Analysis of the Maximum Balanced Load in Long-Reach PONs

Pedro Alvarez, Alan Hill, Nicola Marchetti, David Payne, Marco Ruffini
CONNECT, Trinity College Dublin
University of Dublin
Dublin, Ireland
Email: pinheirp@tcd.ie, alanhill@timewave.f9.co.uk, marchetn@tcd.ie, david.b.payne@btinternet.com, ruffinm@tcd.ie

***Abstract*—Maximum balanced load is the load above which mean packet delay and packet loss ratio increase rapidly, queues start to build up and packets start to drop. In this paper, we propose theoretical formulas to analyse the maximum balanced load in Long-Reach Passive Optical Networks (LR-PONs). This is done for two algorithms, the GIANT algorithm and a proposed BwUpdate algorithm. We compare the expressions obtained with simulation studies and find the simulation results match the theory well. We compare the two algorithms as a function of the Assured Bandwidth Restoration Time (ABRT) and find that the BwUpdate algorithm increases the maximum balanced load, for assured bandwidth restoration times longer than 3.75 milliseconds.**

I. INTRODUCTION

Many Dynamic Bandwidth Assignment (DBA) algorithms in the literature focus on mean packet delay as the overriding delay parameter of Passive Optical Networks (PONs) [1]–[7]. Yet, there is no single value of mean packet delay recommended; only recommended values for maximum packet delays.

As an example, the ITU-T Recommendation Y.1541 allows up to 15 ms delay in the access network, for both upstream and downstream, and for both class 0 and class 1 Hypothetical Reference Paths [8]. Similarly, the FSAN G-PON Common Technical Specification requires 10 ms of maximum upstream delay (including jitter) for its low delay, low jitter, low packet loss class of service [9]. More recently, an Assured Bandwidth Restoration Time (ABRT) with a target of 2 ms, and a few milliseconds expected, is recommended for XG-PON in G.987.3 [10]. ABRT is the worst-case delay (not mean) between the moment a Transmission Container (T-CONT) increases its traffic demand to at least its fixed plus assured level, and the start of the first upstream frame that the specified bandwidth is allocated.

Taking these recommendations into account, it is important to maximize the PON's efficiency while meeting the delay requirements. Beyond a certain PON load, mean packet delay and packet loss ratio increase rapidly, since queues start to build up and packets begin to drop. It is important to know the load at which this rapid increase begins, i.e. the *maximum balanced load*. Obviously, the higher this maximum balanced load is, the better.

Two major factors influence the maximum balanced load: the size of the PON and the DBA algorithm used. In DISCUS [11] we wish to support Long-Reach PONs (LR-PONs) with up to 1,023 Optical Network Units (ONUs), each with up to 16 T-CONTs, i.e. 16,368 T-CONTs altogether, over 125 km logical reach. These are very challenging numbers, compared with those simulated for XG-PON in [6] and [7], and will generate far greater bandwidth overheads. Analytical calculations would make such numbers much easier to deal with, compared to experimental or simulation work.

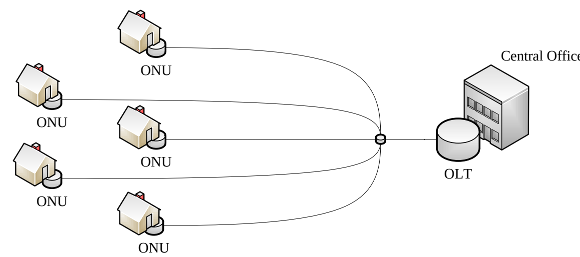

Fig. 1. Typical Fibre To The Home architecture

In this paper, we describe an analytical method of calculating the maximum balanced load of LR-PONs from the PON parameters and compare the theory with computer simulations of the LR-PON protocol. We do these for two DBA algorithms: GIANT [5] and the Bandwidth Update algorithm proposed in this paper. Instead of using mean packet delay as the delay criterion of merit, for which there is no preferred value, the ABRT is employed. Some of the techniques in the literature for minimising mean packet delays, such as predicting current bandwidth demand from earlier demand in a previous DBA cycle [1], do not apply to the ABRT. The ABRT must be met when there was no previous traffic present.

