Capacity sharing approaches in multi-tenant, multi-service PONs for low-latency fronthaul applications based on cooperative-DBA

Arsalan Ahmad[1,2], Frank Slyne[1], Sanwal Zeb[1], Abdul Wahab[2], Rana Azhar Khan[2], Marco Ruffini[1]

CONNECT research centre, The University of Dublin, Trinity College, Ireland
National University of Sciences and Technology (NUST), Pakistan
ahmadar@tcd.ie, marco.ruffini@tcd.ie

Abstract: We propose and compare algorithms to allocate upstream PON capacity, where multiple virtual operators generate independent frame-level allocation over shared infrastructure. Our fragmentation-based approach shows the ability to limit latency increase to a few microseconds. © 2020 The Author(s)

OCIS codes: (060.2330) Fiber optics communications, (060.4256) Networks optimization.

1. Introduction

Continuous growth in Internet traffic, together with the prospect of densified 5G mobile infrastructure and the new high-bandwidth services that it will enable, means that network operators need to continuously plan ahead the provisioning or upgrade of their access networks. Although Passive Optical Networks (PONs) are a future-proof and low-cost solution to deploy fibre-based access networks, large scale deployment is still hindered by the high capital expenditures required upfront. Similarly to what has happened in the mobile communication domain, thus fixed network operators have started designing architectural models to enable sharing network resources across multiple virtual operators and diverse services [1]. While sharing mechanisms that operate at higher layers have already been deployed (i.e., Virtual Unbundled Local Access (VULA) and next generation bitstream), the type of access services that PONs are required to support for 5G networks and beyond will need low latency and strict isolation between network slices. This means that Virtual Network Operators (VNOs) will require the ability to schedule their capacity within each transmission frame (e.g, at the microsecond scale), over a shared infrastructure. For example, work in [2] has shown that fronthaul operations can be supported by PONs in the upstream, if multiple bursts per frame are allowed. In a shared PON environment, VNOs will require the ability to directly schedule upstream transmission within each frame [3].

While PON virtualisation solutions have recently appeared, for example in the Residential-CORD (R-CORD) [4] open source platform, these typically provide resource slicing at the management level, without virtualising the Dynamic Bandwidth Allocation (DBA) protocol. This does not allow VNOs to control the frame-level scheduling required to support services like 5G fronthaul.

We have thus proposed the concept of Virtual DBA (vDBA) [5], recently standardised by the BroadBand Forum (BBF) in [6], to enable different VNOs schedule their frame-level capacity, using dedicated DBA algorithms that produce independent virtual Bandwidth Maps (BMaps) with their preferred allocation. These virtual BMaps are then merged together by a Merging Engine that produces the final BMap that is physically transmitted to all Optical Network Units (ONUs). These operations are summarised in Fig. 1.a. In this paper, we propose two novel stateless algorithms for the merging engine, both based on the use of class priority, and analyze their performance in terms of efficiency of capacity allocation and latency.

