SELECTIVE VOID CREATION/FILLING FOR VARIABLE SIZE PACKETS AND MULTIPLE WAVELENGTHS

KURT VAN HAUTEGEM, WOUTER ROGIEST AND HERWIG BRUNEEL
SMACS Research Group
Department of Telecommunications and Information Processing (TELIN)
Ghent University
St.-Pietersnieuwstraat 41, B-9000 Ghent, Belgium

(Communicated by Wuyi Yue)

ABSTRACT. With ever-increasing demand for bandwidth, both optical packet switching and optical burst switching are proposed as alternatives to increase the capacity of optical networks in the future. In these packet-based switching techniques, Fiber Delay Lines (for delay assignments) and wavelength conversion (for channel assignments) are used to avoid contention between contending packets. The involved scheduling algorithms decide on which Fiber Delay Line and wavelength each packet is scheduled in order to maximize performance. For the setting without wavelength conversion we proposed a scheduling algorithm for assigning delays called void-creating algorithm that outperforms existing void filling algorithms for a variety of packet size distributions. This is achieved by selectively delaying packets longer than strictly necessary based on a numerical procedure that assigns a theoretical value to each void based on how likely the void will eventually be filled and thus prove useful. This contribution extends the concept of void-creation to the important case with multiple wavelengths, where also the channel has to be assigned. Results obtained by Monte Carlo simulation show that with our void-creating algorithm the obtainable improvement in various performance measures highly depends on the number of wavelengths present.

1. Introduction. With ever growing numbers of connected devices, demand for downstream bandwidth is expected to increase by an annual rate of 30 to 40% [17]. Moreover new technologies as connected cars will put even more strain on the backbone network by increasing the upstream requirements. With ever increasing bandwidths of up to 43 Terabit/s with just a single laser [8], optical fiber seems the perfect answer to our insatiable demand for data. In optical networks, however, currently circuit switching is used to establish end-to-end communication. As in circuit switching a dedicated connection is maintained for the whole duration of the communication session, capacity remains unused when the two nodes are temporarily not communicating. Although for some cases this capacity loss does not outweigh the benefits of circuit switching, it does for a type of traffic that is very bursty and packet-oriented by its very nature, i.e. Internet traffic.

As the backbone capacity is curtailed by the circuit switching paradigm, optical packet-based switching techniques are seen as promising solutions to flexibly share network links among communication sessions. Both optical packet switching (OPS)
and optical burst switching (OBS) are packet-based and provide viable alternatives to increase the usage of the available fiber capacity. Although there has been a lot of criticism concerning the feasibility of OPS/OBS in the near future [15], the technology remains an unignorable alternative to slow electronic switching in the backbone [13]. The fact to date that OPS, OBS and hybrid switching technologies are still being studied extensively from a software and hardware point of view [4, 24, 7, 5, 14], shows that they still are very hopeful technologies that in the future will be useful and probably even necessary. If so, efficient void filling and packet scheduling algorithms will be at the heart of the OPS/OBS switch design.

As packet-based switching techniques do not strictly separate different communication flows, contention may arise when more than one packet heads for the same output port at the same time. They therefore require contention resolution in intermediate nodes. The best solution to avoid conversion is the combined use of wavelength conversion and Fiber Delay Lines (FDLs) [2]. In these FDLs, packets that need to be buffered are sent through sufficiently long pieces of coiled fiber to delay them for the time needed [11, 1]. Additionally with the use of wavelength converters (WCs) packets can be converted to a different wavelength thereby also avoiding contention. In this paper we will focus on feed-forward set-up, in which packets traverse only a single line part of a set of Fiber Delay Lines (FDLs) with different lengths. Moreover as the effective use of the wavelength converters is, as opposed to previous work [20, 19], not the main focus for this paper we assume a set-up with an unrestricted capacity for wavelength conversion.

As unresolved contention results in packet loss, it is necessary to schedule arriving packets as efficiently as possible on the available FDLs and wavelengths. Existing algorithms can be split up in to two main categories: void-filling and non-void-filling algorithms [3, 19, 20]. In contrast to the latter, the former allow packets to be scheduled before already scheduled packets, filling the so-called voids (unscheduled periods between already scheduled packets), thereby improving the performance significantly. As these void-filling algorithms keep track of all voids this improvement comes at the cost of an increased computational complexity. In [21, 18] we therefore proposed a totally new type of algorithm for a single wavelength set-up (and thus without wavelength conversion), the void-creating algorithm, that selectively creates larger voids only when they will likely be filled in the future. This algorithm does not only fill the available voids, it also, based on the system conditions, controls the creation of the voids enabling better switch performance. As interesting as these results are, they are limited to a single degenerate case in which the packet size is fixed. In a shorter and earlier version of the current paper [22] we therefore extended the use of the void-creating algorithm to a plurality of packet sizes. Moreover, the simulation results in this paper are backed by a numerical procedure that assigns a theoretical value to each void based on how likely the void will be filled in the future. This framework deepens the insight in the mechanism of the void-creating algorithm and makes it possible to apply it in other settings without (a lot of) additional simulations.