II. DYNAMIC BANDWIDTH ALLOCATION

In PONs, the upstream transmissions from the ONUs are arbitrated by the scheduler at the Optical Line Terminal (OLT), which tells the ONUs when to transmit. The algorithm that schedules these transmissions influences greatly the maximum balanced load, since it will decide the number of bursts created, thus influencing the amount of overheads. For that reason, to understand the maximum balanced load of a PON, it is important to know its particular DBA mechanisms.

In this paper we consider two DBA algorithms: GIANT [5] and the proposed BwUpdate algorithm. In GIANT each bandwidth type is characterized by two parameters: the allocation bytes and the service interval. The allocation bytes dictate how many bytes a T-CONT can transmit in a frame, while the service interval, specified in frames, defines how often a T-CONT can transmit. To decide when a T-CONT is allowed to transmit, the DBA engine also keeps a counter for each bandwidth type. This counter is decremented every frame, and whenever it expires the T-CONT is allowed to transmit.

It is important to note, that there is a difference between the way counters are refreshed for the assured bandwidth type and the non-assured bandwidth type. For the assured bandwidth the counters are refreshed to their service interval whenever the counter expires, regardless of whether the T-CONT needs to transmit or not; this is done to reserve bytes in the frame and ensure the assured bandwidth is guaranteed. For the non-assured bandwidth, the counters are used to limit the rate a T-CONT can transmit at, which means that they are only refreshed to their service interval when an actual transmission is made. This means that non-assured grants can become unsynchronised with the assured grants, thus requiring more bursts. This increases the overheads, thus reducing the maximum balanced load. More recent DBA algorithms based on GIANT [3], [6], [7], allow more than one ONU burst within the same service interval, even for assured grants, compounding the problem.

For this reason, we propose the BwUpdate algorithm. This algorithm joins the assured and the non-assured grants into a single burst, by computing the allocations not for a single frame, but for an entire service interval, as in B-PON [12] Unlike B-PON's bandwidth update algorithm, to reduce the delay that this adds, the service interval is made variable, ranging from the minimum number of frames needed to send all the buffer occupancy information to a configurable maximum. In Fig. 2 this is illustrated, where in the first calculation of the BwMap there is no load, so only enough is assigned to guarantee DBRus. In the next service interval, due to a higher load, allocations span through multiple frames, up to the maximum service interval.

The pseudo-code for this algorithm is shown in Procedure 1. Here, the main function is called in every frame to generate a BwMap. If a frame is the start of a service interval, then all the bursts are computed for the entire service interval. To compute the bursts for the interval, the scheduler first creates a list of bursts for all T-CONTs, with a grant size just big enough for the buffer status reports. Then, the assured and non-assured grants are given, in that order, and the bytes computed are added to the existing bursts.

To limit the allocation to a T-CONT's maximum assured or non-assured bandwith each T-CONT has allocation bytes, similar to GIANT. Unlike GIANT, the allocation bytes will be variable in time, since the service interval is variable too. To compute the allocation bytes, at the beginning of the service interval all the T-CONTs will update their allocation bytes according to equation (1). Here, AB_{min} is the minimum amount of allocation bytes possible (i.e. AB_{max}/SI_{max}), and $FrameCount$ represents the number of frames the last service interval had. This will maintain the maximum allowable rate at AB_{max}/SI_{max}, even with the variable service interval.