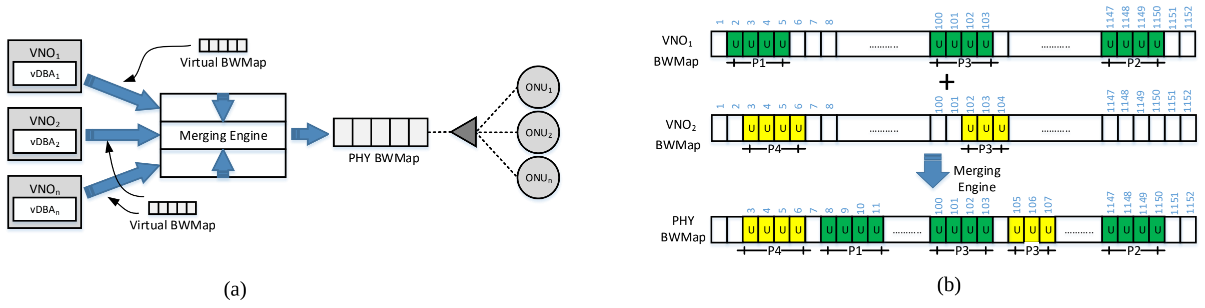


Fig. 1. Architectural design of the virtual DBA mechanism

2. The Merging Engine Architecture and Algorithms Implementation

The Merging Engine carries out the difficult task to merge multiple virtual bandwidth maps, which are sent independently by the VNOs' DBAs, into one final allocation, thus solving all conflicts between the virtual BMaps (i.e., where these are attempting to allocate capacity at the same time). The principle, which operates on the concept of class prioritisation, is that when a collision occurs in the scheduling, the merging engine will shift the allocation with lower priority, which will thus suffer increase in latency. If two conflicting allocations belong to the same class, then the algorithm will shift the one that would result in lower overall delay (shown in Fig. 1.b). In this work we consider 4 different strict priority classes (4 being the highest), where traffic from lower priority is always shifted every time it collides with a higher priority. In addition, we assume that classes 4 and 3 operate ultra-low latency fronthaul services using cooperative DBA, standardised in [7], where the Dynamic Bandwidth Report upstream (DBRu) mechanism is bypassed, and the scheduler of the base station informs the DBA algorithm in the Optical Line Terminal (OLT) directly about the required scheduling information. Since this mechanism operates just in time, it is not possible for the merging engine to anticipate the virtual BMap allocation, as the data to be transmitted would not be yet available at the OLT. Classes 1 and 2 operate instead over the normal DBRu report/grant scheme, where packets are buffered at the ONU for some time before the BMap is received. For this reason, allocations for classes 1 and 2 can also be anticipated within a given frame by the merging engine.

```
Algorithm 1: Priority Based Merging Algorithm (PBMA)
Input: $TC_p(p) \in (0,4), VNO_i(i) \in (int), R_j \in TC_p(j) \in (int)$;
$TC_p \leftarrow$ Traffic class, $R_j \leftarrow$ Requests, $coll \leftarrow$ Collision, $plc \leftarrow$ Placement;
foreach $TC_p$ do
    $p \leftarrow$ start with highest priority;
    foreach $VNO_i$ do
        Traverse all $R_j$ sequentially and do the following;
        $Coll =$ Find $coll()$;
        if $current\ priority == colliding\ request\ priority$ then
            Select request with lower start time;
        else
            Select $max(current.priority, colliding\_request.priority)$;
        Place the selected request in the merged output;
    foreach $VNO_i$ do
        Traverse all unallocated $R_j$ and do the following;
        if $R_j$ $can\ be\ anticipated\ or\ delayed$ then
            added to the merged output;
        else
            add them in $left\_out\_segments$;
```

(a) Priority Based Merging Algorithm (PBMA)

```
Algorithm 2: Fragmentation Aware Merging Algorithm (FAMA)
Input: $TC_p(p) \in (0,4), VNO_i(i) \in (int), R_j \in TC_p(j) \in (int)$;
$TC_p \leftarrow$ Traffic class, $R_j \leftarrow$ Requests, $coll \leftarrow$ Collision, $merged \leftarrow$ Output;
Execution;
while $current\_time \leq frame\_size(1152)$ do
    foreach $VNO_i$ do
        $top\_segment ==$Traverse all requests $R_j$;
        if $R_j.start\_time \leq current\_time$ then
            return $R_j$;
        else
            $find(R_j$ with priority $< 3)$ return $R_j$;
        $coll = check\_coll(top\_segments)$;
        if $coll = True\ and\ coll > 0$ then
            if $len(request.start\_time \leq current\_time) == 1$ then
                select request
            else
                Select $max(Rj.priority)$ or $min(Rj.start_time)$;
        $merged.output \leftarrow selected.request$;
        $current\_time \leftarrow current\_time + request.job\_size$;
```

(b) Fragmentation Aware Merging Algorithm (FAMA)

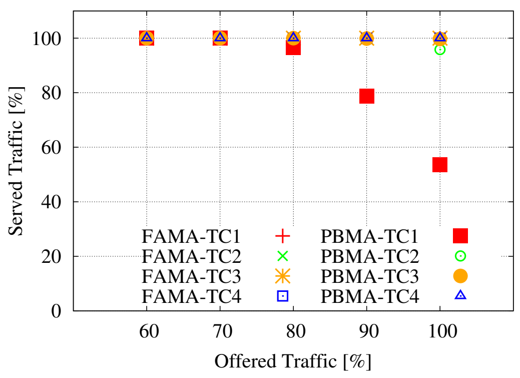
Fig. 2. Pseudo-code for the two Merging Engine algorithms considered in this work

In this paper we define two merging engine algorithms and analyse their difference in performance. The algorithms are named Priority Based Merging Algorithm (PBMA) and Fragmentation Aware Merging Algorithm (FAMA) and their pseudo-codes are reported in Fig. 2.a and 2.b, respectively. In PBMA, all requests are traversed sequentially for each traffic class, starting from the highest priority p. Conflicts are resolved by anticipating and/or postponing allocations on the basis of priority p and in case of similar priority, the allocation having an earlier start time is preferred (as this reduces the overall amount of latency introduced). On the other hand, FAMA works by considering allocations from the start of the frame till end, with all priorities examined at the same time. FAMA works by comparing the VNO's requests whose start time is less than the current time of the merged output. It first compares the priority and in case of similar priority the request with lower start time is selected. If there are no requests with start time less than the current merged output time, FAMA considers the requests that can be anticipated (i.e., those belonging to classes 1 & 2), for each VNO.

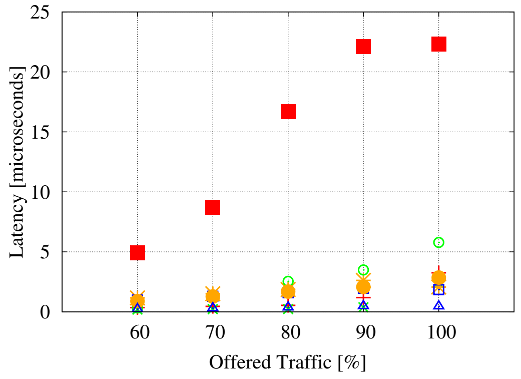
3. Simulation results

In our simulations we consider two VNOs sharing a 10G XGS-PON upstream frame, which is divided into 1152 slots, each of 135 bytes with approximate duration of $0.11\mu s$. We generate virtual BMaps using a random uniform distribution between a minimum grant size of 1 slot and a maximum of 50 slots (i.e., the maximum is about 5% of the total frame size), always leaving one slot as guard interval between the requests, and in a way that balances the requests across each frame.

