Virtual DBA: virtualizing passive optical networks to enable multi-service operation in true multi-tenant environments

JOURNAL OF
Optical Communications
and Networking

Virtual DBA: virtualizing passive optical networks to enable multi-service operation in true multi-tenant environments

Marco Ruffini,* Arsalan Ahmad, Sanwal Zeb, Nima Afraz, and Frank Slyne

CONNECT Research Centre, School of Computer Science and Statistics, The University of Dublin, Trinity College, Ireland
**Corresponding author: marco.ruffini@tcd.ie*

Received 7 October 2019; revised 17 January 2020; accepted 17 January 2020; published 3 March 2020 (Doc. ID 379894)

This paper presents the concept of virtual dynamic bandwidth allocation (DBA), a method we propose to virtualize upstream capacity scheduling in passive optical networks (PONs), which provides multiple independent virtual network operators with the ability to precisely schedule their upstream traffic allocation. After a brief introduction on the evolution of access network sharing, we present our virtual DBA architecture, detailing its main components. We then provide a summary of the work done in this area from both theoretical and practical implementation perspectives. In this paper, we propose a novel stateless algorithm for merging multiple independent virtual bandwidth maps based on priority classes and analyze its performance in terms of efficiency of capacity allocation and latency. Through our results, we discuss the existence of a trade-off between traffic load and grant size distribution versus efficiency and latency. We find that, different from a residential single-tenant application, when PONs are used for low-latency and multi-tenant applications, the system has better overall performance if grants are allocated in small size. In addition, our analysis shows that for high-priority, strict latency services, our proposed merging algorithm presents delay performance that is independent of the traffic distribution considered.

https://doi.org/10.1364/JOCN.379894

1. INTRODUCTION

Access network sharing has been a mainstream topic since 1996, when the Federal Communications Commission (FCC) released the Telecommunications Act in the U.S., opening up the market to competition. Most other countries followed shortly afterwards. Since then, several techniques have been proposed and deployed to give different operators the ability to access their customers through a third party access network (typically belonging to the main incumbent operator). These span from physical link unbundling, where an other licensed operator (OLO) can physically transmit its digital subscriber line (DSL) signal over the copper line linking the end-user to the central office (CO), to bitstream services, where the incumbent runs the access network and handles an aggregate of the customers' data to the OLO at a number of national access points [1,2]. Such techniques have evolved over time and are characterized by a different trade-off between the ability of the OLO to control quality of service (QoS) versus the amount of equipment it needs to deploy at the CO.

A decade after the Telecommunications Act was released, operators started deploying fiber-to-the-home (FTTH) to increase access capacity, as new high-bandwidth multi-media applications started to spread across the Internet. The deregulation policies developed for the copper-based access did not necessarily apply to FTTH networks, which were newly built by the (now private) operators, rather than being state funded as the old copper network. Different countries applied different rules: for example, the U.S. decided not to enforce the Telecommunications Act; on the contrary, Europe did, and the operator with the strongest national presence (i.e., with the highest market share) was forced to share its FTTH network; and Japan, on the other hand, ruled that FTTH networks needed to be shared, but the operator deploying the fiber would benefit from special incentives in order to reduce the risk associated with its initial investment [3]. As a result, today Japan has one of the highest concentrations of fiber in the access; the U.S. has a fair level of coverage, but only in dense areas or areas with a high return on investment; and Europe has lagged behind, as operators did not have a strong business case for investing in fiber [3].

Since fiber was considered a strategic asset for a nation, in some cases, the governments or the municipalities provided a subsidy. In these cases, early deployments were carried out using point-to-point fiber technology, through active optical networks (AONs). By providing individual fiber connection

to each user, AONs enable physical unbundling, facilitating network sharing, a condition that was deemed necessary for state-funded deployments. However, as fiber installation started to take off at a larger scale, most operators adopted the passive optical network (PON) architecture, as this enabled lower cost in the optical distribution network (ODN) and CO, in addition to a smaller footprint and lower energy consumption. The drawback of PONs is that they do not support physical unbundling: since they are based on a power-split architecture, where all users share the same communication channel, an OLO cannot establish a direct physical link to a subset of PON users. The only solution for PON sharing became the evolution of the bitstream technology used for DSL, called next-generation access (NGA) bitstream and virtual unbundled local access (VULA). Both offered access to PON customers at a layer-2 (L2) or layer-3 (L3) data packet level, after the optical line terminal (OLT) of the infrastructure provider (InP) has collected the user data and sent it to an aggregator switch. These enhanced the legacy bitstream by improving QoS differentiation for the OLOs.

The situation has further evolved over the past couple of years, as recent trends of network slicing and CO virtualization are providing an unprecedented opportunity to increase the level of control that OLOs have over a shared network. Once implemented in software, virtualization enables a network function to be replicated, and its control handed directly to the OLO [now called a virtual network operator (VNO)]. This has become especially important in 5G networks, which need to support a large number of highly heterogeneous services, shared across multiple VNOs (e.g., to enable cost sharing [4]).

The project that started the virtual CO revolution was the CO rearchitected as a data center (CORD) [5]. Specifically, the residential CORD (R-CORD) addressed the use case of providing residential broadband through virtualized PON infrastructure, enabling a VNO to manage its own FTTH network slice. R-CORD, however, limits the virtualization to the management plane, while data plane operations are handled by hardware white box PON units [6]. While this does not pose a problem for residential broadband applications, it limits the 5G vision of a multi-service, multi-tenant network capable of delivering highly reliable low-latency connectivity, because the scheduling of capacity cannot be controlled directly by the tenant VNOs.

Our work has progressed beyond the PON virtualization provided by CORD, as it addresses the virtualization of capacity scheduling, targeting both upstream [7] and downstream [8] operations. It has also progressed beyond the basic "softwarization" of the dynamic bandwidth allocation (DBA) proposed in [9], by designing a mechanism that enables running multiple independent DBAs in parallel, thus providing a true multi-tenant solution. In this paper, we focus our attention on the upstream scheduling (involving the DBA process), which is more complex due to its two-way request/grant operations. Downstream scheduling is instead typically implemented by applying well-known L2/L3 queue management schemes, although its use in multi-tenant environments requires some modifications, as suggested in [10].

Our DBA virtualization approach [11], which was standardized by the broadband forum (BBF) [12], allows different VNOs to run their own DBA algorithms, to suit the types of services they want to offer to their customers. This enables, for example, running one DBA for one set of users (e.g., residential broadband) and another DBA for a low-latency application [e.g., cloud radio access network (C-RAN) transport through the cooperative DBA mechanism standardized in [13]].

This paper extends the work presented in [14] at the 2019 OSA Advanced Photonics Congress and is structured as follows. Section 2 describes the overall virtual DBA (vDBA) principle and architecture, discussing building blocks and main requirements. Section 3 summarizes our work to date on the topic, highlighting some of the key performance indicators (KPIs) we have obtained in both our simulations and testbed implementation. Section 4 provides the details of our novel merging engine architectural block and analyzes the performance of the proposed algorithm. Finally, we conclude the paper.