Procedure 1 BwUpdateGenerateBwMap

```
if m_siPerBurstInfo.empty() then
    CreateBurstsAndAllocateDbrus()
    BwUpdateAllocateAssuredBw()
    BwUpdateAllocateNonAssuredBw()
    UpdateAssuredAllocationBytes()
    UpdateNonAssuredAllocationBytes()
    m_frameCount=1
end if
CopyBurstsToThisFrame();
m_frameCount++;
```

$$AB = \min(AB_{max}, AB_{min}FrameCount) \qquad (1)$$

The pseudo-code for assured assignments and non-assured assignment are shown in Procedures 2 and 3, respectively. Here, we can see that to assign bytes to a T-CONT, in both the assured and non-assured case, first the scheduler finds the correct burst for the ONU, and within this burst the correct allocation for the T-CONT. After this, the scheduler can then add the granted bytes to the correct allocation. In the assured part, the allocation takes the minimum of the previously computed assured bytes and the buffer request. In the non-assured, first the space left in the service interval is computed and then the minimum of the allocation bytes, the buffer request and space still available is granted.

Procedure 2 BwUpdateAllocateAssuredBw

```
for paramsIt in tcontAssuredParameters do
    tcont=paramsIt.GetTcont()
    onuId=paramsIt.GetOnuId()
    buffOcc = GetAmountOfDataToServe(tcont)
    if buffOcc is not 0 then
        burst = FindBurstForTcontInList(tcontOlt)
        alloc = burst.FindBwAlloc(tcont)
        abValue = paramsIt.GetABminCounterValue()
        size2assign = min(abValue,buffOcc)
        alloc.AddBytesToAllocation(size2Assign)
    end if
end for
```

At the end of Procedure 1 the BwMap for that particular frame is constructed, by simply getting from the list of bursts the burst that will go in that frame.

III. Maximum Balanced Load

The maximum balanced load is obtained when the available PON capacity, excluding bandwidth overheads, just equals the traffic level arriving. Arriving and departing packets are in equilibrium, no packets are lost, and the T-CONT queues do not build up with time. In this analysis, we consider both the GIANT and BwUpdate DBA algorithms when all T-CONTs

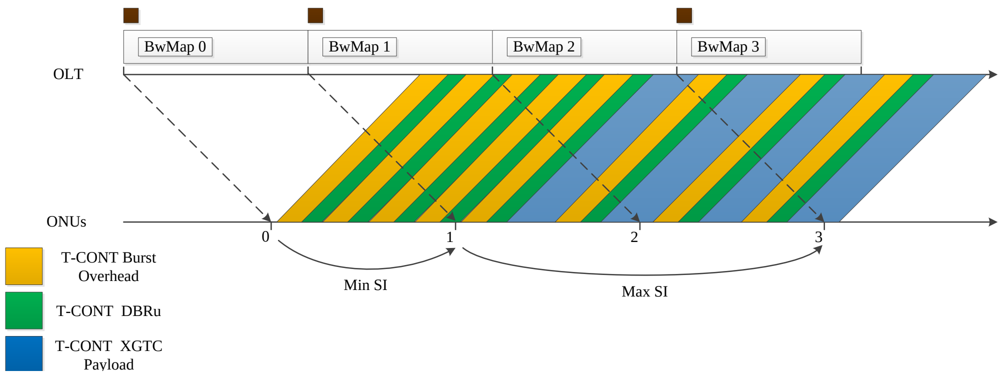

Fig. 2. BwUpdate algorithm

Procedure 3 BwUpdateAllocateNonAssuredBw

```
for paramsIt in tcontNonAssuredParameters do
    tcont=paramsIt->GetTcont()
    onuId=paramsIt.GetOnuId()
    buffOcc = GetAmountOfDataToServe(tcont)
    if buffOcc is not 0 then
        availableSize = siBurstsQueue.GetSiAvailableSize()
        if availableSize == 0 then
            break
        end if
        abValue = paramsIt.GetABsurCounterValue()
        size2Assign = min(abValue,buffOcc,availableSize)
        burst = FindBurstForTcontInList(tcontOlt)
        alloc = burst.FindBwAlloc(tcont)
        alloc.AddBytesToAllocation(size2Assign)
    end if
end for
```