In this paper we extend the use of this different type of algorithm to the frequently occurring, and thus important, setting of multiple wavelengths and fixed size packets. Performance improvements are obtained for an increasing number of wavelengths allowing a comparison across a set of different wavelengths. As the basic void-creating algorithm obtained significant performance improvements in the single wavelength setting, a key research question is whether the idea of deliberately
creating larger voids than necessary, which is the underlying principle of all void-creating algorithms, may yield similar performance benefits in a multiwavelength setting. To answer this question, a basic extension of the algorithm is put to test in a series of simulation runs. By doing so, we demonstrate the potential that can be expected from void-creating algorithms in a broader sense, i.e. all algorithms with the mentioned idea as underlying principle. As such, the performance results obtained for the basic extension may serve as a benchmark for future, more complex, algorithms based on both void creation or other mechanisms. In this setting it is also shown that the void creating mechanism is not a direct competitor of any (advanced) void filling method and could even be merged with these types of algorithms combining the performance improvement of both.

2. Background.

2.1. Motivation and assumptions. Throughout the paper the same continuous-time setting as in [19, 20, 21, 18, 22] is supposed. Fig. 1 shows the assumed $K \times M$ optical switch configuration. Because the exact future and applications of OPS/OBS are still undecided, we study a generalized switch configuration which is not limited to a single type of network infrastructure (i.e. Telco networks or intra/inter data center connections). In this ideal non-blocking switching matrix packets arrive on a finite number of incoming ports $K$, on $c$ different wavelengths $\lambda_1, \ldots, \lambda_c$. Each packet arrives on a certain wavelength and is switched to one of the $M$ output ports which are connected to a single fiber with the same $c$ different wavelengths.

As in [19, 20, 21, 18, 22] an arbitrary single output port is analyzed, marked by the dashed-line box in Fig. 1. As multiple wavelengths are present, wavelength converters (not shown in Fig. 1) are available at the output port to change the wavelength of an arriving packet if demanded so by the scheduling algorithm. This is as opposed to [22], where no wavelength converters are present and the analysis can thus be confined to a single wavelength. Each output port has an optical buffer in which $N+1$ FDLs are available to schedule incoming packets. The lengths of the FDLs are consecutive multiples of a basic value $D$ called the granularity. This is called a degenerate delay buffer [12], in which incoming packets sent through the $j$-th ($j = 0 \ldots N$) delay line encounter a delay of $j \cdot D$. We assume packets arrive at the output port on wavelength $\lambda_1$ according to a Poisson process, with exponentially distributed inter-arrival times $T$ and average $E[T]$. The packet size is denoted $B$, its average is denoted $E[B]$.

In [22] the performance improvement for traffic with fixed packet size, exponentially distributed packet size and either of two uniformly distributed packet sizes was analyzed. There the performance improvements for fixed packet size proved more than those for uniform packet size, and much more than those for exponentially distributed packet sizes. This, together with the practical importance of the case of fixed packet size, not in the least for OPS testbeds, motivates our choice to focus the last part of this paper particularly on the extension of our model toward the case of multiple wavelengths in combination with fixed packet sizes. Particularly, the case is considered of a fixed packet size $B$ equal to the granularity, $B = D$. Matching the granularity and the (average) packet size is a natural and recommendable choice in switch design in view of performance (see [10]) and by doing so results can be easily compared.
The overall incoming traffic load at the output port is equal for all packet size distributions and given by $\rho = \frac{E[B]}{c \cdot E[T]}$. The nature of the arrival process implies possible overlap of packets at the entrance of the output port. It is this overlap that causes contention and is resolved by sending one of the contenting packets on a different wavelength or through one of the FDLs (or a combination of both). A packet is lost if none of the FDLs on none of the wavelengths can be chosen without causing contention with one of the other packets present in the system. It is therefore necessary to schedule packets as wisely as possible among the wavelengths and FDLs. In general, LP (loss probability) and packet delay are the main performance measures. Besides these, we will also consider the average packet gap, i.e. the time between packets on the outgoing line, and the LPsize, i.e. the sum of all lost packets multiplied with their size divided by the sum of all arrived packet sizes.