2. VIRTUAL DBA ARCHITECTURE

The scope of the vDBA architecture is to provide a mechanism to slice the upstream capacity allocation in a PON so that different tenants (e.g., VNOs) can schedule the optical network units' (ONUs') burst allocation precisely within each frame. While higher-layer mechanisms are also capable of providing an average capacity assignment that reflects the VNOs' assured rate allocation, these do not give the VNOs the ability to precisely schedule their capacity. Typical PON standards aim at providing each ONU with its assured bandwidth on timescales of "a few ms" [15]. While this is sufficient for general broadband services, it cannot sustain the upcoming 5G type of applications that require sub-millisecond latency. One example is the support for C-RAN transport, following the low-latency cooperative DBA concept [13], described in more detail in the next section.

A. Overall Architecture

The vDBA gives a VNO the ability to schedule its own capacity allocation within each frame. The VNO can thus select the DBA algorithm that is most appropriate for the service it needs to deliver. For example, supporting a C-RAN transport service on an enhanced common public radio interface [16] interface requires the ability to deliver latency below 100–150 μs for variable rate traffic. C-RAN with a physical layer or low medium access control (MAC) functional split can be supported in a PON only through a mechanism called cooperative DBA [13], which bypasses the high-latency report/grant process of typical DBAs. This was originally introduced in [17], for its application to static fronthaul transmission and then, more recently, extended to also cover variable rate traffic for functional splits higher than pure fronthaul [18]. The cooperative DBA solves the issue by enabling communication between the mobile base band unit (BBU) scheduler and the PON scheduler. Since the BBU will schedule the user equipment (UE) traffic at least one millisecond in advance of the UE transmitting it, this information is passed directly to the OLT DBA scheduler. Thus, the DBA knows in advance when the ONU will receive the upstream traffic from the UE and

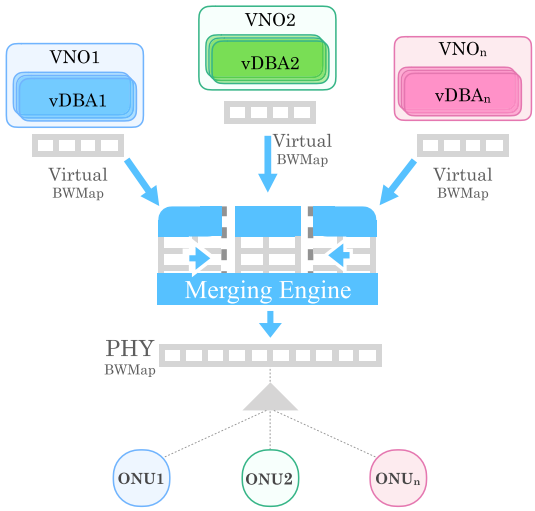

Fig. 1. Architectural design of the virtual DBA mechanism.

can schedule it without waiting for a request from the ONU, enabling just-in-time scheduling.

However, this requires the DBA to be in perfect synchronization with the BBU scheduler, so that when the scheduled data arrive from the UE to the remote unit (RU), the ONU serving that RU can transmit it without delay (this is possible because the DBA would have granted the ONU a transmission opportunity exactly for that time).

In order for the cooperative DBA to work, the VNO needs to precisely allocate capacity for each frame in the PON bandwidth map (BMap), a feature that is not supported by current commercial systems. Our proposed vDBA mechanism provides support for the C-RAN functional split, as it enables each VNO to run one (or more) virtual DBAs, each generating a virtual BMap, where the DBA algorithm can specify the exact position of the desired capacity allocation in each frame.

The vDBA architecture is shown in Fig. 1. Each VNO can operate one or more vDBA algorithms, to suit its customers' requirements. Where the system works with dynamic bandwidth report upstream (DBRus), these are generated by the transmission containers (T-CONTs) in each ONU and from there forwarded by the merging engine block to the corresponding vDBA. For ultra-low-latency mechanisms, where the reporting mechanism is bypassed, it is the application generating the request (e.g., a BBU scheduler in the case of cooperative DBA) that sends the report message to the appropriate vDBA through a standardized interface [12]. After collecting DBRus, each vDBA will run a scheduling allocation, which can be carried out every frame (if ultra-low latency is required) or else every N frames (with N being a value agreeing with the merging engine block). Such scheduling is forwarded from the vDBAs to the merging engine in the form of a virtual BMap, which reports the exact location of the required grants for each customer's T-CONTs. After collecting all virtual BMaps, the merging engine runs an algorithm for combining them into a single physical BMap, which is then distributed to all ONUs. The ONUs then follow the scheduling information reported in the BMap according to typical PON standards.

B. Merging Engine

Since we have assumed that each vDBA operates independently, their virtual BMaps can have conflicting allocations, in which case the merging engine will shift them across the frame to eliminate any overlap. Different algorithms for the merging engine can have different performances, which can favor, for example, efficiency in capacity allocation or latency. We will present performance results of our proposed algorithm in Section 4. In order to prioritize traffic, for example, for low latency, the vDBA communicates the latency requirement of the different applications it is serving to the merging engine by using pre-established labels in the virtual BMap. With this information, the merging engine can decide which allocations to shift and by how much, depending on their relative priorities.

For example, in our initial approach, further described in Section 4, we have defined four priority types. Type 4 has the strictest latency requirement and is followed, in decreasing order, by the other types, down to 1. Also, we assume that types 4 and 3 operate with a cooperative DBA type of approach, which means that in the case of overlapping virtual BMap grants, the merging engine can only postpone allocations (i.e., never anticipate them) with respect to the original vDBA request. These are followed by lower-priority allocations 2 and 1, which instead operate via the common DBRu mechanism. This means that data are available in the ONU's queue at least one frame in advance; thus the allocation can be shifted in both directions in the frame (i.e., either postponed or anticipated), in order to improve the overall frame utilization. Other merging engine algorithms can be devised, operating different prioritization schemes, which can be classified into stateless and stateful. In the stateless case, the decision to shift a particular traffic type does not depend on previous allocations, while in the stateful case, the current decision takes into account previous actions, for example, considering how much additional latency a specific service has already experienced. The latter can be used to minimize the probability of breaking a specific service level agreement (SLA) (which, for example, might require that a certain latency value is achieved 99.99% of the time).

C. System Synchronization Requirements

Since a PON is a synchronous system with a precise framing structure, it is important that all the elements of the vDBA implementation follow the system clock rigorously. In particular, the data plane runs in a hardware platform with a precise 8 kHz clock (i.e., following the 125 μs frame duration).

In addition to hardware/software synchronization, the different vDBA algorithms also need to be in sync with the merging engine. Although not always critical for the basic functioning of the system, vDBA synchronization is important to maintain high efficiency in capacity allocation. Indeed, since the merging engine always sends the BMap in the same part of the downstream frame, if the virtual BMaps from the vDBAs arrive late, i.e., after the merging engine has already sent the BMap, they will not be included in that frame allocation, and that capacity will remain unallocated. Such synchronization becomes instead critical when the VNO needs to provide an

ultra-low-latency service, for example, C-RAN functional split, where the vDBA mechanism is in strict coordination with the BBU scheduler, as mentioned above. In this case, if the virtual BMap misses the transmission opportunity, the mobile service could suffer from excessive hybrid automatic repeat request (HARQ) re-transmission, momentarily decreasing the mobile system throughput.

