanced routing*. The controller computes the routing path for each transmitted packet in $M(t)$ to balance the packets across the links of the DCN, to guarantee that all packets traverse different links. The outcome is a data structure $\mathcal{R}(t)$ that describes the routing path for all the packets in $M(t)$.

The adopted constraints in the packet scheduling and routing phases completely avoid queuing at each interface of the DCN, if assuming the same number of hops in the multilayer DCN and the same propagation delays of all links. In practice, a very limited congestion can be experienced to compensate different path lengths and different link propagation delays. Sec. III-G will be devoted to evaluate the actual queuing.

One major practical issue of the considered SYN approach is the requirement to guarantee a synchronous behavior of all the servers. Indeed, the required time accuracy becomes more strict with high port bitrates. As a reference example, consider that the transmission time of the smallest Ethernet

[1] The model can be easily adapted to DCNs with over-subscription.

packet (e.g., a TCP ACK) is about 50 ns at 10 Gbps and the one of an Ethernet MTU is about 1.2 μs. Given the quartz clock generators present in the servers and their unavoidable thermal drift (affected also by the server computation load), the precise synchronization across all the server can be achieved if relying on expensive dedicated hardware [3].

The choice of the timeslot duration is also very critical for performance, due to the possible partial filling of the transmission timeslots. In terms of throughput, small timeslots permit to reduce the waste due to partial filling, but at the expenses of introducing some control overhead to manage the fragmentation of large packets into multiple timeslots. On the contrary, large timeslots remove or mitigate the fragmentation problem, but suffer from the partial filling of the timeslots due to small packets. In terms of control information, small timeslots require higher bandwidth for the control channel between the controller and the data center components. We will evaluate in Sec. III-F the effects of the partial filling on the performance.

B. Asynchronous (ASY) architecture

We now remove the constraint of synchronization and propose an architecture based on *rate control*; the centralized controller assigns transmission rates at each server, one for each possible destination server, instead of assigning packets to timeslots as in the SYN scenario. The considered scheme runs periodically whenever the offered load changes at the servers, tracking flow-level dynamics instead of packet-level dynamics as in the SYN case. As shown in Fig. 2, a sequence of three phases occurs at time t:

1) *Offered rate estimation*. The control gathers the statistics about the offered load between any pair of servers. Similarly to the previous scenario, the statistics can be obtained by interacting with proper socket monitoring tools installed in the server kernel. Let $R(t) = [r_{ij}(t)]$ be a $N \times N$ matrix, denoted as *offered rate matrix*, with $r_{ij}(t)$ be the offered load from server i to server j at time t, normalized by the link rate, i.e. $r_{ij} \in [0, 1]$.
2) *Rate normalization*. Now $R(t)$ is renormalized to become admissible and avoid link overloading, i.e. the overall transmission rate from any server and towards any server must be lower than the link rate. The outcome of this phase is a $N \times N$ matrix $W(t) = [w_{ij}(t)]$, denoted as *transmission rate matrix* and $w_{ij}(t)$ be the actual transmission rate to adopt from server i to server j at time t. The algorithm maximizes the overall throughput $\sum_i \sum_j w_{ij}(t)$. By construction $W(t)$ is a double sub-stochastic matrix, i.e., $\sum_i w_{ij}(t) \leq 1$ and $\sum_j w_{ij}(t) \leq 1$.
3) *Load-balanced routing*. Based on $W(t)$, the controller chooses the paths to balance the traffic across the DCN. The outcome is a proper data structure $\hat{\mathcal{R}}(t)$ that describes the routing paths for the packets transferred starting from t.

Similarly to the SYN architecture, the ASY one can be easily extended to support traffic priorities, by properly scaling each value in $R(t)$.

Whenever the offered rate changes, i.e. $R(t)$ varies, the controller must re-run the three above steps. In the worst case, this occurs with a frequency which is related to the packet transmission time. Thus, the ASY system incurs (in the worst case) in an overhead similar to the SYN system in terms of exchanged control information. Instead, during the time intervals in which $R(t)$ does not change, ASY incurs in a much lower overhead than SYN.

