Deterministic Dynamic Networks (DDN)

N. Benzaoui, M. Szczerban Gonzalez, J. M. Estarán, H. Mardoyan, W. Lautenschlaeger, U. Gebhard, L. Dembeck, S. Bigo, Y. Pointurier

***Abstract*— Today's network infrastructure evolves into two seemingly opposite directions: cloudification centralizes functions that used to be distributed for economies of scale, at the expense of latency, while latency-constrained applications are surging. this calls for a new architecture capable of distributed computing: The Edge Cloud network. Future 5G applications will impose strict latency and dynamicity requirements on the Edge Cloud, in intra- and inter-data center networks. The Edge Cloud needs a network infrastructure able to deliver: low latency (~microseconds), deterministic data delivery in time (~nanoseconds jitter) and dynamic reconfiguration (~milliseconds) between objects (antennas, robots) in data centers or across data centers, through a fronthaul network.**

In this paper we propose, implement and demonstrate Deterministic Dynamic Network (DDN)-based Edge Cloud network. On a real-time testbed we achieve network slicing, low, deterministic latency of only tens of microseconds per-application (per-flow), when competing technologies cannot provide per-flow guarantee. We also show that the network can be dynamically reconfigured at the millisecond timescale.

***Index Terms*— Time slot network, deterministic network, edge cloud, data center, 5G, industry 4.0, latency, jitter, end-to-end performance, quality of service guarantee, slicing.**

I. Introduction

Edge Cloud is a network architecture [1] based on distributed Data Centers (Fig. 1), where raw time-sensitive data is sent, through an optical network infrastructure, from endpoints (e.g., antennas, sensors, users) to the closest data center to be processed. The Edge Cloud came as an evolution of the Centralized Cloud network. The initial motivation to adopt a Centralized Cloud architecture was to reduce costs by sharing the processing hardware among multiple endpoints. Centralized Cloud was originally proposed as an infrastructure for the Centralized Radio Access Network (CRAN) [2] where mobile user data processing functions are virtualized and moved from the antenna to a centralized data center. However, long propagation time on the optical fronthaul link – between the antenna and the data center – that led to increased latency, motivated the move towards a decentralized solution – the Edge Cloud.

By bringing a pool of processing resources where the traffic is – closer to the user (e.g., human, machine) – the Edge Cloud architecture offers the opportunity to cost-effectively support ultra-low latency and dense traffic demand. This architecture opened a real opportunity to the Internet of Things (IoT) to expand its use cases from non-real-time communication between static objects (e.g., printer, sensors) to the support of massive, dynamic, time-sensitive 5G applications such as:

- 5G RAN with mobile objects [3]
- Industry 4.0 [4] with collaborating machines
- Vehicle-to-everything communications with self-driving cooperative cars and road traffic regulation [4]
- Health sector with remote surgery intervention [6]
- high-frequency trading [7].

The conjuncture of the Edge Cloud architecture, the Internet of Things and 5G applications in general is transforming the telecommunications landscape from a user-to-user or user-to-

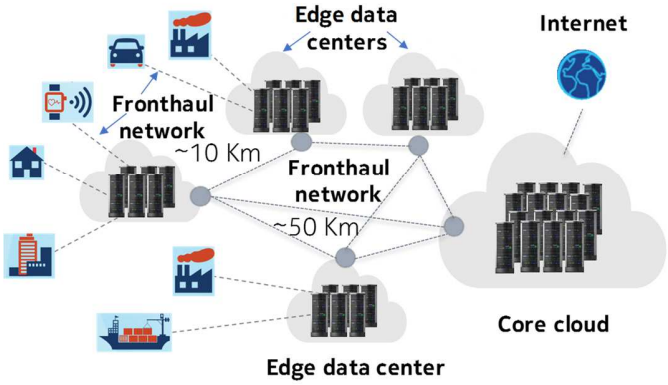

Fig. 1. Edge Cloud architecture network.

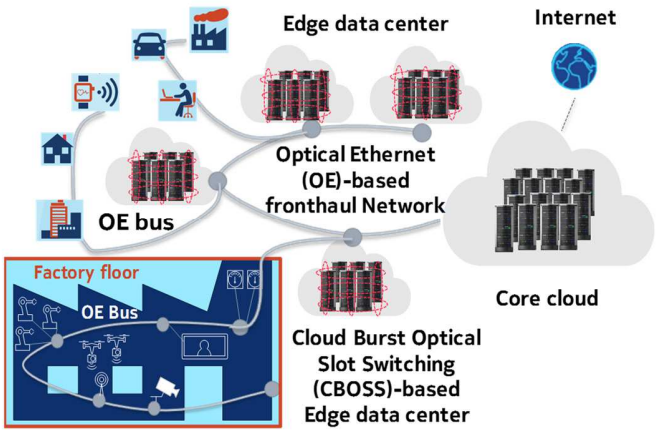

Fig. 2 Deterministic and Dynamic Network-based Edge Cloud network

17 December 2018.
This is an extended version of the ECOC'18 post-deadline paper "DDN: Deterministic Dynamic Networks" [27].

N. Benzaoui, M. Szczerban Gonzalez, J.-M. Estarán, H. Mardoyan, Y. Pointurier and S. Bigo, are with Nokia Bell Labs, Nozay, France, (email: nihel_djoher.benzaoui@nokia-bell-labs.com)
W. Lautenschlaeger, U. Gebhard, and L. Dembeck are with Nokia Bell Labs, Stuttgart, Germany.

TAB. 1: FUTURE EDGE CLOUD REQUIREMENTS AND CHARACTERISTICS

(a) Support of time-sensitiveness and statistical multiplexing	Yes
(b) End-to-end service turn-up time	<1 ms
(c) End-to-end latency (excluding propagation)	10's-100's µs
(d) End-to-end jitter	<1 µs
(e) End-to-end Packet loss rate	<< 10^{-10}
(f) Number of competing time-sensitive flows	>100
(g) Number of machines in each edge cloud	~200
(h) Typical end-to-end propagation distance	~1-50 km

object, to an object-to-object communication paradigm.

In such a vision, any object can request an IT service with any other distant object, to potentially exchange data for a very short time (sub-second). This new communication pattern increases the time-dependence volatility (dynamicity) of the traffic between the data centers. We expect traffic dynamics in the fronthaul (inter-data center) to be similar to the one currently governing intra-data center communications (70% of data traffic lasts less than 500 milliseconds [8]). Hence, the service turn-up time, that needs to be several orders of magnitude lower than the service duration, is required to be sub-milliseconds.

Intrinsically to the time-sensitive nature of those 5G applications, strict constraints on absolute latency and its standard deviation – jitter – are put on the Edge Cloud network. The latency requirements of already identified use cases is today ranging from few hundreds of microseconds to tens of milliseconds. As examples of latency and jitter requirements, we give:

- 5G RAN: 100 microseconds latency [3][9]and below 100 nanoseconds jitter [10].
- Industry 4.0: less than 100 microseconds latency and jitter from 30 nanoseconds to a few microseconds [11][12].
- Vehicle-to-everything communications: 10 to 100 milliseconds [13].