3. vDBA STATE OF THE ART

This section summarizes the work we have carried out on PON virtualization, comparing it, where available, to that from other research institutions. Our initial work, reported in Section 3.A, showed how a novel frame-sharing approach could improve sharing efficiency while maintaining isolation across slices. Giving VNOs full control over capacity scheduling, however, requires novel incentive schemes to induce them to share their unused capacity, thus maintaining high overall PON efficiency. This work is described in Section 3.B. Finally, in Section 3.C, we provide details of our vDBA prototype implementation.

A. Intra-Frame Sharing Approach

The work we take as reference and use for baseline comparison is that proposed in [19], where each VNO is assigned one entire upstream frame, in a periodic round-robin fashion. Our intra-frame sharing approach is instead based on the possibility for each frame to be shared between VNOs. Our system starts by dividing the assured bandwidth portion of the total capacity across VNOs. In addition, the system re-assigns any leftover capacity, adopting approaches similar to those used in non-assured capacity allocation in regular PONs, i.e., proportionally to the value of assured capacity [15]. Our results, reported in [20], showed three main improvements over the approach presented in [19]. First, assigning one frame per VNO introduces a latency dependence on the number of VNOs, which instead disappears in our vDBA approach. Second, statically assigning a frame per VNO is not bandwidth efficient, as unused capacity from one VNO cannot be used by users belonging to other VNOs, a problem that is also solved by our approach.

Finally, the approach in [19] can allocate larger capacity to a specific VNO by increasing the ratio of frames assigned to that VNO. This, however, means that VNOs with higher capacity also gain lower latency, because having a higher number of frames available gives more opportunities to transmit. Thus, the VNO with less capacity suffers higher latency, meaning that the approach in [19] does not provide adequate isolation with respect to latency performance. Our vDBA has instead the ability to provide latency performance isolation with respect to VNO capacity allocation.

B. Capacity Sharing Through Auctions

Giving VNOs the ability to fully schedule their own capacity creates a new problem. If a VNO has leftover capacity on any given frame, it would have no gain in leaving this unallocated, as the merging engine would redistribute it to other VNOs that are likely to be its competitors. Thus, its best strategy would be to always send a fully allocated virtual BMap to the merging engine. This constitutes a problem, as it reduces the overall upstream PON efficiency.

To address this challenge, we have proposed [7] monetization of the excess capacity on PON slices, where, using an auction mechanism, the VNOs can trade their capacity in return for monetary compensation (shown in Fig. 2). Through rigorous theoretical proofs and market simulations in [21], we validated that the proposed market will incentivize the VNOs to trade their excess resources in a setting that is robust against market manipulation (i.e., where any attempt to manipulate the market by a player will not give any benefit to that player).

However, while our approach proved successful in maintaining high overall resource utilization, it worked under the assumption of an open-access architecture, where a fully trusted, independent central authority (e.g., the InP) is in charge of operating the market (i.e., bookkeeping, conducting the auction, settlements, etc.). This assumption may not always be valid for today's network ownership models, where often the incumbent operator owning the physical infrastructure is also a competing VNO. We have thus tackled the more general problem of an untrusted InP through the use of distributed consensus mechanisms (e.g., blockchain-based smart contracts) that do not rely on a central entity. Our initial results [22], implemented over Hyperledger Fabric [23], showed that we could achieve more than 160 transactions (i.e., auction rounds) per second on a system with eight virtual CPUs. This result already makes it possible to carry out a capacity auction every 6 ms, i.e., averaged over about 50 frames.

It is expected that transaction speed will further increase in the near future, as the Hyperledger Fabric community has recently set up a dedicated working group to address performance scalability [24]. In addition, the latest experiments [25] have shown that performances on the order of 20,000 transactions per second are already possible for some types of applications in Hyperledger Fabric.

C. vDBA System Implementation

As mentioned above, CORD was the first system implementing the idea of CO virtualization. The R-CORD release extended virtualization to the OLT, although slicing operated at the management level because virtualizing the data plane was considered difficult to achieve. However, the advanced development of Intel's Data Plane Development Kit (DPDK) software packet processing libraries has boosted the ability to implement telecommunication virtual network functions (VNFs) on general-purpose processors. For example, the DPDK has recently been used to deliver a full software implementation of the MAC layer of the DOCSIS 3.0 standard [26]. In addition, DPDK-enabled virtual packet switches have recently reached performance in excess of 70 million packets per second [27]. Considering its improved ability to operate with real-time algorithms requiring precise synchronization, we have worked in conjunction with Intel's DPDK development team to implement our virtual DBA platform prototype.

Our vDBA concept was demonstrated in [28] on a testbed incorporating a physical PON, a set of emulated ONUs, a

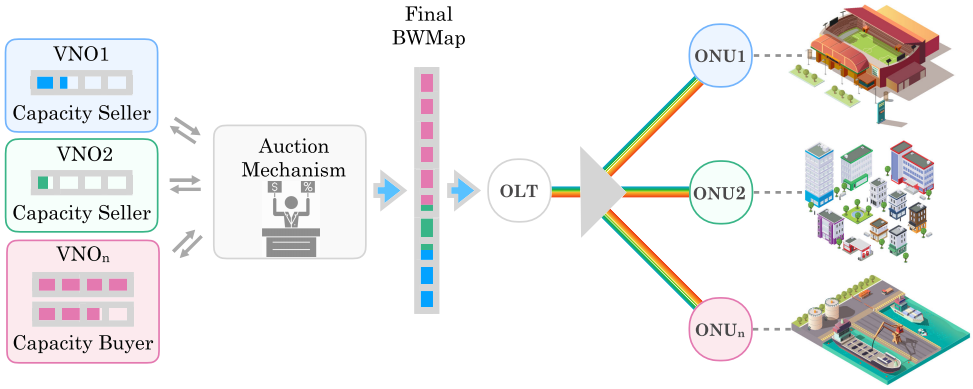

Fig. 2. Schematic of the PON auction mechanism.

traffic generator, and a server running the virtual data plane instances. The physical layer was implemented on Xilinx VC709 FPGA boards, operating at a symmetric 10 Gb/s line rate.

The PON hardware and software virtualization architecture is shown in Fig. 3. One of the servers hosted the merging engine and the vDBA functions for the VNOs, implemented as VNFs. Due to the real-time critical nature of receiving and transmitting status report messages from the ONUs and BMap data, our latest implementation (reported in [29,30]) made advanced use of the DPDK toolkit, further described below, to optimize the packet transfer through the physical host, to and from the VNFs. To the best of our knowledge, the only other implementation of a DBA in software is the work from NTT. In [31], the authors present a software-based implementation of an Ethernet passive optical network (EPON) system, showing line-rate throughput performance for 1G EPON. In [9], their architecture was extended to a flexible access system architecture (FASA), where the system can dynamically change the DBA running in the PON. This however, runs only one DBA algorithm at a time, unlike our proposed vDBA, which allows multiple VNOs and services to operate in parallel.