TABLE I
PON PARAMETERS

N_G, N_{up}	average number of packets served per service interval
SI	service interval in number of XGTC frames
F	number of bytes per frame (e.g. 155,520 at 10 Gbit/sec)
B	$SI.F$ = number of bytes per service interval
N_{onu}	number of ONUs
N_{alloc}	number of Alloc-IDs (T-CONTs)
AB_{ass}	Assured Bandwidth allocation bytes
AB_{sur}	Non-assured bandwidth allocation bytes
P_s	mean packet size
O_{dbru}	dynamic bandwidth report size
O_{xgem}	XGEM frame header
O_{xgtc}	XGTC header + trailer
O_{phy}	guard time + preamble + delimiter
O_{burst}	$O_{xgtc} + O_{phy}$

support both assured and non-assured bandwidth types. To calculate the maximum balanced load, we compute the amount of payload bytes transported in a service interval when the amount of packets arriving equals the amount departing, and all other bytes in the service interval are overheads. A summary of the relevant parameters required for the derivation of these equations is given in Table I.

A. GIANT Maximum Balanced Load

The GIANT DBA algorithm operates frame-by-frame. Assured bandwidth grants are always given in the same frame within the service interval. Non-assured bandwidth grants are assigned at the first available opportunity, thus are served in any frame within the service interval. So assured and non-assured bandwidth grants to the same T-CONT can become decorrelated, i.e. unsynchronised. Furthermore, in GIANT, each T-CONT's payload is served in one frame only; grants are not continued into the next XGTC frame.

To calculate the maximum balanced load, first we compute the overheads for the assured bandwidth grants in the service interval (SI). The physical and XGTC burst overhead is given by multiplying the number of ONUs (N_{onu}) by the overhead of each burst ($O_{burst} = O_{phy} + O_{xgtc}$). Besides the burst overhead, it is necessary to consider the overhead due to buffer reports, given by multiplying the number of T-CONTs (N_{alloc}) by the buffer occupancy report size (O_{dbru}). Also, each packet will have an XGEM header (of size O_{xgem}) that needs to be taken into account. Thus, the XGEM overheads are given by $N_G.O_{xgem}$ and we can get equation (2) for the overall overheads in a service interval, due to assured grants.

$$ABW_o = N_{onu}(O_{burst} + \frac{N_{alloc}O_{dbru}}{N_{onu}}) + N_G O_{xgem} \quad (2)$$

We assume that AB_{ass} is large enough to carry only part of the packet. This is not unreasonable to assume in very large PONs with short service intervals. The payload transmitted by the assured grants (ABW_p) is then given by (3). Therefore, the number of bytes available per SI for non-assured grants plus their overheads will be given by equation (4).

$$ABW_p = N_G AB_{ass} \quad (3)$$

$$P_{available} = B - ABW_o - ABW_p \quad (4)$$

For non-assured bandwidth grants, it is assumed that within each frame, more than one packet is waiting in the same ONU, but in a different T-CONT queue, so multiple packets share the same burst overhead. On average, the number of packets waiting in each ONU when being granted non-assured bandwidth in a service interval is $\frac{N_G}{N_{onu}}$ packets per ONU. Therefore, the total overhead plus payload bytes per non-assured burst ($NABW_{o+p}^{burst}$) is given by eq. (5), where P_s represents the mean packet size. Hence, the number of ONUs served with non-assured bandwidth per SI is given by (6).

$$NABW_{o+p}^{burst} = O_{burst} + \frac{N_G}{N_{onu}}(O_{xgem} + P_s - AB_{ass}) \quad (5)$$

$$N_{onu}^{nabw} = \frac{P_{available}}{O_{burst} + \frac{N_G}{N_{onu}}(O_{xgem} + P_s - AB_{ass})} \quad (6)$$

The payload occupied by the non-assured grants is then given by multiplying the number of packets per ONU, the number of ONUs served per SI and the size of the remaining part of the packet, thus giving equation (7).

$$NABW_p = \frac{N_G}{N_{onu}} \frac{P_{available}.(P_s - AB_{ass})}{O_{burst} + \frac{N_G}{N_{onu}}(O_{xgem} + P_s - AB_{ass})} \quad (7)$$