III. Performance evaluation

To analyze the performance of SYN and ASY architectures, we first describe the specific algorithms adopted for the controllers. Then, we detail the simulation methodology and, finally, we present the numerical results.

A. Algorithms for SYN architecture

We assume a greedy maximal matching adopted for packet scheduling, based on the state of the transmission queues at the server. We consider different priority functions to define the $Q(t)$ matrix:

- Oldest cell first (OCF): the matching is performed by looking at the queuing delay of the head-of-line packets.
- Shortest remaining job first (SRJF): the matching is performed by looking at the amount of residual bytes of each flow needed to be transferred. Shortest flows are prioritized over longest ones.
- Max-min fair (MMF): the matching is performed by looking at the waiting time of the packets since they become the head of the queues.

Routing computation, coherently with Fastpass, is performed to minimize the contention on the switch output ports. To achieve this, we implemented the solution of the edge coloring problem for bipartite graphs starting from $M(t)$ by adapting the classical Paul algorithm for Clos networks in [4]. Notably, in the case of a two layers DCN, each color is associated with one distinct spine switch.

B. Algorithms for ASY architecture

To implement rate normalization, we consider a simple algorithm based on iterative matrix renormalization presented in [5]. The algorithm iteratively renormalizes rows and columns of the rate matrix $R(t)$ and it is amenable to a parallel implementation.

Each server implements one leaky bucket scheduler for each active per-destination queue, to guarantee an instantaneous rate $w_{ij}(t)$ between server i and server j.

For load-balanced routing, we deployed a standard flow-by-flow ECMP, which will be shown in Sec. III-G to be good enough to provide low buffer occupancy, and, as a consequence, small queuing delays inside the DCN.

C. Simulation methodology

We performed the analysis using the discrete-event simulator OMNeT++ [6] in combination with the libraries of INET framework, which provides detailed simulation models for the Internet protocols stacks from MAC layer up to application layer.

We considered a standard Ethernet-based leaf-and-spine topology for the DCN to compare SYN with ASY architectures, as shown in Fig. 1. The chosen topology provides full bisection bandwidth, connecting $N = 120$ servers, built with 3 spine switches, 4 leaf switches and 30 servers per leaf switch. All servers are connected to the leaf switches via a 1 Gbps link while leaf switches are connected to spine switches via a 10 Gbps link. The buffers at the servers and at the switches are assumed infinite.

Traffic flows are generated according to a Poisson process with bursty arrivals of packets belonging to the same flow and the flow size is exponentially distributed with average 46 kB. This value has been derived by Facebook data center [7].

IP packet lengths are chosen according to one of the following methods:

1) fixed and equal to 1500B to simulate bulk data transfers.
2) randomly chosen according to a bimodal distribution with 0.4 probability of generating 40B packets and 0.6 probability of generating 1500B packets; this scenario approximates the scenario typical of many applications such as Hadoop, as observed in [7];
3) randomly chosen according to the Facebook Web Server distribution (FBW) taken from [7], which refers to a web service scenario in a data center. In this scenario only 15% of the packets have are 1500B size with a median centered around 150B.

In SYN architecture, we set the timeslot corresponding to 1500B at IP layer, coherently with Fastpass [2].

We considered two different traffic patterns:

1) *incast*, where all the servers sends traffic to the same hot-spot server;
2) *uniform*, where the traffic is uniformly distributed across all servers;

The data center load is defined as the average normalized amount of traffic destined to the servers. Finally, the centralized controller is assumed to operate out of bandwidth and with zero latency.

We evaluate the normalized per-server throughput and the Flow Completion Time (FCT), in terms of average and coefficient of variation. FCT is measured from the generation of the first packet belonging to a flow until the reception of the last packet by the destination application. Each FCT is then normalized by the theoretical minimum FCT that would be achieved by that flow in an empty DCN.

FCT has been chosen among the primary confrontation metrics due to its importance in assessing the performance of typical DC delay-sensitive applications, which directly affects the quality of experience of end users. We further highlight the composition of FCT by analyzing the server latency, i.e. the average amount of time packets spend in the transmission queues inside each server before entering the DCN and the network latency, which is the average time it takes for packets to arrive to the destination server once they enter the DCN.