We expect future applications to tend to take full advantage of the Edge Cloud performance and stretch the latency at the network limits, hence setting the latency constraint at few tens of µs (propagation excluded). Consequently, the jitter – that should be around one order of magnitude lower than latency – should be set a sub-micro second value. In addition, for some applications such as Industry 4.0, a high network reliability is needed. Hence the network should provide a packet loss ratio (PLR) below 10^{-10}. This PLR target can be met using protocols like TCP – based on packet retransmission in case of losses, but at the expense of prohibitive latency and jitter incompatible with Edge Cloud requirements.

Tab. 1 summarizes the characteristics and requirements foreseen for the Edge Cloud. Based on the discussion in previous paragraphs, we envision that future Edge Cloud networks will need an infrastructure able to support both time-sensitive and best effort traffic (a). This infrastructure has to support dynamic traffic (b) and deliver very low latency (c), jitter (d) and PLR (e). All this in a per-application (f) and end-to-end fashion; from an object to another, potentially inside an edge data center (g), and crossing the fronthaul or factory floor network (h). Note that Tab. 1 gives estimates; exact values depend on the application.

The Edge Cloud network asks for a deterministic (guaranteed latency and jitter) and dynamic infrastructure. But most of the solutions proposed for future networks are still relying either on technologies using inflexible, quasi-static optical pipelines – unable to support highly dynamic traffic – or on electronic technologies using transmission with no guarantee of delivery – incompatible with the strict needs of 5G applications in terms of quality of service.

This motivated us to propose a radical technological shift by leveraging two time-slotted network technologies proposed by Bell Labs, which altogether meet particularly well the requirements of Edge Cloud networks: CBOSS and OE. CBOSS [14], is an all-optical (no opto-electric conversions at intermediate nodes) technology optimized for energy-greedy environment, therefore, preferred for intra-data center interconnection. OE [15] is a partially opaque technology (transiting traffic is partially processed at intermediate nodes) with low latency Forward Error Corrector (FEC), optimized for long-reach transmission. OE has already been proposed for metro networking and is positioned for Edge Cloud fronthauling.

In this paper, and for the first time, we propose, implement and demonstrate a combination of those two technologies within an SDN-controlled environment.We demonstrate an end-to-end Deterministic Dynamic Network (DDN)-based Edge (Fig. 2) Cloud network. We show low latency of only tens of microseconds (excluding propagation delay) and sub-100 nanoseconds jitter per-application on a network that can be dynamically reconfigured at the millisecond timescale.

In the following we discuss the relevance of existing technologies for the Edge cloud (Section II). We then present Deterministic Dynamic Network (DDN), a new candidate for Edge Cloud intra and inter edge data center network that can deliver a per-flow performance guarantee in a very dynamic fashion (Section III). We explain the mechanisms used to provide such low and controlled latency in a dynamic environment (Sections IV and V). We evaluate the performance of DDN and compare it to the most promising transport solution for the Edge Cloud in Section VI. Finally, in Section VII we provide main conclusions.

II. FUTURE EDGE CLOUD SOLUTIONS

In the following we discuss the relevance of existing solutions by evaluating their compliance with the foreseen Edge Cloud characteristics and requirements listed on Tab. 1.

A. Optical circuit switched

Circuit switching, e.g., OTN in its most successful form, whether paired with FlexE or not, has prevailed over years as the natural technology to allow for deterministic performance, especially in long haul networks. In OTN each service needs to be allocated a dedicated a set of network resources. This hard slicing has the benefit of isolating services from each other without any risk of mutual influence. But the obligation of hard slicing all services makes OTN fail criterion (a) of Tab. 1 since no statistical multiplexing is possible. Also, OTN requires heavy signaling for service turn-up, which results in service

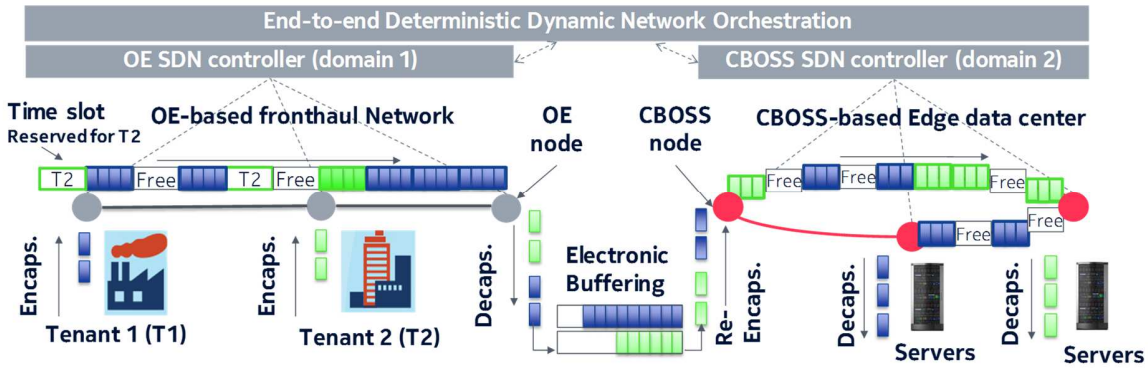

Fig. 3. End-to-end Deterministic and Dynamic Network.

turn-up times often well above the second-time scale, consequently criterion (b) cannot be met.

B. Electrical packet switched

In a highly traffic dynamic environment, Ethernet is undoubtedly the most successful implementation to cope with dynamic traffic but suffers from unbounded jitter. At constant load, average latency increases weakly when the number of competing flows increases, but peak latency (hence, jitter) grows rapidly. Large queueing delays may be rare events, but they will happen owing to statistical multiplexing, irrespective of network load; therefore, criterion (c), (d) and (e) of Table 1 cannot be met.

C. Optical and electrical TDM

Workaround approaches against the limitations of Ethernet have been implemented to support the determinism of time-sensitive traffic. They all rely on the introduction of time slots of fixed duration. For example, in PON networks using fixed bandwidth allocation (FBA), once connectivity is established, deterministic performance can be delivered. But, when multiple flows compete there is no guarantee when connectivity is granted. Hence, PONs needs to work in a static bandwidth allocation mode and consequently fail criterion (b). Industrial Ethernet was specifically designed for time-sensitive industrial applications but is not dynamically reconfigurable and can only sustain a few flows over kilometer-long distances, therefore failing criteria (b), (f) and (h). In addition, none of the above approaches can support best effort and time-sensitive traffic over the same infrastructure (criterion (a)). By contrast, IEEE 802.1 TSN [16] leverages duration time-slots which may be preempted (802.1Qbu) or reserved per class of service (802.1Qbv). Unfortunately, even if TSN has been proved to guarantee highly controlled latency as in FUSION [17][18] for two classes of service, we will show in this paper (Section VI) that performance guarantee cannot be achieved for a large number of flows belonging to a same class of service – as expected in the Edge Cloud. A workaround solution would be to ensure performance guarantee per-flow instead of per class of service, but since TSN is a fully opaque solution – all transiting traffic is processed and buffered at intermediate nodes, TSN cannot scale to hundreds of time-sensitive flows; failing criterion (f) and consequently (c) and (d). Also, TSN is not dynamically reconfigurable, failing criterion (b). [19]-[22]

In the next section we present an alternative solution that leverages complementary, highly dynamic optical slot switching technologies to provide guarantees end-to-end on a per-flow basis, as seen below.