2.2. Scheduling basics. At the output port under study, scheduling is done separately for each packet upon its arrival. Scheduling amounts to the assignment of two variables ($i$ and $j$) to the given packet, corresponding to the outgoing wavelength ($i = 1 \ldots c$) and the delay line ($j = 0 \ldots N$). This is done by means of a provisional schedule, of which an example for the case of a single wavelength is given in Fig. 2. A provisional schedule shows the already scheduled packets (grey boxes) upon arrival of an arbitrary packet that is yet to be scheduled (and is thus not displayed on the provisional schedule). The arrival instant of this packet corresponds with the reference time (time=0) and thus the zero delay reference line on the provisional schedule. The vertical lines represent the delays of the FDLs ($j = 0 \ldots N = 5$), with the granularity in the example assumed as unity ($D = 1$) to ease notation.

In this representation, the provisional schedule evolution can be seen as a choppy (observed from arrival to arrival) but uniform (all packets move alike) movement of all packets to the left, with packets disappearing (because they are being transmitted) when crossing the zero delay reference line. The provisional schedule, although similar at first sight, should not be confused with a slotted arrival process as the inter-arrival times are distributed continuously and the vertical lines represent the delays of the FDLs and not slot boundaries. In the case of a single wavelength a packet has to be scheduled on an FDL without overlapping with any of the already present packets. If no such position is available, the packet is lost. Scheduling is done according to a scheduling algorithm, of which most can be split up in two main categories:
void-filling algorithms can schedule packets on any suitable FDL. This implies the possibility of filling up voids, defined as unscheduled periods followed by one or more packets scheduled beyond it. Only voids overlapping an FDL more than the packet size to the right of this FDL on the provisional schedule are fillable. The algorithm which schedules each packet on the lowest possible delay line (D-VF algorithm (delay-void-filling)) schedules the packet in the example of Fig. 2 on the FDL with delay 2 if the size of the packet does not cause an overlap with the packet already scheduled behind this FDL. If the packet size is too large, the packet is scheduled on the FDL with delay 4.

non-void-filling algorithms only allow packets to join at the back and voids can not be filled. The algorithm is thus not obliged to keep track of all voids but merely of the horizon, defined as the latest time at which the output is currently scheduled to be in use. Graphically, this corresponds to the right edge of the rightmost packet. In Fig. 2 the algorithm which schedules each packet on the lowest possible delay line (D-NVF algorithm (delay-non-void-filling) aka horizon scheduling [16]) schedules the packet on the first FDL to the right of the horizon, i.e. with delay 4, independently of the packet’s size.

As non-void-filling algorithms do not keep track of the voids, they have a lower computational complexity but they do so at the cost of a lower performance, i.e. higher LP, LPsize, packet delay and gap than the void-filling algorithms. Besides delay oriented scheduling algorithms as D-VF and D-NVF many other, more complicated and better performing scheduling algorithms exist [23, 6]. These, often void-filling, algorithms focus on the gaps before and after the FDLs that can schedule an arriving packet. Min-SV (or LAUC-VF) for example schedules the packet on the FDL resulting in the smallest starting void. Min-EV on the other hand minimizes the ending void if multiple voids are present. Others like the best fit void-filling algorithm [9] or cost-based algorithms [19, 20] take into account both start and ending void or create cost functions that incorporate both delay and gap information. Depending on the exact setting (packet size distribution, number of wavelengths, ...) performance differences between these algorithms can be non-existent, negligible or huge. It is therefore important that for each setting a well performing reference algorithm is chosen to compare the performance of the void-creating algorithm with.

3. Void-creating algorithms.

3.1. Fixed size packets on a single wavelength. To achieve a better performance without excessively increasing the computational cost we proposed the void-creating algorithm in [21, 18] for the setting of a single wavelength and a fixed packet size equal to the granularity \( E[B] = B = D \). In this setting a possible provisional schedule looks like Fig. 3a with all packets’ size equal to \( D \) (\( D = 1 \) again to simplify notation). The only packet that can be shorter than \( D \) is the one
crossing the zero delay line because at this instant the concerned packet is being sent and has thus already partially departed.

As \( D = B \) in the example of Fig. 3a only voids smaller than the fixed packet size (unfillable voids) are created by both void-filling and non-void-filling algorithms. No voids larger than the packet size (fillable voids) are created and thus none can be filled. In this setting of a single wavelength, and \( D = B \), all algorithms (both basic and more advanced) tend to schedule the packet on the first FDL to the right of the horizon (dot in Fig. 3a). This is unless the horizon is larger than \( N \cdot D \), in which case the packet is lost. As performance is thus the same for all the considered algorithms in this setting, we therefore chose the D-VF algorithm as our benchmark.