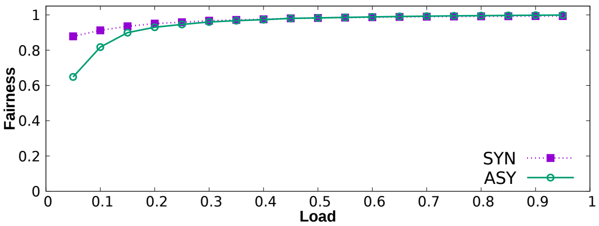


Fig. 3: Fairness comparison between SYN MMF packet scheduling and ASY rate normalization algorithms under uniform traffic

D. Fairness comparison

As a preliminary result, we show that the two architectures behave in a similar way in terms of fairness. We compare the ASY architecture computing the MMF matching with the SYN architecture with previously described rate normalization algorithm. Fig. 3 depicts the Jain's fairness index of the two systems under uniform traffic for different loads. For small load, the ASY system achieves lower fairness due to the sparseness of the offered rate matrix. However, for load higher than 0.2, the matrix becomes denser and the two architectures rapidly converge to the same, close to the optimal one, fairness index.

E. Influence of the matching policy on FCT

We investigate the effect of the metrics adopted in the computation of maximal size matching, as described in Sec. III-A, under fixed-length packets. For the sake of space, we do not report the results for variable-size packets, because they are very similar.

Fig. 4 depicts the influence of each matching metric on the FCT, under incast traffic. OCF yields the highest average FCT but at the same time lowest variance (results not reported for the sake of brevity). On the contrary SRJF by its nature minimizes the FCT but may lead to unfairness which in return increases variance and the amount of potentially missed application deadlines. MMF is a reasonable trade-off between average FCT and the corresponding variance as it balances the two metrics. All the metrics permit to achieve the maximum throughput and the same (optimal) fairness. The main factor responsible for the slight reduction in FCT in the case of ASY system is packet contention at the switches, which will be shown in Sec. III-G to be negligible.

F. Influence of packet-length distribution

In our subsequent analysis we considered the influence of packet-length distribution on the two systems.

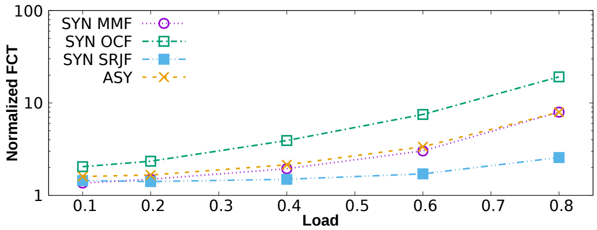
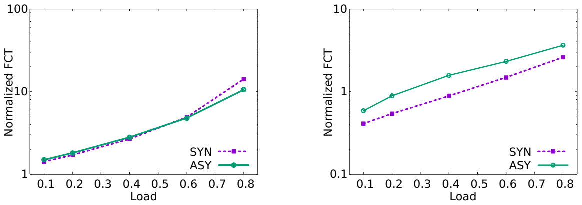


Fig. 4: Comparison among different maximal size matchings under incast traffic pattern.



(a) Average

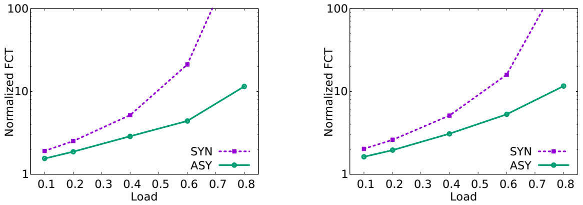
(b) Coefficient of variation

Fig. 5: FCT for fixed-length packets and uniform traffic pattern.

1) Fixed packet lengths: We obtained results similar to the case of the incast traffic pattern. From Fig. 5a it can be seen that there is no significant variation in terms of FCT between the two systems. Noticeably at higher loads the ASY system yields smaller FCT while still maintaining comparable variance. The reason behind this behavior will be later explained in Sec. III-G.