III. DDN ARCHITECTURE

DDN (Fig. 3) is a homogeneous time slotted network fabric, where client packets are aggregated into short time slots (few microseconds), as shown in Fig. 3. For each time slot a header is built and sent either, out-of-band, over control channel (CBOSS) or, in-band, over data channel (OE). A header contains control information (e.g., routing, quality of service management) common to all client packets carried in the corresponding time slot. In DDN, time slots may be reserved to carry time-sensitive data traffic in order to guarantee channel access in time and/or capacity. The main differentiator of DDN and classical time division multiplexing (TDM) calendar-like allocations - used for instance in TSN - is the opportunistic use of time slots. Indeed, in DDN any node (CBOSS/OE) can claim any empty and unreserved slot to insert its own best effort traffic. Opportunism decreases scheduling complexity while allowing statistical multiplexing: that is an appreciable benefit

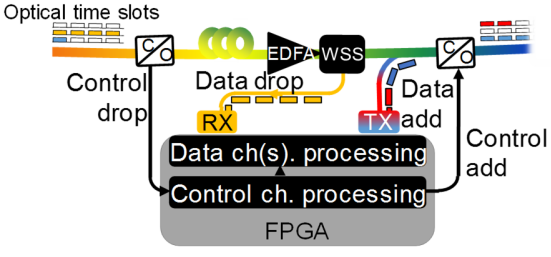

Fig. 4.a. CBOSS node architecture.

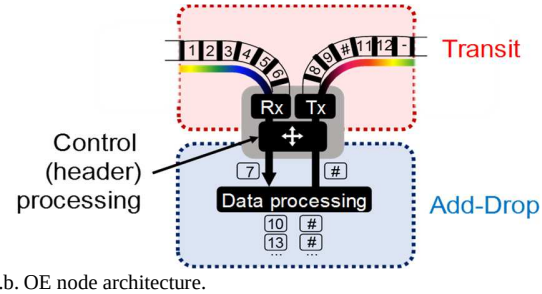

Fig.4.b. OE node architecture.

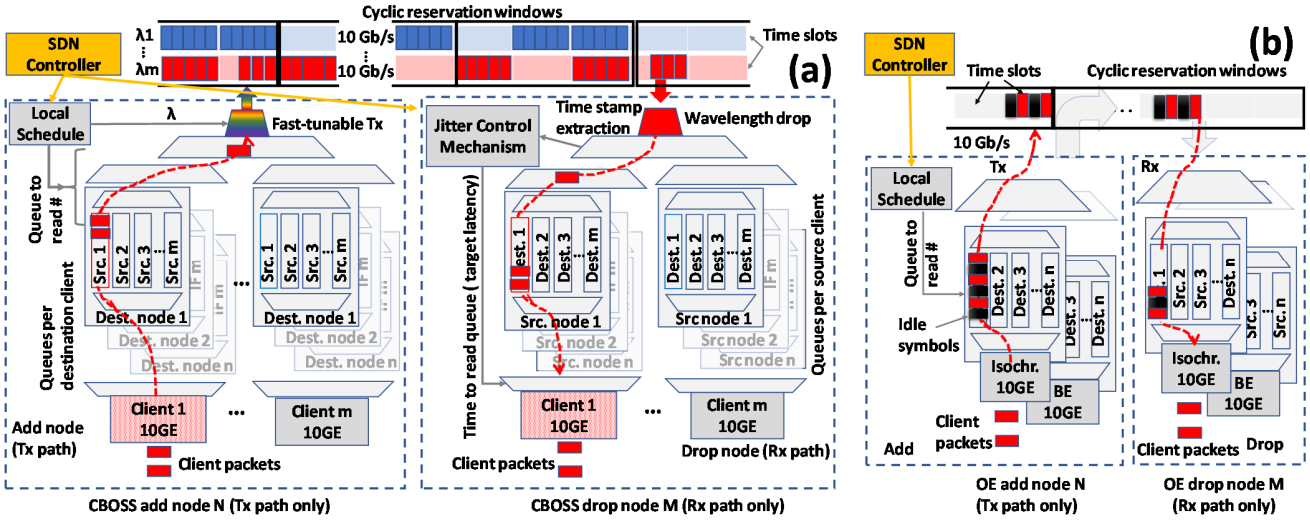

Fig. 5. Queue management for latency and jitter control in (a) CBOSS and (b) OE

in the Edge Cloud where we expect best effort traffic to maintain dominance. Note that DDN can encapsulate any upper layer protocol. To optimize throughput-efficiency during client data encapsulation into time slots, a segmentation and reassembly (SAR) mechanism is used.

From an end-to-end point of view, transitions between CBOSS and OE domains are done in the electronic domain (Fig. 3) to avoid the need of slot synchronization and slot size compatibility across domains. For both OE and CBOSS domains, resource allocation is centrally managed by an SDN controller. Each controller compute and distributes to DDN nodes in its perimeter a schedule of slot reservations. By making the SDN controllers of different domains collaborating, slots may be dynamically reserved end-to-end to deliver slot-based virtual circuits. The cooperation of the SDN controllers can be helped by an Orchestrator that simplifies the communication phase between the SDN controllers (Section V). Time-sensitive flows can therefore be physically isolated and carried across the network without interaction with best effort traffic or between themselves, hence providing hard slicing.

A. *CBOSS for intra data center networks*

CBOSS (Figs. 4.a and 5.a) is a time and wavelength division multiplexed network that relies on an all-optical switching fabric to provide high-performance communication between Top of Racks (ToRs). CBOSS network interconnects ToRs through colored fixed-duration optical slots that transport client data. At the physical layer, we demonstrated that CBOSS can multiplex dozens of 200 Gb/s channels on the same ring and traverse tens of nodes [23]. Transmission of optical slots is done using fast-tunable transmitters that adapt wavelength for each slot according to the destination node. The optical per-wavelength routing is possible since, at reception, each node implements a wavelength dropper (1x2 Wavelength Selective Switch) that extracts a set of pre-defined wavelengths. The data carried by these wavelengths is transported all optically (without opto-electronic conversion), hence latency due to electronic processing and buffering at intermediate nodes is removed. Only the control channel carrying the slot headers is electronically processed at each node. A one slot fiber delay line is used to re-align control and data channel.