Now, because we assume that over the SI the number of arriving packets equals the number of granted packets (transmitted/forwarded from the T-CONTs), i.e. no queue build-up overall, we can say that the number of bytes of payload per packet equals the average packet size (P_s). Therefore, the number of assured and non-assured payload bytes arriving and transmitted per service interval is given by (8).

$$N_G.P_s = ABW_p + NABW_p \quad (8)$$

By combining equations (8), (3) and (7) we get the number of packets per SI, N_G, as depicted in (9).

$$N_G = \frac{B - 2N_{onu}O_{burst} - N_{alloc}O_{dbru}}{2O_{xgem} + P_s} \quad (9)$$

Since the load is given by equation (10), we get the final expression for the maximum balanced load in (11).

$$L_G = \frac{N_G P_s}{SI.F} \quad (10)$$

$$L_G = \frac{SI.F - 2N_{onu}O_{burst} - N_{alloc}O_{dbru}}{SI.F\left(1 + \frac{2O_{xgem}}{P_s}\right)} \quad (11)$$

B. BwUpdate Maximum Balanced Load

In the bandwidth update algorithm, the assured bandwidth overhead bytes are the same as in GIANT, see (12), because the same number of bursts will happen during a SI, to guarantee buffer occupancy reports. Since in the bandwidth update algorithm the non-assured bursts are arranged to go with the assured grants, the number of available payload bytes per SI is given by the equation (13).

$$ABW_o = N_{onu}(O_{burst} + \frac{N_{alloc}O_{dbru}}{N_{onu}}) + N_{up}O_{xgem} \quad (12)$$

$$P_{up} = B - N_{onu}O_{burst} - N_{alloc}O_{dbru} - N_{up}O_{xgem} \quad (13)$$

Since the average number of packets arriving and transmitted per SI is given by dividing the available payload by the average packet size, as in equation (14), we can then get equation (15) to express the available payload.

$$N_{up} = \frac{P_{up}}{P_s} \quad (14)$$

$$P_{up} = \frac{P_s\,(B - N_{onu}O_{burst} - N_{alloc}O_{dbru})}{P_s + O_{xgem}} \quad (15)$$

Since the load can be expressed as the ratio between the used payload and the total available bytes, as in (16), we can then finally obtain the expression for maximum balanced load for the bandwidth update algorithm, given in equation (17).

$$L_{up} = \frac{P_{up}}{SI.F} \quad (16)$$

$$L_{up} = \frac{SI.F - N_{onu}O_{burst} - N_{alloc}O_{dbru}}{SI.F.(1 + \frac{O_{xgem}}{P_s})} \quad (17)$$

IV. Assured Bandwidth Restoration Times

When using assured bandwidth, there is a delay between the moment traffic demand increases, to the moment the DBA engine actually reacts to the increase. To give some guarantees on this delay, the XG-PON recommendations [10] define the Assured Bandwidth Restoration Time. This time is defined as the worst-case delay, as observed by the ONU, from an increase in traffic demand, to the departure of the first grant where the new assured bandwidth is allocated. According to the recommendations, this value has a target value of 2 ms with a few milliseconds expected.

To be able to use the ABRT as a performance metric, we relate this parameter to the service interval of both the GIANT and the BwUpdate algorithm. In GIANT, if we assume that each T-CONT is configured to send DBRus synchronously with any assured bandwidth grants, then we can express the ABRT as a function of the round-trip delay time (RTT) and the service interval. This is illustrated in Fig. 3. Here we can see the three cases for the computation of the ABRT: when the SI is larger than the RTT, when the SI is smaller, but a multiple of RTT, and when the SI is smaller and not a multiple of RTT. An arriving packet may just miss a DBRu reporting opportunity, and the DBRu may just miss a downstream BWmap opportunity.