To improve the performance, the void-creating algorithm in \([21, 18]\) creates fillable voids by scheduling the packet on the second FDL (triangle) instead of the first FDL (dot) to the right of the horizon. This creates voids larger than the fixed packet size \( B \) (fillable voids) which, if favorably positioned, can be filled in the future. A favorable position occurs when the void overlaps an FDL at least a packet size to the right of the FDL, i.e. the (fillable) void is reachable. Analogously, a (fillable) void that is not positioned favorably with respect to an FDL is called unreachable.

When a fillable void created by a first packet is filled by a subsequent packet the average gap and delay encountered per packet will be lower than in the case when a fillable void was never created.

This is illustrated with an example in Fig. 3 in which the scenario of three subsequent packet arrivals is analyzed. Fig. 3 compares the evolution of the provisional schedules from arrival to arrival without (Fig. 3b) and with (Fig. 3c) void creation. In Fig. 3b both packet (1) and (2) are scheduled on the first FDL to the right of the horizon, corresponding to the strategy used by the D-VF algorithm. This results in an average delay of 4.5 for packets (1) and (2) (delay of 4 for packet (1) and 5 for packet (2)). The total gap assigned to packets (1) and (2) equals \( \text{gap}_1 + \text{gap}_2 \). Moreover at its arrival instant there is no FDL available to schedule packet (3), resulting in a lost packet. In Fig. 3c packet (1) is scheduled on the second FDL to the right of the horizon, in this way creating a fillable void. Upon its arrival packet (2) is able to fill this fillable void as it is reachable (it overlaps the FDL with a delay of 3 in a favorable way). The average delay for packets (1) and (2) equals 4 (delay of 5 for packet (1) and 3 for packet (2)) as opposed to an average delay of 4.5 for packets (1) and (2) without void creation. The total gap assigned to packets (1) and (2) equals \( \text{gap}_3' + \text{gap}_4' \) which is always smaller than in the case without void creation: \( (\text{gap}_3' + \text{gap}_4' = \text{gap}_3 - D = \text{gap}_1) < (\text{gap}_1 + \text{gap}_2) \). The stacking of packets (1) and (2) is thus more dense in Fig. 3c and the last FDL is available to schedule packet (3) when it arrives.

It is clear from the above example that filling a void lowers the average delay and gap. Moreover the example shows that reducing the average gap size mitigates the number of packets lost as it makes the stacking of the packets dense. On the other hand creating but not filling voids is disadvantageous for the performance as the stacking becomes less dense. Whether a void is reachable, depends on the choppy evolution of the provisional schedule and thus on the stochastic arrival process. The chance that a void is reachable is maximized by creating only voids that are likely to be filled. This is done by imposing a threshold on the size of the created fillable voids. As a void only slightly larger than \( D \) will have a small chance of being reachable, the condition states that a fillable void is only created when it is larger than a certain value \( D + y_n \cdot D \), i.e. \( D + y_n \cdot D < \text{fillable void} \). Here \( y_n \) is called the
VOID CREATION FOR MULTIPLE WAVELENGTHS

\( \lambda_1 \)  \hspace{1cm} \lambda_2 \hspace{1cm} \lambda_3

\begin{align*}
0 & \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad (= N \cdot D) \\
\text{void 1} & \quad \text{void 2} \quad \text{gap} \\
\text{time} & \quad \text{delay} = 4 \text{ for packet (1)} \\
\text{delay} = 5 \text{ for packet (2)} & \quad \text{delay} = 3 \text{ for packet (2)} \\
\text{packet (3) lost} & \quad \text{packet (3) not lost (delay \neq 5)} \\
\end{align*}

**Figure 3.** Evolution of the provisional schedule for a single wavelength when the packet size is fixed and equal to the granularity \((B = E[B] = D)\).

\textit{gap threshold}, an algorithm parameter varied in simulations \((0 < y_n < 1 \text{ in steps of 0.01})\) with \(n\) the \textit{horizon index}. The horizon index is defined based on the position of the horizon, i.e. the two specific and consecutive FDLs the horizon is in between: \((n - 1) \cdot D < \text{horizon} < n \cdot D\). By optimizing the gap threshold \(y_n\) for each horizon index, the condition on the gap size can either be more or less strict depending on the position of the horizon.