B. *OE for fronthaul networks*

OE (Fig. 4.b and 5.b) is a time slotted network, partially regenerating (electrically) data at each intermediate node. OE is designed to scale to 1.6 Tb/s (4 wavelengths at 400 Gb/s each). In OE, slot headers are attached to the slot itself. Even if OE is an opaque technology it reduces electronic processing and buffering through two mechanisms:

1) At intermediate nodes, only headers are processed and full latency-hungry processing (e.g., at least 5 – 20 microseconds FEC delay per hop [24]) is performed only at in/egress nodes,

2) The transiting traffic has strict priority over the inserted traffic at intermediate nodes.

In the following section, we explain how we control latency and jitter in both CBOSS and OE.

Note that thanks to the all-optical transport of data in CBOSS and the priority of transiting traffic in OE, in DDN each node needs to keep track only of flows that are connect to itself, as opposed to opaque solutions that needs to keep track of all flows crossing by the node. This feature combined to the opportunistic use of slots allows to DDN to remain a scalable solution while offering per-flow guarantee of service – compared to state-of-the-art solutions that can propose only guarantee per class of service.

IV. Latency and Jitter Control in DDN

A. *Latency control*

In addition to the opportunistic slot access, CBOSS and OE allow slots reservation, in a periodic way over a fixed and cyclic window (Fig.5), for dedicated per-flow connection. We define a flow as a traffic exchanged between a source and a destination client interface. As shown, in Fig.5 we perform queuing and slot allocation (scheduling) in a per-flow manner in order to guarantee quality of service per application and to satisfy future

Edge Cloud requirements (Tab. 1). Fig. 5 shows how at each time slot the schedule is used to identify the queue to read, and for CBOSS the wavelength to transmit. Figs. 5.a and 5.b shows an example of guaranteed flow transmission (in dotted lines) for CBOSS and OE, respectively. The incoming client packets are first directed to the corresponding queue – according to their source and destination client addresses. Then, at each time slot, the scheduler selects which queue to read (if the current slot is a reserved slot, otherwise state-of-the-art queue management policy is used on the best effort queues). Client packets in the selected queue are encapsulated in slots and sent over the channel – even if not fully filled for the reserved slots. To optimize latency, if a client packet arrives in the middle of its reserved slot, it directly starts to be sent in the current reserved slot. If needed, client packet segmentation and reassembly can be used to resume the packet transmission in the next reserved slot. At reception, client packets are again buffered on the corresponding queue before being directed to the right client interface.

Using a periodic reservation of slots for a given flow, we achieve network slicing at the physical layer and can isolate flows and provide guaranteed latency according to each flow requirements, unlike TSN that cannot achieve per-flow latency control as explained in Section II.C.

B. Jitter control

In a time-slotted network using slots reservations, we identify two main jitter sources:

- Forcing a client packet that can arrive at any time on the cyclic window to wait for its reservation; its waiting time may vary from 0 to the duration of the reservation window.
- Segmentation and reassembly mechanism; assuming client packets and slots of a same size. 1) If a client packet is already buffered, it will be entirely transmitted in the next reserved slot. 2) If a client packet arrives near the end of a reserved slot, a first fragment of it may be sent, then the remaining fragment will be sent in the following reserved slot. Difference between 1) and 2) creates jitter.

To deal with latency variations, two approaches are possible:

1) Jitter compensation: In this paper we propose to compensate the jitter at the destination node where all client packets are buffered at reception (Fig. 5.a) until their *time-in-network* (latency experienced in the network) reaches a pre-defined target latency. First, at the reception of a client packet, the node time stamps the client packet before buffering it. To insert the time stamp, each node relies on a local clock counter that provides the local reference time. Then, each node broadcasts its current reference time through the control channel. Each node builds a lookup table where it stores the difference between its own local reference time and that of the other nodes in the network. The lookup table is updated periodically accounting for any change in the difference of time reference of nodes. Finally, at the destination node the time stamp is read, and the lookup table is used to calculate the *time-in-network* and estimate the penalty time that the client packet must wait, so it reaches the target latency. The target latency should be larger or equal to the worst latency (T_{max}) in the network. In DDN, T_{max} can be pre-calculated and is defined as:

$$T_{max} = T_{min} + \left(T_s * \frac{\mathrm{W}}{\mathrm{N}}\right) * \left\lceil \left(\frac{T_p}{T_s}\right) \right\rceil \quad \text{(eq. 1)}$$

where T_m is a minimum latency due to fixed client delay, encapsulation, segmentation and reassembly processing within the node. T_p is the client packet duration, T_s is the slot duration and N the number of reserved slots (uniformly spread) over a window W.

Note that in [27] this mechanism was software-emulated, while in this paper we propose a hardware implementation on the DDN nodes. Running the jitter compensation mechanism in real time that provides a deterministic jitter transmission is the major contribution of the paper.

(2) Isochronous interface: In [24] we proposed a new mechanism where client packet traffic is shaped into a bitstream flow before insertion. More precisely, and as shown in Fig. 5.b, the incoming client packets are stored in a rate shaping buffer where idle symbols are inserted at the same rate if the shaping buffer is empty. The resulting bit stream is carried at the slot reservation rate to equalize client packet inter-arrival times. Consequently, *time-in-network* variations is removed. Idle symbols are removed at reception (Fig. 5.b). Note that isochronous interface is well-adapted to constant bit rate (CBR) client traffic. Explanation are provided in [24]

Since CBR traffic is more likely to be found in the fronthaul (e.g., CPRI [3]) where OE is positioned, we implement mechanism (1) in CBOSS and mechanism (2) in OE as shown in Fig. 5. In Section VI we benchmark both solutions.

V. DYNAMICS IN DDN

Fig. 3, describes the proposed DDN-based Edge Cloud network. The control architecture of CBOSS and OE is based on a centralized SDN controller that decides on the slot allocation. Each domain controller (CBOSS or OE) computes and distributes to the network nodes in its domain a schedule of slot reservations. The schedule may be recomputed every few tens or hundreds of microseconds. Because the dynamic and fast reconfigurability of the schedule is the key parameter to adapt the network to the fast traffic variations in the Edge Cloud, in DDN slot reservations will be done in parallel across network domains from end-to-end – from object to object. This is done by making SDN controllers of different network segments collaborating to deliver slot-based slices (Fig. 3). The cooperation of the SDN controllers can be helped by an Orchestrator that simplifies the synchronization phase between the SDN controllers through three basic steps:

1) The orchestrator receives a traffic connectivity request from an object (e.g., server).
2) The orchestrator relays the request to the SDN controllers.
3) Each SDN controller translates the traffic request into slots reservation and either validates the request and apply the new reservation schedule or rejects the request.