2) Bimodal packet lengths: Bimodal packet length distribution is common to applications using TCP as the transport protocol. In the case of fixed-size packets, each timeslot of SYN system was fully exploited because the packet lengths were tailored in such a way to perfectly fit inside the timeslot, leading to 100% throughput inside each timeslot. We observed that the performance of SYN system changes significantly in the presence of variable-size packets. For a bimodal packet distribution, the FCT of SYN system quickly diverges from the ASY one, which as it can be seen from Fig. 6a increases by more than one order of magnitude. It is easy to build an adversarial traffic pattern repeatedly composed by one MSS sized packet followed by one ACK packet which may reduce the throughput of the SYN system down to $\approx 50\%$.

3) FBW packet lengths: Similarly to bimodal packet lenghts distribution, in the case of FBW the SYN system does not achieve 100% throughput due to the partial timeslot filling. Results depicted in Fig. 6b show how at low load the two distributions lead to similar FCT. However, at higher load, even if FBW packet lengths lead to smaller FCT with respect to the pure bimodal one, it is still one order of magnitude larger with respect to the ASY case.



(a) Bimodal

(b) FBW

Fig. 6: FCT for bimodal and FBW packet lengths and uniform traffic pattern.

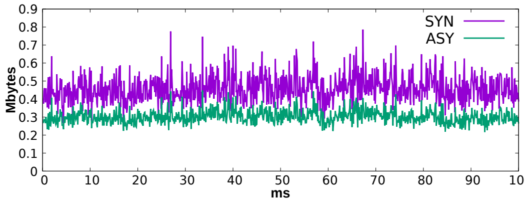


Fig. 7: Transmission buffer occupancy at the servers

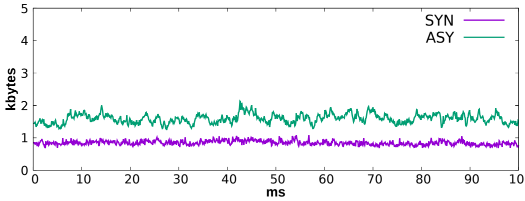


Fig. 8: Per-port buffer occupancy at the leaf switches

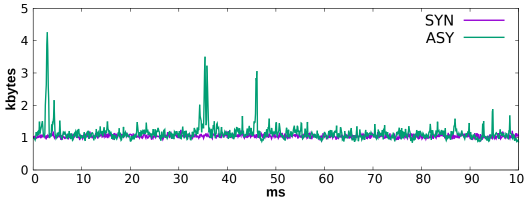
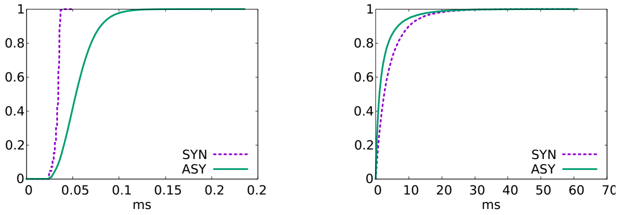


Fig. 9: Per-port buffer occupancy at the spine switches

G. Queueing within the data center

Figs. 7-9 show a 100ms trace of buffer occupancy averaged across all servers, for 0.8 load under uniform traffic pattern and fixed packet lengths. It is immediate to notice that the ASY system provides a more balanced distribution of buffer occupancies across the entire data center. The ASY system is able to keep the server memory 40% lower with respect to the SYN architecture at the expense of slightly bigger buffering at the switches. Notably, this reduction is relevant because it mitigates the resources overhead of managing per-destination queuing at the servers. In particular, we observed that the 40% reduction in the buffer occupancy at the servers corresponded to a 45% increase in the average buffer occupancy at leaf switches (but no significant variation in the 99th percentile with respect to the SYN case) and 100% increase in the average buffer occupancy at spine switches (with a 99-percentile around 7.5 kB, which is still very small).

In Fig. 10a we show the impact of queuing on the experienced delays under uniform traffic pattern with fixed length packets. Although the 50th percentile of the packet network



(a) CDF of network latency.

(b) CDF of FCT.