We can see the relation between the service interval and the ABRT for GIANT algorithm in (18).

$$ABRT_G = \begin{cases} SI + rtt & \text{if } SI < rtt, rtt = nSI \\ 2SI + SI\lfloor\frac{rtt}{SI}\rfloor & \text{if } SI < rtt, rtt \neq nSI \\ 2SI & \text{if } SI \geq rtt \end{cases} \quad (18)$$

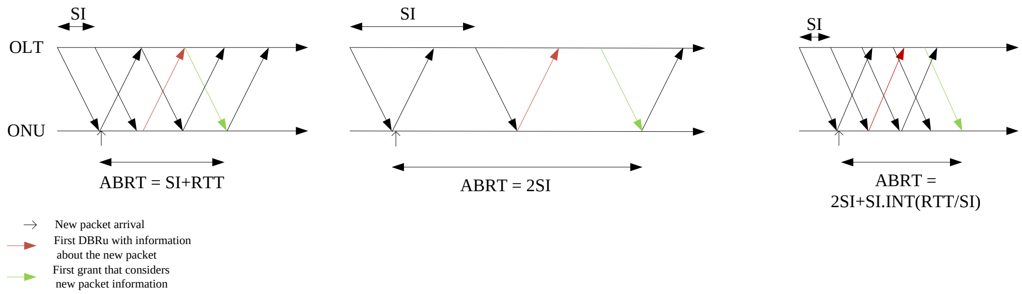

Fig. 3. Assured Bandwidth Restoration Time for GIANT.

We can see the relation between the service interval for the BwUpdate algorithm in (19). An additional SI delay is incurred because the DBA engine computes bandwidth assignments for all T-CONTs over an entire SI. It is assumed that during the ABRT, the location of a T-CONT's DBRu within each of the SIs remains roughly constant, because the probability of an additional SI variation due to statistical traffic fluctuations on this timescale is so extremely low that it can be ignored. It would require all T-CONTs before the one of interest in the list of Alloc-IDs to be inactive, when the first packet of a new session arrives in the T-CONT, and all the T-CONTs after it in the list to be active, then this state completely reversing before the first payload grant is transmitted upstream at the end of the ABRT, i.e. 2 or a few msec later. The values in (19) represent the worst-case delay under normal circumstances.

$$ABRT_{up} = \begin{cases} 2SI + rtt & \text{if } SI < rtt, rtt = nSI \\ 3SI + SI\lfloor \frac{rtt}{SI} \rfloor & \text{if } SI < rtt, rtt \neq nSI \\ 3SI & \text{if } SI \geq rtt \end{cases} \tag{19}$$

V. Maximum Balanced Load Theory Evaluation

To evaluate our maximum balanced load theory, we simulated an LR-PON using the 10-gigabit-capable Passive Optical Network (XG-PON) module for *ns-3* [13], and compared the results with our theory. To do this, we varied the load offered to the PON at various service intervals and looked for the point at which the network started to saturate.

We considered a PON that supports 1023 ONUs, each ONU with 16 T-CONTs. Being a LR-PON, we considered a 100 km distance between the OLT and the ONU, which implies a two-way propagation delay of 1 ms. An RTT value of 1.25 ms was used, to include one frame for DBRu collection and one frame for DBA computation. We also considered a throughput of 10 Gbit s^{-1} in the upstream, which is not standard, but we believe is a requirement for such a large number of clients. To generate the traffic, we considered a tri-packet model with standard Poisson distributed inter-arrival times, as in [5]. The relevant simulation parameters are given in Table II.

To assign allocation bytes, both in GIANT and BwUpdate, we allocated 90% of the service interval to assured bandwidth

TABLE II
PON Simulation Parameters

ONUs	1023
T-CONTs per ONU	16
SI	{6,8,10,12,16,20,24,30,32}
Upstream Throughput	10 Gbit s^{-1}
Simulated time	2 s
Queue Size	100 kB
Application Traffic Type	Poisson
Packet Size (Bytes)	64 (60%), 500 (20%), 1500 (20%)

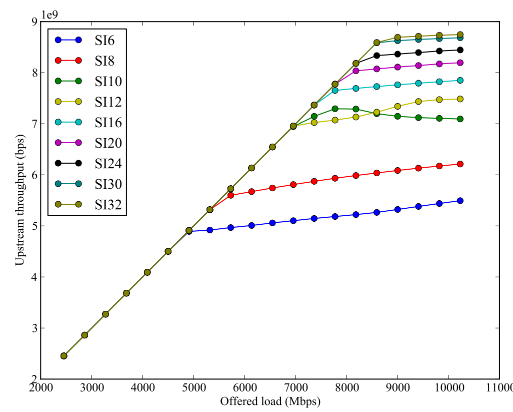

Fig. 4. Network rate vs. offered load for GIANT.

and allowed each T-CONT up to 200 Mbit s^{-1} of peak-rate limit, including assured and non-assured capacity.