Note that in this paper, the DDN testbed includes the SDN controllers that receive traffic request from a user interface, and not yet from the orchestrator. The communication between the object, the orchestrator and the SDN controllers is an ongoing

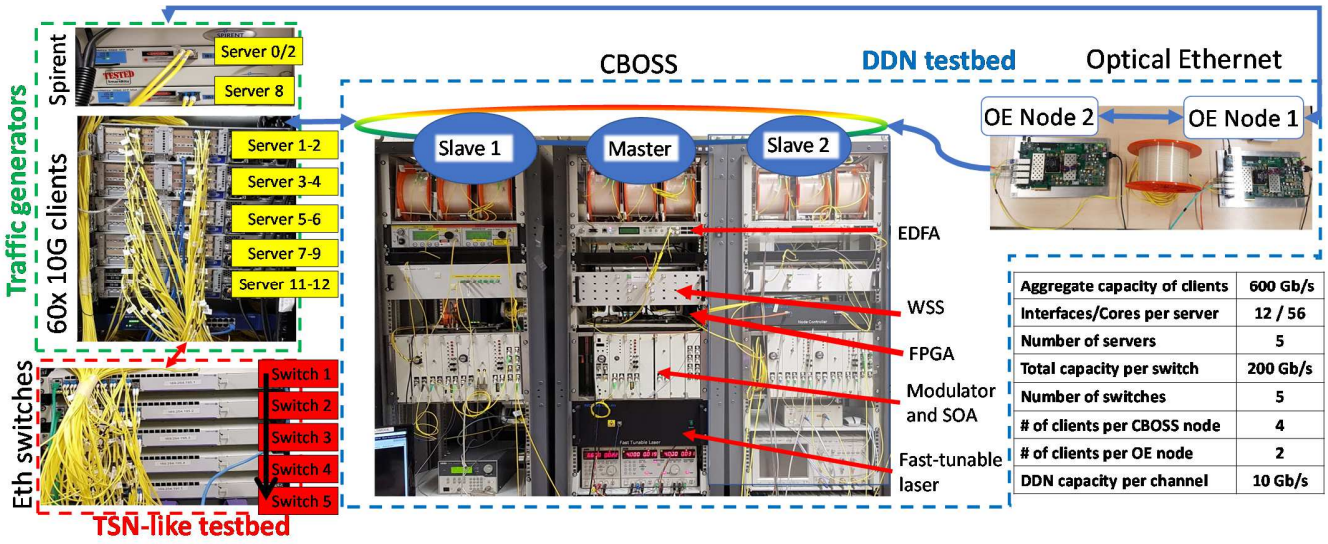

Fig. 6. DDN and TSN-like switches testbed (hardware).

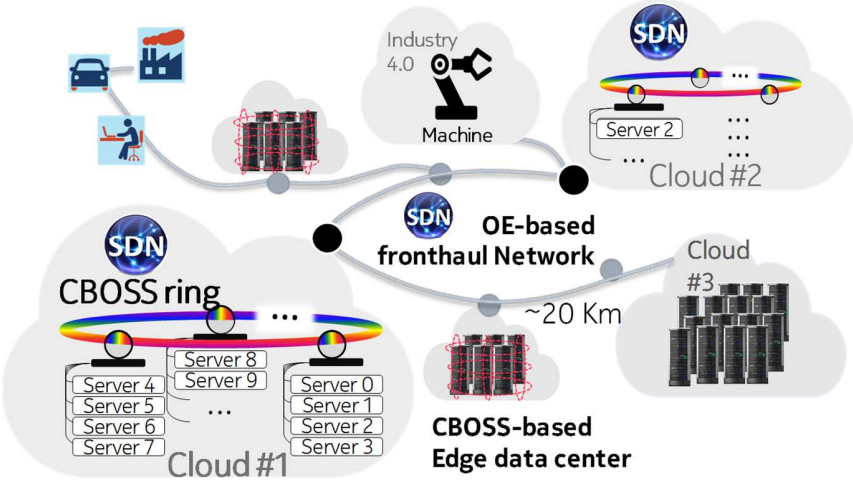

Fig. 7. DDN and Ethenet/TSN testbed (logical connection).

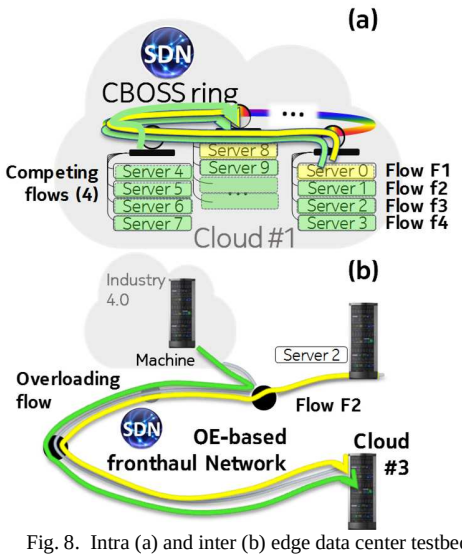

Fig. 8. Intra (a) and inter (b) edge data center testbed.

work.

In DDN, control plane – communication from the SDN controller down to the DDN node – is usually distributed over the IP layer. In this case, control information coming from the SDN controller is sent over an IP/Ethernet network to each SDN agents interfacing a node. This control plane architecture is used in OE, where an SDN agent is dedicated per OE node to relay the control information carried by the OE bus after being encapsulated into Ethernet frames.

An alternative to this control plan, is a proposition we implanted in CBOSS. In a CBOSS domain, the control information is transmitted to a single SDN agent which interfaces with the CBOSS network through a specific node called master node. Once the information is received by the master node the control information is sent to each other node through the control channel. This control plane architecture has the advantage to use a dedicated path – the control channel, hence avoiding any switching processing.

VI. End-to-end DDN evaluation

In the following, we evaluate the end-to-end performance of DDN on a real testbed. We compare DDN to what the community is considering as the most relevant solution to deliver controlled latency and service guarantee, TSN (802.1 Qbu and Qbv) network. To do so, we use in our testbed Ethernet switches to emulate the behavior of TSN switches in a time-sensitive environment. Indeed, as explained in Section II.C, TSN (802.1 Qbu and Qbv) guarantees performance per-class of service. Therefore, in an environment where more than one flow belongs to a same time-sensitive class of service, no per-flow performance can be guaranteed pre-emption becomes ineffective. In the following we call the Ethernet switches emulating TSN in a time-sensitive environment: TSN-like switches.