The remainder of this section describes some of the challenges incurred when implementing the DBA scheduling algorithm in software, together with our proposed solutions.

1. *Time Critical Issues for DBA Virtualization*

Since PONs run synchronously, one of the first problems we addressed was that of providing a precise 8 kHz clock to the hardware/software integrated system. The 8 kHz clock is derived from the hardware clock, typically between 2 and 3 GHz, by querying the DPDK rte_get_time_hz() function. By counting down a specific number of hardware clock cycles using the rte_rdtsc() function, a tick of the 8 kHz clock is generated. To compensate for variations in the frequency of the hardware clock caused by temperature and environmental effects, we call on the rte_timer_manage() function to calibrate the software clock against the hardware clock every 10 ms. While the vDBA is generally constructed to run in a non-blocking and freewheeling manner, we make use of one system-level semaphore to distribute the system 8 kHz clock to all the applications that are synchronized to run in tandem. These include the VNO and the downstream merging engine applications. The clock application sets the semaphore using the tick() function, while each blocking application reads it using the tock() function.

When the DBA is architected in a traditional manner, DBRu packets are accepted into the system through a packet driver running in kernel space and passed to the vDBA application running in userspace. A number of issues arise with this approach, which leads to slow packet processing. First, the hardware interface may generate an interrupt to the kernel for each packet processed in the kernel network stack. Second, the packet must be copied from a data structure in kernel space to a data structure in userspace, which requires CPU processing. Third, when the vDBA application is blocked awaiting the input of each packet, physical resources such as memory and locks are exclusively locked. The vDBA application must process many tens of thousands of small DBRu packets every second, resulting in a large number of interrupts, packet copying, and locking/unlocking of resources. This leads to threshing of the L1/L2 cache and memory and high latency in packet processing, on a system otherwise busy with the calculation of the DBA BMaps.

We make use of the DPDK open-source software to overcome the issues of latency and performance degradation and thereby achieve packet processing at line rates. We make extensive use of DPDK's environment abstraction layer (EAL) and the four core components: ring manager (librte_ring), memory pool manager (librte_mempool), network packet buffer management (librte_mbuf), and timer manager (librte_timer). DPDK allows the vDBA application to be distributed across the cores of a multi-core x86 CPU.

2. *Interface Between Hardware and Software (I/O Virtualization)*

In the hypervisor approach [i.e., used to support virtual machines (VMs)], I/O virtualization is accomplished using the hardware emulation layer under the control of the hypervisor, whereas in Linux containerization (LXC), this is achieved through device mapping. As a result, containers have easier direct access to the hardware than VMs, since they operate at the host OS level. In VMs, additional techniques might be needed (e.g., paravirtualization or CPU-assisted virtualization)

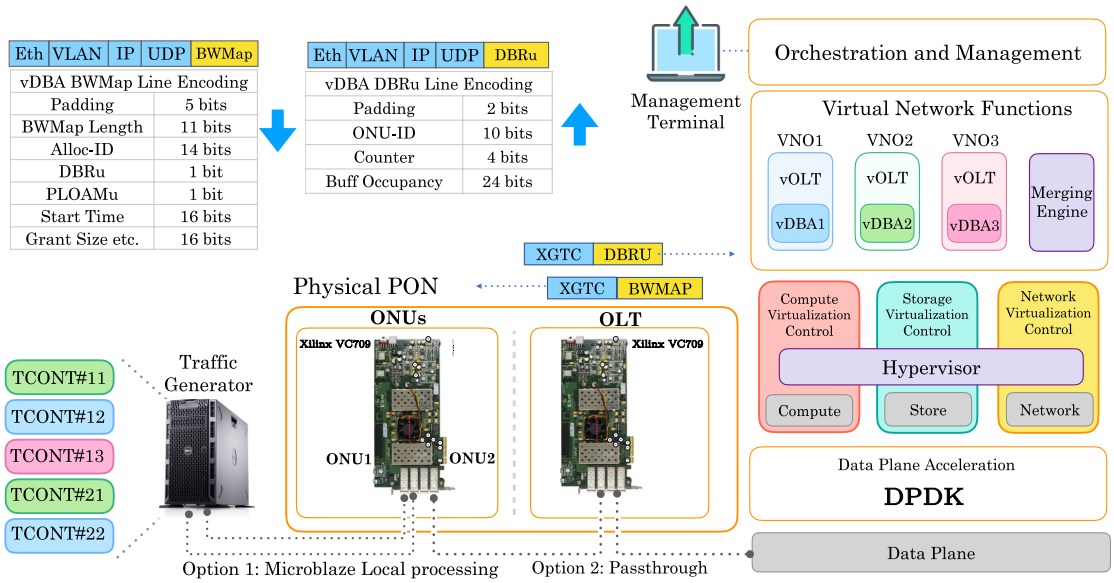

Fig. 3. Overall architecture of the PON system prototype running the virtual DBA.

to provide direct or fast access to the hardware. We used the subset of the DPDK data plane acceleration toolkit to reduce I/O virtualization overhead and accelerate packet I/O as well as improve packet processing performance within the vDBA. In particular, we used a userspace device driver to provide fast access to the network interface card, which enabled the vDBA within each VNO to send and receive packets from the network efficiently.

3. *Interface between the vOLT and vDBA*

The virtual OLT (vOLT) software and vDBA run on the same machine and communicate with each other using shared physical resources such as memory, as shown in Fig. 4. We can thus avail the DPDK toolkit to optimize the interface between the vOLT and vDBA, by incorporating techniques such as zero copy, batch processing of the packets, buffer allocations, and interrupt-less I/O. In order to transfer DBRus and BMap between the main vOLT and vDBA functions on the host, we assign DPDK software rings at the inputs and outputs of the main vDBA functions. A DPDK software ring has the characteristics of a lockless ring buffer where producers enqueue and consumers dequeue memory buffers (mbufs), respectively. It is transaction safe; however, the use of locks is minimized. Locks can slow down enqueuing and dequeuing data on a busy system and can give rise to mutex (mutual exclusion) deadlocks in poorly designed applications, particularly under load. In order to further avail the DPDK's optimized data handling mechanisms, we mapped the key DBA data transfer structures, i.e., DBRus and BMaps, onto data structures provided by the DPDK (mbufs). The DPDK mbuf data structure is constructed and managed in such a way as to minimize copying and locking of data, particularly during periods of high processor utilization. The mbufs are grouped into two groups or pools, appropriate to the dimensions and volume of DBRus and BMaps being processed in a particular time interval. We

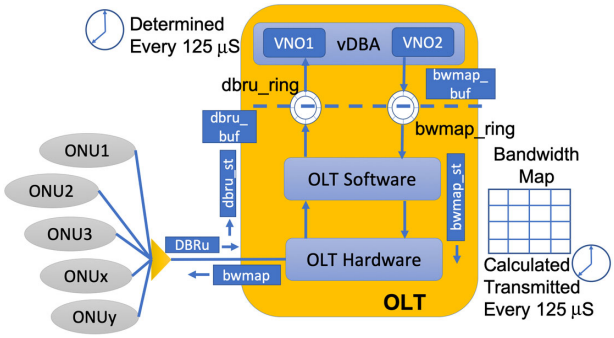

Fig. 4. Interface between a virtual OLT and vDBA.

minimize the amount of processing necessary during the runtime phase of the DBA algorithm, so we offload operations such as the preparation of pools and mbufs to the preliminary or setup phase. This allows data structures to be easily reused or reallocated in runtime, rather than having to be created or deleted on demand, which can be both expensive and an unnecessary waste of processing resources.