The upstream network throughput when using the GIANT algorithm is shown in Fig. 4. Here, we can see that increasing the service interval increases the maximum balanced load, at the cost of extra delay, as expected. We can see some fluctuations in the load, especially for SI 10 and 12, but this is attributed to a variation in the degree of synchronization between assured and non-assured grants.

Fig. 5 shows the network throughput using the BwUpdate algorithm. Here, we can see the same principle, that increasing the SI decreases the overheads, at the cost of extra delay. It is notable, that the throughput is much more stable in BwUpdate, since it is not possible for assured and non-assured bursts to be unsynchronised, thus avoiding the fluctuations of GIANT.

By looking at the points where the throughput saturates, we can see the load where the maximum balanced load is reached. We then compare the maximum balanced load for different

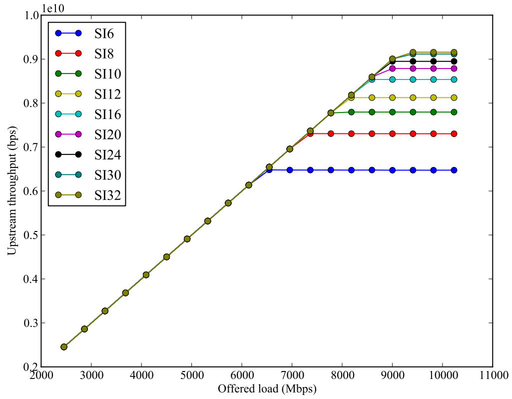

Fig. 5. Network rate vs. offered load for BwUpdate.

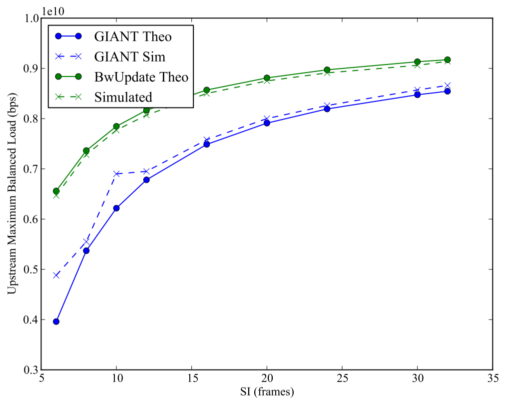

Fig. 6. Theoretical and simulated maximum balanced load vs. service interval.

SIs for both GIANT and BwUpdate in Fig. 6. For the same SI the BwUpdate algorithm has much higher maximum balanced load. We can also see the closeness of the simulated results to the theory. Differences can be found, mostly in GIANT, since the assumption of all assured and non-assured bursts always being unsynchronised is not always the case.

However, the comparison between the two algorithms is not quite as simple as this, because each algorithm incurs a different ABRT for the same service interval SI. Fig. 7 shows the maximum balanced load when the SI is converted into ABRT, according to equations (18) and (19). We see that the BwUpdate algorithm maintains an advantage, in terms of maximum balanced load, when compared with GIANT as a function of ABRT, for ABRTs longer than 3.75 ms. Theory shows up to $500\ \mathrm{Mbit\,s^{-1}}$ (5%) higher load, while simulations show up to $340\ \mathrm{Mbit\,s^{-1}}$ (3.4%).