We implement, evaluate and compare DDN to Ethernet switches emulating TSN using a testbed illustrated in Fig. 6 (hardware) and Fig. 7 (logical connections). In this setup we reproduce an example of traffic exchange in an Edge Cloud

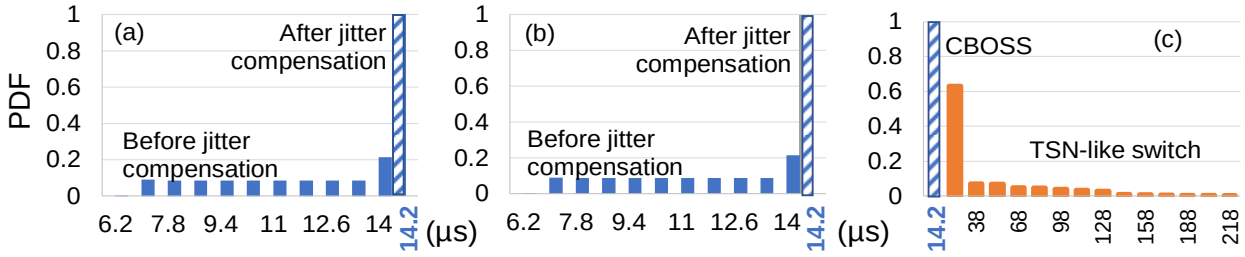

Fig. 9 Latency distribution for CBOSS: a) single-flow 100 Mb/s flow, b) single-flow 1Gb/s and c) multi-flow.

network. We connect two Edge data centers. The first Edge data center hosts servers with 12 client interfaces. These servers are interconnected using a CBOSS network with 3 nodes and 3 wavelengths. The second Edge data center is emulated by a a single server. Both data centers are connected using a 2-node OE bus. In addition, we setup a communication across the Edge Cloud between two distant machines e.g., industrial robot and the third Edge Cloud.

In the current implementation, CBOSS and OE nodes supports 4 and 2 clients, respectively. While OE and CBOSS off-line interfaces were reported earlier at higher rate, we demonstrate here an end-to-end network where all equipment run at 10 Gb/s, accounting for the constraints of real-time implementation in FPGAs. The net available capacity is 6.5 Gb/s per CBOSS wavelength and 8 Gb/s for the OE bus. The difference between the raw and neat capacity is mostly due to encoding overhead (8b/10b), and inter-slot (gap) fixed-time for CBOSS. Note that we use a hardware traffic generator from Spirent to generate constant bit rate traffic from/to some servers in Fig. 7. All other flows are generated using *Pktgen* and *MoonGen*, software-based traffic generators powered by the DPDK fast packet processing framework [25][26].

We benchmark DDN vs. TSN by replacing each involved CBOSS or OE node by a 20-ports 10G Ethernet switch (Fig. 6) acting as a TSN (802.1Qbu and Qbv) switch. We removed propagation in reported latencies. In the following, we first evaluate the CBOSS intra-DC performance and OE inter-DC performance separately, then we evaluate the end-to-end performance of the integrated DDN (CBOSS+OE) testbed.

A. *Intra-DC performance*

In CBOSS, latency determinism is guaranteed through two mechanisms: periodic slot reservation (over a window of length W slots, each slot of duration T_s = 1.46 µs and the gap time around 100 ns) to cap the maximum queuing delay, and jitter compensation (explained in Section IV.B). In order to evaluate the performance of both mechanisms combined, we choose client packets of duration T_p = 1.45µs, a window size W = 10 and reserve N = 2 slots (uniformly spread over the window). From testbed measurements, we report a minimum latency (Section IV.B) T_m of 6.4 µs, hence set a target latency for jitter compensation to 14.2 µs (T_{max} = 13.3 µs (eq. 1) with a margin). In Fig. 9.a and 9.b, we report latency distribution for a constant bit rate flow F1 sent from a Spirent replacing server 0 and 8 on Fig. 8.a.

TAB. 2: AVERAGE LATENCY AND JITTER IN CBOSS

Flow	Latency before jitter compensation	Jitter	Latency after jitter compensation
F1	10.9 µs	2.5 µs	14.2 µs +/-100ns
f2	16.9 µs	6.35 µs	37 µs*
f3	16.9 µs	6.37 µs	37 µs*
f4	16.9 µs	6.35 µs	37 µs*

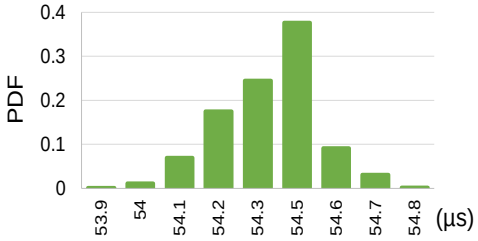

Fig. 10. Latency distribution for OE.

First F1 is set to a low rate of 100Mb/s (Fig. 9.a), then increased to a rate of 1Gb/s (Fig. 9.b). Before jitter compensation, in both cases, the pdf is spread from T_m to T_{max}. After jitter compensation, the pdf is narrowed around the target latency of 14.2 µs (at +/- 100ns precision dur to hardware measurements resolution). Thereby, CBOSS node is deterministic.

To prove per-flow guarantee in CBOSS, we keep flow F1 and inject three additional competing flows: f2, f3 and f4 (Fig. 8.a) with bursty client packet arrivals (1 burst = 2 packets) from three servers of rack 1 to three servers of rack 3. Each flow is 1 Gb/s with 2 reserved slots. Tab. 2 shows average latency and jitter for all flows. Note that due to implementation limitation (buffer size), for bursty flows (f2, f3, f4) we compensate jitter by soft (offline latency equalization) using a target latency of 37 µs corresponding to T_{max} of 34 µs with a margin. This is denoted by (*) in Tab. 2.

Determinism for F1 is maintained network-wide and all flows (with same characteristics) experience the same performance; thus, CBOSS ensures per-flow deterministic latency.

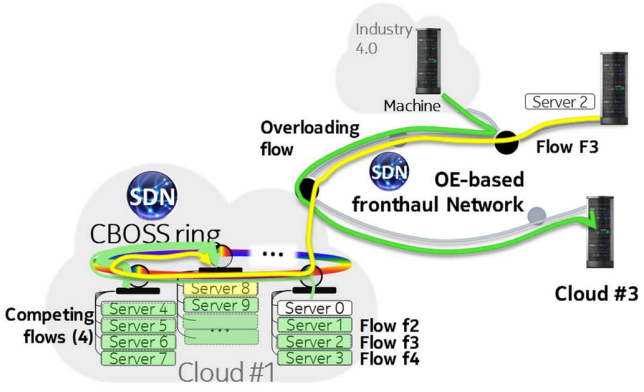

Fig. 11. End-to-end integrated DDN-based edge data center testbed.

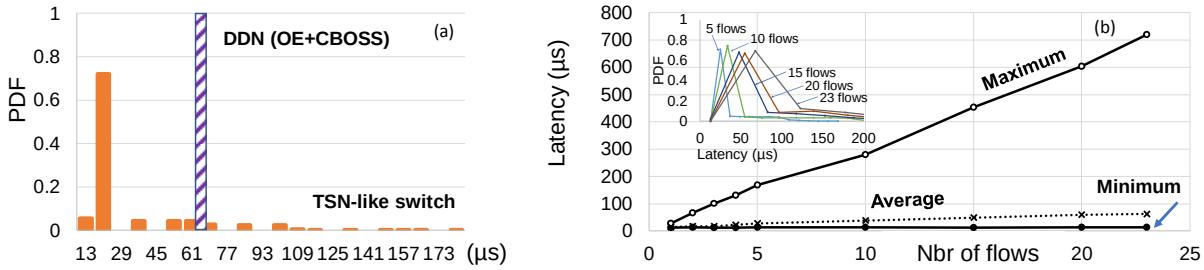

Fig. 12 End-to-end latency distribution for DDN: a) comparison with TSN-like; b) impact of number of flows for 50% load using TSN-like switches.