Fig. 10: Latency distribution inside the network for SYN and ASY systems under 0.8 load and uniform MTU-zised traffic.

latency for ASY architecture is double that of the SYN, the overall FCT is still dominated by the latencies at server queues. In fact, when compared with the CDF of FCT in Fig. 10b, the contribution due to network latency can be seen to be negligible. The cost of providing protection against contention inside the DCN in SYN architecture is that of experiencing larger delays at the servers, which, as shown, for high load becomes the dominant cause of an increased FCT.

IV. Related work

The most relevant work is Fastpass [2], which has motivated our work and has been already discussed in the previous sections, as a reference model for the SYN architecture.

Flowtune [8] follows all the three phases described for ASY architecture. It adopts a per-flowlet offered load estimation. Furthermore, it proposes a centralized rate normalization algorithm, based on the solution of a network utility maximization (NUM) problem. Thus, the transmission rate matrix is obtained based on a generic fairness function and can become equal to the one adopted in our work for a proper chosen utility function. The load-balanced routing adopts ECMP as in our case. Moreover, [8] focuses completely on the scalability aspect of centralized rate assignment while obviating the comparison of the proposed approach with the synchronous one. Numfabric [9] leverages the same architecture than [8] but now the NUM problem is solved in a distributed way by relaxing the constraint on the maximum utilization of a link, which leads to suboptimal rate allocations. Finally, due to its distributed nature, the solution for the NUM problem requires multiple RTT in order to converge.

Hedera [10] follows just the two phases of offered rate estimation and of load-balancing routing in ASY architectures. Differently from our model, the load balancing scheme reroutes flows to balance the offered traffic across the topology, leaving to TCP the overall maximization of the network utilization. Thus, the rate estimation is based on the link load, without the interaction with the servers.

Finally, all these previous works [8], [9], [10] do not provide any performance comparison with the SYN architecture, which is instead our main contribution.

V. Conclusions

We investigated how centralized asynchronous traffic control architecture compares to the synchronous one. We show that by relaxing the constraint on time synchronization and by employing simple rate limiting at each server, it is possible to obtain comparable performance for both systems, in terms of throughput, fairness and flow completion time (FCT). In terms of impact on queuing, the asynchronous system distributes the queuing delays across all the components of the data center, with smaller delays at servers at the expense of slight increase in queuing inside the DCN switches, with respect to the synchronous architecture. Notably, due to the large number of (logical) per-destination queues managed by the server in both architectures, the asynchronous approach reduce the memory overhead of the transmission queues at the servers. Moreover the synchronous system suffers from partial slot filling due to the packetization process.

In conclusion, asynchronous architectures appear very promising for their trade-off between performance and complexity. An in-depth investigation of all the implementation issues involved in asynchronous architectures with centralized traffic control and their experimental performance assessment deserves future investigations.

References

[1] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano *et al.*, “Jupiter rising: A decade of Clos topologies and centralized control in google’s datacenter network,” in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 183–197.

[2] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, “Fastpass: A centralized zero-queue datacenter network,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 307–318, 2015.

[3] K. S. Lee, H. Wang, V. Shrivastav, and H. Weatherspoon, “Globally synchronized time via datacenter networks,” in *SIGCOMM*. ACM, 2016, pp. 454–467.

[4] J. Y. Hui, *Switching and traffic theory for integrated broadband networks*. Springer Science & Business Media, 2012.

[5] R. Sinkhorn, “A relationship between arbitrary positive matrices and doubly stochastic matrices,” *The annals of mathematical statistics*, vol. 35, no. 2, pp. 876–879, 1964.

[6] A. Varga and R. Hornig, “An overview of the OMNeT++ simulation environment,” in *Simutools*. ICST, 2008.

[7] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, “Inside the social network’s (datacenter) network,” in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 123–137.

[8] J. Perry, H. Balakrishnan, and D. Shah, “Flowtune: Flowlet control for datacenter networks.” in *NSDI*. USENIX, 2017, pp. 421–435.

[9] K. Nagaraj, D. Bharadia, H. Mao, S. Chinchali, M. Alizadeh, and S. Katti, “Numfabric: Fast and flexible bandwidth allocation in datacenters,” in *SIGCOMM*. ACM, 2016, pp. 188–201.

[10] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, “Hedera: Dynamic flow scheduling for data center networks,” in *NSDI*. USENIX, 2010.