This section has described our proposed vDBA architecture, summarizing the work carried out so far. The next section describes a novel merging engine algorithm and reports simulation results showing the trade-off we achieve between accuracy of data allocation (i.e., in order to assure low latency) and the efficiency of PON capacity utilization.

4. MERGING ENGINE

This section reports the details of a specific merging engine algorithm we have developed in order to implement the strict prioritization scheme described in Section 2.B. The algorithm is named the priority-based merging algorithm (PBMA) and the pseudo-code is reported in Algorithm 1.

Algorithm 1. Priority-Based Merging Algorithm

```
1:  Input TC_p(p)∈(0, 4), VNO_i(i)∈(int), R_j∈TC_p(j)∈(int)
2:  TC_p ← trafficclass
3:  R_j ← Requests
4:  coll ← Collision
5:  plc ← Placement
6:  for all TC_p do
7:    p ← start from highest priority
8:    for all VNO_i do
9:      Traverse all R_j sequentially and do the following;
10:     coll = Find coll();
11:     if coll is TRUE then
12:       (coll R_j will be returned)
13:     else
14:       (0 will be returned)
15:     end if
16:     if coll > 0 then
17:       plc = Find plc ();
18:       if coll R_j is either of less TC_p or has a greater start time
          then
19:         (plc is TRUE)
20:       else
21:         (plc is FALSE)
22:       end if
23:     end if
24:     if coll == 0 OR plc == TRUE then
25:       Allocate_request_in_final bandwidth map
26:     else
27:       Mark_request_unallocated
28:     end if
29:   end for
30:   for all VNO_i do
31:     Traverse all unallocated R_j and do the following;
32:     slots_req = Cal_req_time slots();
33:     slots_available = find next empty fragment with
        slots_req();
34:     if slots_available == TRUE then
35:       Allocate_request_in_final bandwidth map
36:     else
37:       Mark_request_rejected
38:     end if
39:   end for
40: end for
```

In the algorithm, all operations are repeated for each traffic class, starting from the highest priority p (line 6). For a particular traffic class, all requests belonging to it are considered sequentially for each VNO (line 7 and line 8). For each incoming virtual BMap, the algorithm checks each burst allocation, determining whether the first requested allocation is in conflict with any other virtual BMap from other VNOs (line 9). In case a collision exists, if the colliding requests are of lower priority, then the allocation will remain. If the conflicting request belongs to the same traffic class, priority will be given to the request whose start time is earlier. If the colliding requests are of higher priority, then this request will be shifted forward until enough free slots are available or else it is possible to preempt allocations with lower priority. If this is not possible, then the request will be temporarily marked as unallocated for this frame. As an example, if VNO1 has scheduled its priority 3 request from slot 20 to 30 and VNO2 has scheduled its priority 3 request from slot 24 to 32, while merging the two allocations, the PBMA will allocate VNO1's request from 20 to 30 and then VNO2's request from 30 to 38. In this case, the request of VNO2 is shifted by six slots in time. On the other hand, had the allocations been merged in the opposite manner, the request of VNO1 would have been shifted by 12 slots.

The same operations are repeated for all other requests of priority p. After this, before moving to the next lower priority, the algorithm will consider the requests with priority p, which were included in the pool of unallocated requests (line 30). For each VNO, all its unallocated requests of priority p are examined one by one, and the merging engine tries to allocate them by shifting the requests to the next available fragment of empty slots fulfilling the requirement (lines 31–33). If an empty fragment with required capacity is found, the request is allocated in the final merged BMap; otherwise it is marked as rejected. The same operations are repeated for all priority p in TC_p.

A. Simulation Results

In our simulation, we consider two VNOs, sharing a 10 gigabit symmetric PON (XGS-PON) upstream frame, which is divided into 1152 slots, each of 135 bytes with approximate duration of 0.11 μs. The slot subdivision is somewhat arbitrary, and should reflect the tradeoff between granularity of grant allocation (larger number of slots preferred) and computational complexity (smaller number preferred). In addition, the duration of one slot determines, in our implementation, the minimum guard interval, as we always leave at least one empty slot between grant allocations. According to the XGS-PON standard [32], the minimum suggested guard interval is 64 bytes, while the suggested largest (i.e., worst case) interval is 256 bytes. Our minimum grant size was chosen to be close to twice this minimum and half this maximum value.

1. Uniform Traffic Distribution

The BMaps of both VNOs are randomly generated using a uniform distribution with different maximum grant sizes, in order to evaluate the dependency between performance and grant size. Grants are also uniformly distributed across traffic classes (except for some of the plots, as described below). We take into consideration three different traffic distribution scenarios, with varying average grant size and interval (both measured in terms of number of slots). The interval is selected so that the grant request is balanced across the frame. For scenario 1, the min and max values of the grant requests are uniformly distributed between one and 10 slots; for scenario 2, between one and 50 slots; and for scenario 3, between one and 100 slots. The load per VNO is then generated as a percentage of the upstream frame duration. In addition, in order to consider burst overheads, we always leave at least one unallocated slot between grants in the final BMap. Since in this study we are interested in analyzing the ability of the merging engine to allocate the incoming traffic and the latency distribution across the different priority classes, we choose to drop a grant request

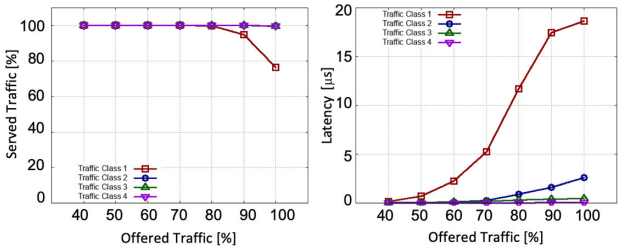

Fig. 5. Traffic scenario 1: (a) served traffic and (b) latency versus offered traffic.

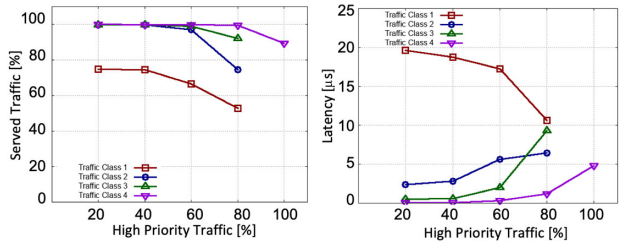

Fig. 6. Traffic scenario 1: (a) served traffic and (b) latency versus high-priority traffic.