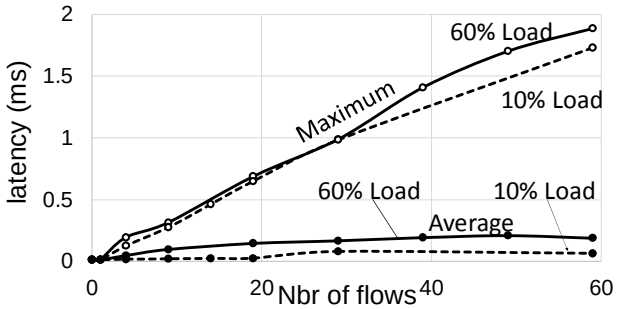

Fig. 13. Impact of number of flows on latency for 60% and 10% load using TSN-like switches.

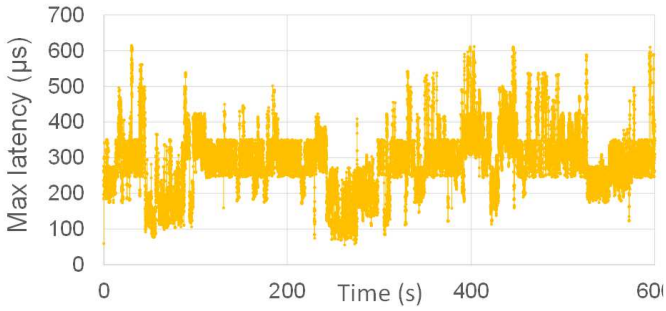

Fig. 14. Maximum latency over time, per interval of 10ms (10% load, 20 flows) using TSN-like switches.

To further stress the DDN (CBOSS) structure we increase the burstiness of the random flows (f2, f3, f4) and insert new competing flows from four servers on rack 2 to four servers on rack 3 (Fig. 8.a). Two slots are reserved for F1 and 1 slot for each other flow. Fig. 9.c compares DDN and TSN-like switches for F1. CBOSS achieves per-flow deterministic latency of 14.2 µs, while TSN-like presents an average/max latency of 33.6/205.5 µs and a jitter of 37.1 µs, which may be compatible with current applications, but will not be sufficient in future Edge Cloud time-sensitive environments.

B. *Inter-DC performance*

In OE, latency is also controlled through slot reservation and jitter is avoided at insertion by treating an input traffic as an isochronous flow shaped at the rate of reserved slots: the delay between two payloads is conserved by en/decapsulating the dummy data in-between. In OE, with a fully reserved window, we report a latency T_m = 11.2µs and a jitter of 83 ns. Here again, jitter precision is limited by hardware measurements resolution and could even be lower than what we report.

Fig. 10 reports the latency distribution of a 1Gb/s flow F2 crossing two OE nodes, using N = 3 reserved slots (quasi-uniformly spread) over a window of W = 16 slots (slot duration: T_s = 8.1 µs). F2 is a constant bit rate traffic generated by the Spirent. Again, the very narrow pdf, around T_{max} = 54 µs (eq. 1), shows that OE is deterministic.

To prove per-flow guarantee in OE, we keep F2 and introduce a competing overloading flow at the ingress OE node (Fig. 8.b). With OE the flow F2 has same performances as Fig. 10, while using TSN-like switches, we measure for F2 an average and max latency of 3.778 and 3.949 ms respectively. High jitter and losses are reported for both competing flows.

These results show that OE can transport per-flow latency-sensitive traffic in 5G fronthaul or across a factory floor.

C. *Distributed data center computing*

We evaluate the distributed data center computing performance by measuring the latency of flow F3 exchanged between two servers on two different data centers (server 2 to 8). In this scenario, we also emulate a communication between a machine in a factory floor and a distant server in a third edge data center. As can be seen in Fig. 11 this last flow is competing with F3 at the ingress OE node. Inside the first Edge data center (#1), F3 is competing with three flows from rack 1 and four flows from rack 2. Fig. 12a represents the latency distribution of the end-to-end flow F3 for a moderate load (50%). Fig. 12.a shows that DDN guarantees latency end-to-end, while F3 experiences large (maximum) latency and high jitter in the TSN-like network. Moreover, Fig. 12.b (inset: pdf) shows that even if the minimum and average latency increments slightly with the number of flows, the maximum latency increases rapidly. Fig. 12.b shows that the TSN-like performance (maximum latency) is degrading when the number of flows increases even for a constant load.

Fig. 13 shows that even when decreasing the load to 10%, the maximum latency that can be experienced by time-sensitive flows is as high as for a 60% load. Fig. 14 shows the variation of maximum latency through time for 10% load generated by 20 flows. This figure shows that even for a network with a 10% load generated by a small number of time-sensitive flows, latency and jitter are out of control even within a TSN-like network.

D. *Connection establishment time*

In DDN architecture resource allocation is centrally managed in each domain by an SDN controller. Flow establishment time was measured over CBOSS. Similar results are expected for OE at the except for control information distribution delay.

The schedule establishment time takes around 1.96 ms, as can be observed in Tab. 3 (excluding scheduling algorithm computation time). Schedule establishment time is broken down as follows:

- SDN to master node communication time: In our current setup, the communication between the SDN agent and the master node is carried out through a Universal Asynchronous Receiver/Transmitter (UART) interface running at 500 kbaud. By default, the window size is set to 10 slots, then, the full schedule size for 3 nodes is 30 bytes (1 byte per time slot). The UART transmission time is 19.4 ms. The UART communication can be substituted by a faster interface such a 10G Ethernet interface. This substitution is currently under implementation in our DDN setup.
- Distribution time: Once the master node retrieves the schedule it is distributed networkwide in few microseconds (16.7 µs mainly due to propagation for nearly 3.3 km CBOSS network).
- Execution time: Once the schedule received by a node, this later one implements and executes the new schedule in 3 clocks (19.2 ns).

Tab. 3 shows clearly that CBOSS nodes are designed to adapt fast to new resource allocations and sub-millisecond flow establishment time can envisaged by replacing the UART interface with a faster interface as explained above.

TABLE 3: AVERAGE LATENCY AND JITTER IN CBOSS

Overall schedule establishment time	UART trans. time	Distribution time	Per node schedule execution time
1.96 ms	1.94 ms	16.7 µs	19.2 ns

VII. CONCLUSIONS

The hard challenge of Edge Cloud transport network is to provide the jitter of circuit switching with the dynamics of statistical multiplexing. To address this challenge, we proposed implemented and demonstrated on a testbed a Dynamic Deterministic Network that meets future Edge Cloud requirements (Tab. 1). Our Dynamic Deterministic Network can guarantee end-to-end deterministic per-flow latency of 70 microseconds (excluding propagation delay) with sub-100 nanoseconds jitter and millisecond-timescale flow establishment, paving the way for 5G uses cases such as Industry 4.0, and future highly dynamic deterministic low latency 5G applications.