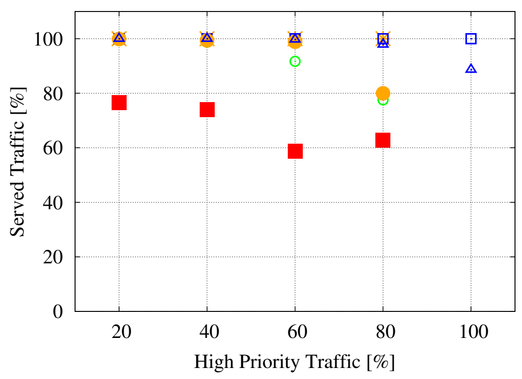
Figures 3.a and 3.b compare the performance of both algorithms as a function of the offered traffic, the first showing the percentage of traffic that was actually served (i.e., not dropped) and the second the average additional latency introduced by the merging engine operations. In this case the traffic load is distributed uniformly across the 4 priority classes. With PBMA, with the increase in offered traffic, the served traffic of the lowest priority type starts to decrease noticeably. Latency tends to affect mostly the lowest priority class 1, with some repercussions also noticeable for classes 2 and 3. The FAMA algorithm instead, by its tight packing approach achieved by



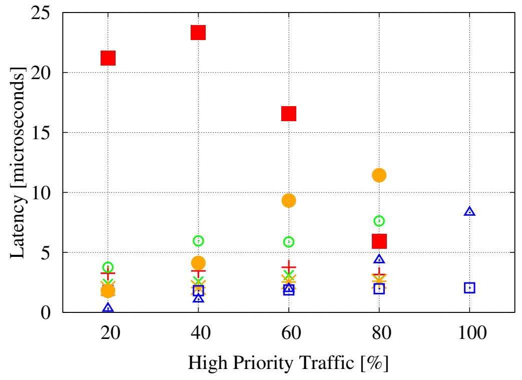
(a) served traffic vs offered traffic



(b) Latency vs offered traffic



(c) served traffic vs high priority traffic



(d) Latency vs high priority traffic

Fig. 3. Simulation Results

looping over each slot of the frame one by one, is capable of reducing latency across all classes, with the most noticeable improvement for class 1 (the latency is comparable to that of priority 3 in the PBMA algorithm). The main advantage of PBMA is that it can achieve the lowest latency for class 4 (less than 1 μs), because this priority is allocated in advance of all other calculations, which however leads to lower performance for all other classes.

As PONs might be used for transporting fronthaul 5G traffic, it is also important to understand the amount of highest priority traffic that an operator can support before violating a given latency requirement. Thus, in Fig. 3.c and 3.d, we show the system performance when we vary the amount of highest priority traffic (up to 100% of the total). The remaining traffic is equally distributed across the other three classes (and only shown up to the 80% point, as at 100% only priority traffic 4 is present). The figures show again how the FAMA algorithm outperforms the simpler PBMA approach, as it optimises frame utilisation by doing a fragment-less packing of allocations. For the FAMA approach, the latency added to the highest priority is only about 2 μs, and even the lowest class remains below 5 μs. On the other hand, the simpler PBMA algorithm tends to drop the second-highest priority traffic at 80% traffic level and also about 10% of the highest priority traffic at full load, with higher latency across all classes. The trade-off is on the higher complexity of FAMA, because for each allocation in the merged output FAMA has to choose a suitable request for each VNO that can compete for the current time slot. This trade-off will be further evaluated in our future work.

Acknowledgements

Financial support from SFI grants 14/IA/252 (O'SHARE), 13/RC/2077 and $12/RC/2276_P2$ is gratefully acknowledged.

References

1. BBF standard TR-370:Fixed Access Network Sharing - Architecture and Nodal Requirements. Nov. 2017
2. S. Bidkar, et al., First Demonstration of an Ultra-Low-Latency Fronthaul Transport Over a Commercial TDM-PON Platform. OFC 2018, p. Tu2K.3.
3. M. Ruffini, et al. Moving the Network to the Cloud: the Cloud Central Office Revolution and its Implications for the Optical Layer. JLT 37(7), Apr. 2019.
4. L. Peterson et al., Central office re-architected as a data center. IEEE Communications Magazine Oct. 2016.
5. A. Elrasad et al., Frame Level Sharing for DBA virtualization in multi-tenant PONs. ONDM 2017.
6. BBF Standard, TR-402: Functional Model for PON Abstraction Interface. Oct 2018.
7. ITU-T standard. G.989.3-Am1. NG-PON2: Transmission convergence layer specification Amendment. Nov. 2016.