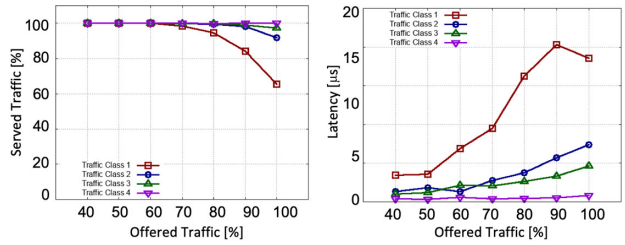

Fig. 7. Traffic scenario 2: (a) served traffic and (b) latency versus offered traffic.

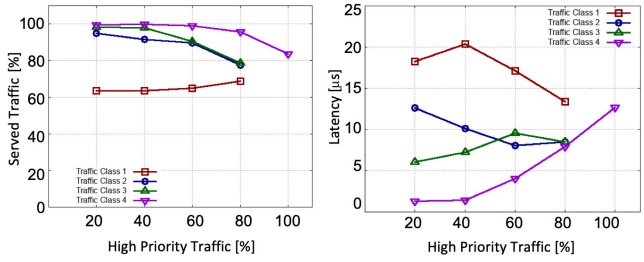

Fig. 8. Traffic scenario 2: (a) served traffic and (b) latency versus high-priority traffic.

if this cannot be accommodated within the frame (i.e., we do not move its allocation into the next frame).

We first evaluate the performance of our merging algorithm by varying the combined offered loads of both VNOs. All results are reported with a 99% confidence interval.

Figure 5(a) shows the performance of the PBMA in terms of offered versus served traffic, for each traffic class. Here the load is equally distributed across the four priority classes. As expected, with the increase in offered traffic, the served traffic of the lowest-priority type starts to decrease. However, since the maximum grant size is only on the order of 1% of the frame size, the system can accommodate all traffic of priority 4 to 2, while the latency introduced by the process remains below 3 µs for these classes [shown in Fig. 5(b)].

For an InP, it is also important to understand how much high-priority traffic the system can accommodate before violating a given latency requirement. Thus, in Fig. 6(a), we report the percentage of traffic the PON can serve when the offered traffic is at 100%, but the highest-priority traffic is increased from taking up 20% of the overall traffic, up to 100%. In this case, the remaining traffic is equally distributed across the other three classes (and when the highest-priority class is at 100%, classes 3, 2, and 1 have no traffic). The figure suggests that as priority class 4 traffic increases, all lower-priority classes suffer both some drop and additional latency [shown in Fig. 6(b)]. However, traffic with priority 4 shows no drop in up to 80% of traffic volume, for which case, the additional latency is on the order of 1 µs. In Fig. 6(b), we can also notice that the latency of low-priority class 1 decreases with the increase in high-priority traffic, which can appear counterintuitive. This is due to the fact that class 1 is the first to be dropped and as a grant is dropped, its latency is not accounted for in the plot [the amount of dropped grants can be seen in Fig. 6(a)].

We then increase the average grant size in the second scenario, so that the maximum value is about 5% of the frame duration. As we can see in Fig. 7(a), priorities 4 and 3 remain unaffected, while some grants are dropped for the lower-priority classes. In this scenario, latency remains negligible for traffic class 4 [see Fig. 7(b)], while it increases to 4 µs for class 3.

As we increase the highest-priority traffic, for scenario 2, now we can see that up to about 60% of high-priority traffic can be accommodated without grant loss [Fig. 8(a)], in which case, the latency increase remains around 2 µs [Fig. 8(b)].

We finally increase the maximum grant size, in the third scenario, up to 10% of the frame size. It should be noted that this can be considered large for a single grant, especially for low-latency applications which could transmit more than once per frame. When traffic is balanced across the classes [Fig. 9(a)], the two highest priorities can still perform well up to the full PON rate, although the latency increases further for all classes, except the highest [shown in Fig. 9(b)]. In Fig. 9(a), we notice some minor fluctuations in traffic classes 4 and 3, which are due to statistical effects caused by the fact that in this analysis, we do not move unallocated grants to the next frame.

Finally, as we increase the highest-priority traffic also for scenario 3 [Fig. 10(a)], we see that still 60% of the highest-priority class can be accommodated without loss, with a slight increase in latency to around 4 µs [Fig. 10(b)].

As mentioned above, we currently drop all grants if they cannot be accommodated, as we aim to analyze the pure latency performance of the proposed algorithm. In a practical system, it would be preferable to carry unallocated high-priority grants into the beginning of the next frame.

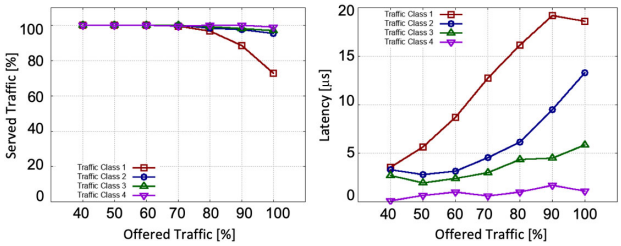

Fig. 9. Traffic scenario 3: (a) served traffic and (b) latency versus offered traffic.

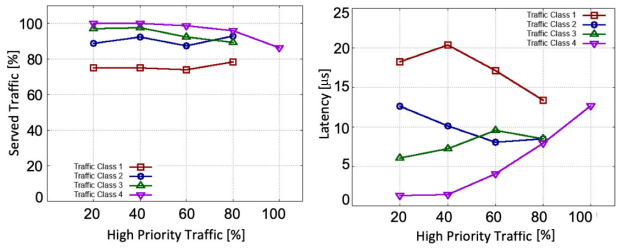

Fig. 10. Traffic scenario 3: (a) served traffic and (b) latency versus high-priority traffic.

2. Poisson and Self-Similar Traffic Distributions

While the use of the uniform distribution has allowed us to broadly assess and compare different traffic scenarios, in this section, we provide further comparison with two other traffic distributions: Poisson and self-similar. This allows us to investigate and quantify how the merging engine performance might vary for different traffic profiles, for example, adding burstiness to the traffic. For the Poisson distribution, we have selected a value for lambda equal to the average traffic as measured for the uniform distribution case. For the self-similar distribution, we have selected a Hurst value of 0.55, as this denotes a burtsiness characteristic of Internet traffic and enables reproducing the long tailed distributions used for our grant size ranges (i.e., from 1 to 10 up to 1 to 200 min-max slots per grant). Figure 11 provides the comparison among uniform, Poisson, and self-similar traffic for scenario 1, for traffic types 3 and 4. We can see that independent of the distribution, there is no traffic drop [Fig. 11(a)], and the difference in latency [Fig. 11(b)] is negligible for both high-priority types 4 and 3, meaning that the merging algorithm operates efficiently for high-priority traffic, independent of the traffic profile.