VI. Conclusion

We analysed the maximum balanced load in LR-PONs – i.e. the load at which the PON saturates – through simulation studies and a proposed theoretical analysis. We did this for two different algorithms, GIANT and the proposed BwUpdate algorithm, and found that our simulation results match our theoretical results well. Some discrepancies were found at lower service intervals for the GIANT algorithm, as we assumed that bursts from assured and non-assured grants are

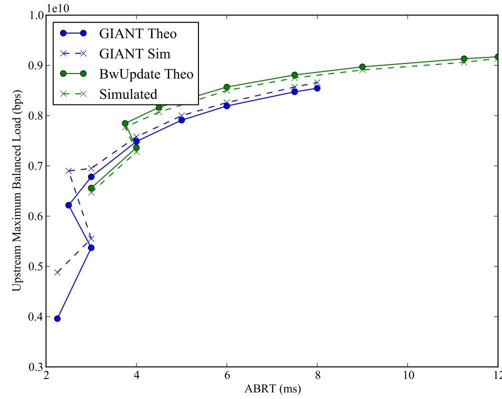

Fig. 7. Theoretical and simulated maximum balanced load vs. ABRT.

always unsynchronised, which is not always true. We then compared the maximum balanced load of these algorithms as a function of the ABRT, showing that BwUpdate improves the efficiency of a 100 km PON for ABRTs longer than 3.75 ms.

Acknowledgment

This project has been part funded by the European Regional Development Fund through the Science Foundation of Ireland research centres programme under Grant Number 13/RC/2077 (CONNECT) and part funded by the European Union Seventh Framework Program (FP7/2007-2013) under grant agreement n. 318137 (Collaborative project DISCUS).

References

[1] C.-H. Chang *et al.*, "Dynamic bandwidth assignment for multi-service access in long-reach GPON," in *33rd European Conference and Ehxibition of Optical Communication (ECOC)*, Sept 2007, pp. 1–2.

[2] N. Merayo *et al.*, "Adaptive polling algorithm to provide subscriber and service differentiation in a long-reach epon," *Photonic Network Communications*, vol. 19, no. 3, pp. 257–264, 2010.

[3] M.-S. Han *et al.*, "Efficient dynamic bandwidth allocation for fsan-compliant gpon," *J. Opt. Netw.*, vol. 7, no. 8, pp. 783–795, Aug 2008.

[4] H. Song *et al.*, "Multi-thread polling: a dynamic bandwidth distribution scheme in long-reach pon," *Selected Areas in Communications, IEEE Journal on*, vol. 27, no. 2, pp. 134–142, February 2009.

[5] H. C. Leligou *et al.*, "Efficient medium arbitration of fsan-compliant gpons," *International Journal of Communication Systems*, vol. 19, no. 5, pp. 603–617, 2006.

[6] M. S. Han *et al.*, "Development of efficient dynamic bandwidth allocation algorithm for XGPON," *ETRI Journal*, vol. 35, no. 1, pp. 18–26, February 2013.

[7] J. Arokkiam *et al.*, "Refining the giant dynamic bandwidth allocation mechanism for xg-pon," in *IEEE International Conference on Communications (ICC)*, June 2015, pp. 1006–1011.

[8] *Network performance objectives for IP-based services*, ITU-T Recommendation Y.1541, ITU Std., dec 2011.

[9] A. Cauvin *et al.*, "Common technical specification of the g-pon system among major worldwide access carriers," *Communications Magazine, IEEE*, vol. 44, no. 10, pp. 34–40, Oct 2006.

[10] *10-Gigabit-capable passive optical networks (XG-PON): Transmission convergence (TC) layer specification*, ITU-T Recommendation G.987.3, ITU Std., Jan. 2014.

[11] M. Ruffini *et al.*, "DISCUS: an end-to-end solution for ubiquitous broadband optical access," *Communications Magazine, IEEE*, vol. 52, no. 2, pp. S24–S32, Feb. 2014.

[12] *A broadband optical access system with increased service capability using dynamic bandwidth assignment*, ITU-T Recommendation G.983.4, ITU Std., Nov. 2001.

[13] X. Wu *et al.*, "An XG-PON Module for the ns-3 Network Simulator," in *6th International ICST Conference on Simulation Tools and Techniques (SimuTools)*, 2013.