ACKNOWLEDGMENTS

This work was supported by the DGE organization from the French Ministry of Industry and the German Federal Ministry of Education and Research (BMBF) through the CELTIC+ SENDATE-TANDEM project.

The authors would like to thank their colleague Tod Sizer for their helpful discussion.

REFERENCES

[1] J. Pan and J. McElhannon, “Future edge cloud and edge computing for internet of things applications,” IEEE Internet of Things Journal, vol. 5, no. 1, pp. 439–449, Feb. 2018.

[2] S. Namba, T. Warabino, and S. Kaneko, “BBU-RRH switching schemes for centralized RAN,” in Proc. International Conference on Communications and Networking, 2012, pp. 762–766.

[3] D. Chitimalla, K. Kondepu, L. Valcarenghi, M. Tornatore, and B. Mukherjee, “5G fronthaul — latency and jitter studies of CPRI over Ethernet,” IEEE/OSA Journal of Optical Communications and Networking, vol. 9, no. 2, pp. 172–182, Feb. 2017.

[4] W. A. Khan, L. Wisniewski, D. Lang and J. Jasperneite, “Analysis of the requirements for offering industry 4.0 applications as a cloud service,” in Proc. International Symposium on Industrial Electronics, 2017, pp. 1181–1188.

[5] Qualcomm, “Leading the world to 5G: Cellular Vehicle-to-Everything (C- V2X) technologies,” Jun. 2016.

[6] M. A. Lema, A. Laya, T. Mahmoodi, M. Cuevas, J. Sachs, J. Markendahl, and M. Dohler, “Business Case and Technology Analysis for 5G Low Latency Applications,” in IEEE Access, vol. 5, pp. 5917–5935, 2017.

[7] A10 Networks, “Low Latency Trading in FIX Environments,” white paper, 2014.

[8] A. Roy, H. Zeng, J. Bagga, G. Porter, and A.-C. Snoeren, “Inside the social network’s (data center) network,” in Proc. Association for Computing Machinery Magazine, Aug. 2015.

[9] IEEE Std. P802.1CM, “Time sensitive networking for fronthaul,” Sep. 2017, http://www.ieee802.org/1/pages/802.1cm.html

[10] H. Li, L. Han, R. Duan, and G. M. Garner, “Analysis of the synchronization requirements of 5G and corresponding solutions,” IEEE Communications Standards Magazine, vol. 1, no. 1, pp. 52–58, 2017.

[11] 5G-PPP, “5G and the factories of the future,” 2015, white paper, https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-White-Paper-on-Factories-of-the-Future-Vertical-Sector.pdf.

[12] Intel, “Achieving real-time performance on a virtualized industrial control platform,” 2014, white paper, https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/industrial-solutions-real-time-performance-white-paper.pdf.

[13] K. Lee, J. Kim, Y. Park, H. Wang, and D. Hong, “Latency of Cellular-Based V2X: Perspectives on TTI-Proportional Latency and TTI-Independent Latency,” in IEEE Access, vol. 5, pp. 15800–15809, Aug. 2017

[14] N. Benzaoui, J. Estaran, E. Dutisseuil, H. Mardoyan, G. de Valicourt, A. Dupas, Q. Pham Van, D. Verchere, B. Uscumlic, M. Szczerban Gonzalez, P. Dong, Y. Chen, S. Bigo, and Y. Pointurier, “CBOSS: Bringing Traffic Engineering Inside Data Center Networks,” IEEE/OSA Journal of Optical Communications and Networking, vol. 10, no. 7, pp. 117–125, July 2018.

[15] W. Lautenschlaeger, N. Benzaoui, F. Buchali, L. Dembeck, R. Dischler, B. Franz, U. Gebhard, J. Milbrandt, Y. Pointurier, D. Roesener, L. Schmalen, A. Leven, “Optical Ethernet — Flexible Optical Metro Networks,” Journal of Lightwave Technology, vol. 35, no. 12, pp. 2346– 2357, Jun. 2017.

[16] 802.1 TSN, https://1.ieee802.org/tsn/

[17] R. Veisllari, S. Bjornstad, J. Braute, K. Bozorgebrahimi, and C. Raffaelli, “Field-trial demonstration of cost efficient sub-wavelength service through integrated packet/circuit hybrid network,” IEEE/OSA Journal of Optical Communications and Networking, vol. 7, no. 3, pp. 379–387, Mar. 2015.

[18] S. Bjornstad, “Can OTN be replaced by Ethernet,” in Proc. Optical Networks Design and Modeling, May 2018, pp. 220–225.

[19] N. J. Gomes, P. Sehier, H. Thomas, P. Chanclou, B. Li, D. Munch, P. Assimakopoulos, S. Dixit, and V. Jungnickel, “Boosting 5G through Ethernet,” IEEE Vehicular Technology Magazine, vol. 13, no. 1, pp. 74– 84, Mar. 2018.

[20] F. Durr and N.-G. Nayak, “No-wait packet scheduling for IEEE timesensitive networks (TSN),” in Proc. International Conference on Real-Time Networks and Systems, Oct. 2016.

[21] N. G. Nayak, F. Durr, and K. Rothermel, “Time-sensitive software defined networks (TSSDN) for real-time applications,” in

Proc. International Conference on Real-Time Networks and Systems, Oct. 2016.

[22] S.-S. Craciunas, R. S. Oliver, M. Chmelik, and W. Steiner, "Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks," in. Proc. International Conference on Real-Time Networks and Systems, Oct. 2016.

[23] M.-A. Mestre, G. de Valicourt, P. Jenneve, H. Mardoyan, S. Bigo and Y. Pointurier, "Optical slot switching-based datacenters with elastic burst-mode coherent transponders," in Proc the European Conference on Optical Communication, Th.2.2.3, 2014, pp. 1–3.

[24] W. Lautenschlaeger, L. Dembeck, and U. Gebhard, "Prototyping Optical Ethernet — A Network for Distributed Data Centers in the Edge Cloud," IEEE/OSA Journal of Optical Communications and Networking, vol.10, no. 12, pp. 1005–1014, 2018.

[25] P. Emmerich, S. Gallenmuller, D. Raumer, and F. Wohlfart, "MoonGen: A Scriptable High-Speed Packet Generator," in Proc. Internet Measurement Conference, Oct. 2015, pp.275–287.

[26] D. Turullab, P. Sjodina, and R. Olsson, "Pktgen: Measuring performance on high speed networks," Computer Communications, vol.82, pp. 39– 48, May 2016.

[27] N. Benzaoui, M. Szczerban Gonzalez, M.-V. Rivera, J.-M. Estaran, H. Mardoyan, W. Lautenschlaeger, U. Gebhard, L. Dembeck, Y. Pointurier, and S. Bigo, "DDN: Deterministic Dynamic Networks," in Proc. European Conference on Optical Communication, Th3B.6, 2018, pp. 1-3.