The situation is different for the lower-priority traffic, 1 and 2, as can be seen in Fig. 12. While the drop rate is somewhat similar, the latency has higher variation, with the uniform traffic presenting lower latency compared to Poisson and self-similar traffic. This is a result of the overall higher burstiness of the traffic, which, while not affecting the other priorities, is all borne by the lower-priority class. In practice, this does not constitute an issue for the merging engine, because priority 1 is the lowest, which would normally be associated with a service with no strict latency constraints.

If we look at scenario 3, we see that the difference in performance is more pronounced for low priorities 1 and 2 (Fig. 13). While the drop rate is similar across the distributions,

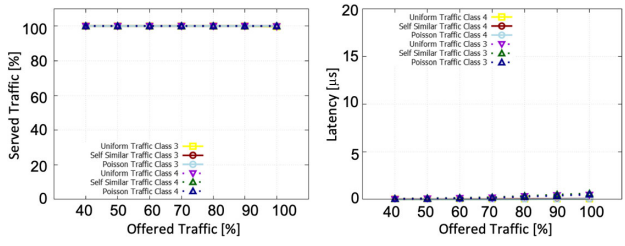

Fig. 11. Traffic scenario 1: (a) served traffic and (b) latency versus offered traffic for high traffic priority, for uniform, Poisson, and self-similar distributions.

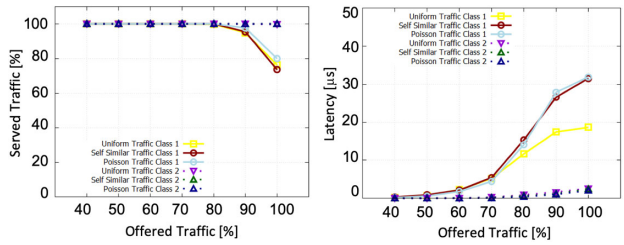

Fig. 12. Traffic scenario 1: (a) served traffic and (b) latency versus offered traffic for low traffic priority, for uniform, Poisson, and self-similar distributions.

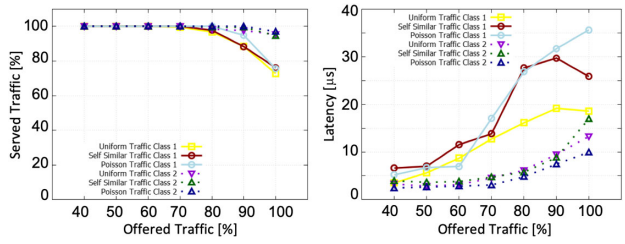

Fig. 13. Traffic scenario 3: (a) served traffic and (b) latency versus offered traffic for low traffic priority, for uniform, Poisson, and self-similar distributions.

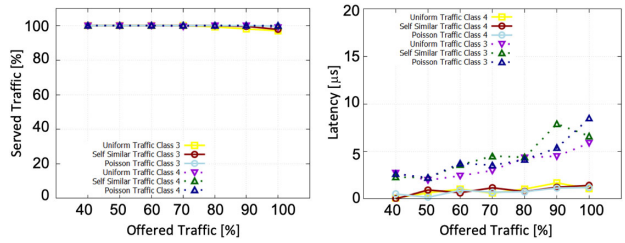

Fig. 14. Traffic scenario 3: (a) served traffic and (b) latency versus offered traffic for high traffic priority, for uniform, Poisson, and self-similar distributions.

the latency has a higher variation for both priorities 2 and 1, with the uniform traffic again showing lower latency on the lowest-priority class. However, similar to scenario 1, for the higher priorities (Fig. 14), the latency is not dependent on the type of traffic considered for priority 4 and only slightly dependent for priority 3.

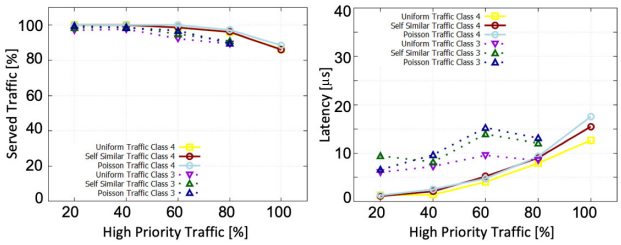

Fig. 15. Traffic scenario 3: (a) served traffic and (b) latency versus high-priority traffic, for traffic priorities 3 and 4, for uniform, Poisson, and self-similar distributions.

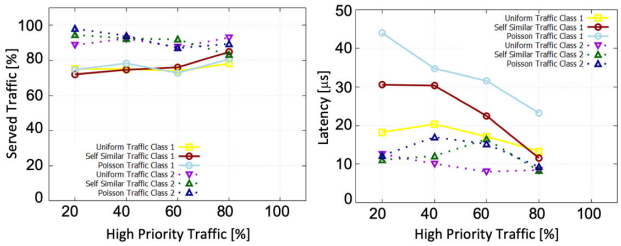

Fig. 16. Traffic scenario 3: (a) served traffic and (b) latency versus high-priority traffic, for traffic priorities 1 and 2, for uniform, Poisson, and self-similar distributions.

Finally, we briefly report the results for the case where high-priority traffic 4 increases from 20% up to take up the entire PON capacity. Figures 15 and 16 report the results, respectively, for the high-priority and low-priority classes, for the third scenario (i.e., the most challenging of the three presented). For the high-priority traffic (Fig. 15), we see that the drop rate is rather independent of the distribution considered, with some minimal difference in terms of latency for priority 3, noticeable at 60% high-priority traffic level. For the highest priority 4, we notice that there is no drop in up to 60% of traffic (with the exception of a small loss for the self-similar traffic case), while the delay difference becomes somewhat noticeable only when all traffic is the highest priority 4 (i.e., at 100% value on the x axis). For the lower-priority classes (Fig. 16), we see again that the uniform distribution is the least affected in terms of delay, although the drop rates remain similar.

In conclusion, this analysis has shown that in multi-tenant scenarios, if low latency is a requirement, allocating smaller grant sizes (i.e., for scenario 1) gives better performance, despite the increase in burst overhead. This is also in line with typical traffic patterns of low-latency applications, where transmitting smaller packets reduces packetization delay. In addition, we have seen that different traffic distributions affect only low-priority traffic, while the performance remains mostly independent for high-priority traffic. Intuitively, even with bursty distributions, the merging algorithm is capable of allocating grant allocations without increasing the latency of high-priority traffic, because it is able to move all latency increase to the lower-priority classes, where strict latency is not a requirement.

5. CONCLUSION

This paper has introduced the concept of vDBA in virtualized PON systems, which we use to enable support for multi-tenancy and multi-service in access networks. After providing details on both the architecture and system prototype implementation, we have described a specific algorithm for merging virtual BMaps from multiple VNOs. Our work has analyzed the effects that multiple traffic sources across two VNOs have on the additional latency that is potentially introduced during the merging process of vDBAs. We first show that by applying strict prioritization between classes it is possible to limit the impact the overall traffic load has on the highest-priority classes. In addition, we quantify, for the types of traffic scenarios considered, the percentage of highest-priority traffic that an InP should allow in the PON, for a given latency target.

An important observation we can make from our analysis is that while having larger bursts from fewer ONUs is typically a good strategy in residential PON applications, here our results show that in multi-tenancy scenarios, it is more efficient to have smaller grants, as these can be merged more effectively into the common BMap, also leading to lower latency. Although there is a capacity loss due to the increased overhead, the merging algorithm suffers from lower additional latency and is able to accommodate a larger number of grants.

Our future work will provide a more in-depth study on how this relation changes for different types of traffic mix and will also propose stateful types of merging engine algorithms. We will also examine the effect that breaking larger grants into smaller sizes and carrying some unallocated grants to the next frame has on the overall system performance.

Funding. Science Foundation Ireland (14/IA/252, 13/RC/2077, 12/RC/2276P2).

REFERENCES

1. R. Gaudino, R. Giuliano, F. Mazzenga, L. Valcarenghi, and F. Vatalaro, "Unbundling in current broadband and next-generation ultra-broadband access networks," Fiber Integr. Opt. **33**, 129–148 (2014).
2. N. Afraz, F. Slyne, H. Gill, and M. Ruffini, "Evolution of access network sharing and its role in 5G networks," Appl. Sci. **9**, 4566 (2019).
3. S. Beardsley and L. Enriquez, "Creating a fiber future: the regulatory challenge," Tech. Rep. (McKinsey & Co. Inc., 2011).
4. 5G-PPP, The 5G infrastructure public private partnership: the next generation of communication networks and services (2015).
5. L. Peterson, A. Al-Shabibi, T. Anshutz, S. Baker, A. Bavier, S. Das, J. Hart, G. Palukar, and W. Snow, "Central office re-architected as a data center," IEEE Commun. Mag. **54**(10), 96–101 (2016).
6. "Seba white box access: XGS-PON and beyond," 2019, https://www.opennetworking.org/wp-content/uploads/2019/09/3pm-Jeff-Catlin-SEBA-White-Box-Access.pdf.
7. N. Afraz, A. Elrasad, and M. Ruffini, "DBA capacity auctions to enhance resource sharing across virtual network operators in multi-tenant pons," in *Optical Fiber Communication Conference and Exposition (OFC)* (2018).
8. F. Slyne, B. Cornaglia, M. Boselli, and M. Ruffini, "3-stage hierarchical quality of service for multi-tenant passive optical network," in *23rd Conference on Optical Network Design and Modeling (ONDM)* (2019).
9. K. Asaka, H. Ujikawa, J. I. Kani, and A. Otaka, "Flexible access system architecture (FASA)," in *Optical Fiber Communication Conference* (Optical Society of America, 2018), paper Tu3L.7.

10. F. Slyne, B. Cornaglia, M. Boselli, and M. Ruffini, "Single-stage scheduler for accurate QoS delivery in virtualised multi-tenant passive optical networks," in *European Conference on Optical Communications (ECOC)* (2019).
11. M. Ruffini, A. Elrasad, and N. Afraz, "System and method for dynamic bandwidth assignment (DBA) virtualization in a multi-tenant passive optical network," Patent, International (PCT) Application No. PCT/EP2018/056767 (Sept. 20, 2018).
12. "Functional model for PON abstraction interface," TR-402, (Standard BroadBand Forum (BBF), 2018).
13. "40-gigabit-capable passive optical networks (NG-PON2): transmission convergence layer specification amendment 1," ITU-T Recommendation G.989.3am1, 2016.
14. N. Afraz, F. Slyne, and M. Ruffini, "Full PON virtulisation supporting multi-tenancy beyond 5G," in *OSA Advanced Photonics Congress (AP) 2019 (IPR, Networks, NOMA, SPPCom, PVLED)* (Optical Society of America, 2019), paper NeT2D.2.
15. "10-gigabit-capable passive optical networks (XG-PON): transmission convergence (TC) layer specification," ITU-T Recommendation G987.3, 2010.
16. "Common public radio interface: ECPRI interface specification v1.1," Standard (2018).
17. T. Tashiro, S. Kuwano, J. Terada, T. Kawamura, N. Tanaka, S. Shigematsu, and N. Yoshimoto, "A novel DBA scheme for TDM-PON based mobile fronthaul," in *Optical Fiber Communication Conference* (Optical Society of America, 2014), paper Tu3F.3.
18. H. Uzawa, H. Nomura, T. Shimada, D. Hisano, K. Miyamoto, Y. Nakayama, K. Takahashi, J. Terada, and A. Otaka, "Practical mobile-DBA scheme considering data arrival period for 5G mobile fronthaul with TDM-PON," in *European Conference on Optical Communication (ECOC)* (2017), pp. 1–3.
19. C. Li, W. Guo, W. Wang, W. Hu, and M. Xia, "Bandwidth resource sharing on the XGPON transmission convergence layer in a multi-operator scenario," J. Opt. Commun. Netw. **8**, 835–843 (2016).
20. A. Elrasad and M. Ruffini, "Frame level sharing for DBA virtualization in multi-tenant PONs," in *International Conference on Optical Network Design and Modeling (ONDM)* (2017), pp. 1–6.
21. N. Afraz and M. Ruffini, "A sharing platform for multi-tenant PONs," J. Lightwave Technol. **36**, 5413–5423 (2018).
22. N. Afraz and M. Ruffini, "A distributed bilateral resource market mechanism for future telecommunications networks," in *IEEE Global Communications Conference (GLOBECOM)* (2019).
23. "Hyperledger Fabric," 2019, https://www.hyperledger.org/projects/fabric.
24. "Performance and scale working group," https://wiki.hyperledger.org/display/PSWG/Performance+and+Scale+Working+Group.
25. C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "FastFabric: scaling Hyperledger Fabric to 20,000 transactions per second," in *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)* (2019).
26. M. O'Hanlon and B. Ryan, "DPDK Intel NIC performance report release 18.02" (Intel, 2018).
27. "DPDK intel NIC performance report release 18.02," Report (Intel, 2018).
28. F. Slyne, A. Elrasad, C. Bluemm, and M. Ruffini, "Demonstration of real time VNF implementation of OLT with virtual DBA for sliceable multi-tenant PONs," in *Optical Fiber Communication Conference and Exposition (OFC)* (2018).
29. F. Slyne, R. Giller, J. Singh, and M. Ruffini, "Experimental demonstration of DPDK optimised VNF implementation of virtual DBA in a multi-tenant PON," in *European Conference on Optical Communication (ECOC)* (2018).
30. M. Ruffini and F. Slyne, "Moving the network to the cloud: the cloud central office revolution and its implications for the optical layer," J. Lightwave Technol. **37**, 1706–1716 (2019).
31. K. Nishimoto, M. Tadokoro, T. Mochida, T. Yamada, T. Tanaka, A. Takeda, and T. Inoue, "Implementation of software-based EPON-OLT and performance evaluation," IEICE Commun. Express **6**, 467–472 (2017).
32. "10-gigabit-capable symmetric passive optical network (XGS-PON)," ITU-T Recommendation G.9807.1, 2016.