### 3.2. General sized packets on a single wavelength

To create a void-creating algorithm for different packet distributions the same single wavelength with \(E[B] = D\) is considered. In this setting no fillable voids are created when the packet size is fixed \((B = E[B])\) and each packet is scheduled on the first FDL to the right of the horizon for all void-filling and non-void-filling algorithms. This is different for other packet size distributions in which each void larger than the minimal packet size \((B_{\text{min}})\) has a chance to be filled in the future. As a consequence the performance of non-void-filling and void-filling algorithms is no longer the same in this setting. More specifically, as void-filling algorithms enable voids to be tracked and filled they will typically outperform their non-void-filling counterparts. Among the different void-filling algorithms however, little performance difference exists. This is as the more advanced void-filling algorithms tend to only thrive when there are many voids to choose from to schedule a packet, i.e. in a multiple wavelengths setting. As little performance difference exists with the other void-filling algorithms, we therefore choose to keep consistent with the fixed size packet setting and keep the D-VF algorithm as our benchmark. In this setting, it is thus the D-VF algorithm to which our improved void-creating algorithm will be compared in terms of performance.
Similar to the D-VF algorithm the void-creating algorithm first tries to fill a reachable void with the packet that has to be scheduled. The void closest to the zero delay reference line that overlaps an FDL in such a way that it can accommodate the packet is filled and split in two new voids, one leading and one trailing the newly scheduled packet (void 2 (the circle) for packets smaller than 0.58 in the example of Fig. 2). In order to improve the performance the void-creating algorithm schedules in a different way if no reachable void is present in the system. If no void can be filled it sometimes schedules on the second FDL (triangle) instead of the first FDL (dot) to the right of the horizon. This is opposed to the D-VF algorithm, which always schedules on the first FDL to the right of the horizon. By doing so the void-creating algorithm creates voids that are larger and thus will be reachable more often than voids created by scheduling on the dot. It thus speculates that the filling of this larger void and its descendants (the successive leading- and trailing-voids) will result in an overall denser stacking of the packets.

To decide upon scheduling on the triangle or the dot a more thought through consideration has to be made than in the case of fixed packets size. In the fixed packets size case the chance of filling a void smaller than the packet size \( B = D \) is zero. The value of a void smaller than \( D \) is thus zero, irrespective of its length. The value of a void larger than \( D \) on the other hand rises monotonously with its length: the longer a fillable void is, the more reachable it will be in the future and thus the more value it has. If the difference in value between scheduling on the triangle and the dot is sufficient, the algorithm schedules on the triangle. Note that this theoretical value is not calculated in the void-creating algorithm of [21, 18]. Instead the gap threshold for which the difference in value is large enough is simply figured out by simulation. For other packet sizes however, the theoretical value of the void created by scheduling on the dot does not remain zero as the size of the gap changes. Instead the value of both the dot and triangle voids increase as the gap and thus their length increases. The difference in their value as a function of the size of the gap depends on the exact distribution of the arriving packets.

As the method used in [21, 18] provides little basis for extension towards applications with general packet size distribution we therefore in [22] proposed a method to calculate a theoretical value of a void. The theoretical value assigned to a void is directly linked to the chance of the void being filled in the future and thus its ability to improve performance. In a nutshell, our approach is to confront this theoretical value of a void created by scheduling on the dot and that of a void created by scheduling on the triangle, basing our scheduling decision on the difference between the two. To assign a value to a void we introduced the concepts lifetime and life cycle of a void. A void’s lifetime is defined as the period between the void’s creation and the last moment during which it can be filled, under the hypothesis that no packet arrives. A void’s life cycle \( L_{\text{max}}(t) \) is a function of time that spans the void’s lifetime, indicating for each time instant the maximum packet size allowable for a hypothetically arriving packet in order to still fit in the void. In Fig. 4 the life cycles of both the void by scheduling a packet on the dot and the triangle in Fig. 2 are shown. For both voids the life cycle is, just like for any other void, a sawtooth-shaped function. For the void created by scheduling on the dot, this sawtooth-shaped curve by definition never rises above the gap size, and, in the given example, periodically drops to zero, going through periods of complete unreachability for arriving packets. Complementary to this, the curve in case of scheduling on the triangle remains above zero throughout the void’s entire lifetime.
Although different packet sizes are allowed in [22], all packets arrive according to a Poisson process, i.e. with exponentially distributed inter-arrival times. The average inter-arrival rate $\lambda = 1/E[T]$ can be derived from the chosen load $\rho = E[B]/E[T]$ resulting in $\lambda = \rho/D$ as for all assumed packet size distributions $E[B] = D$. Ignoring the presence of other voids (current as well as future voids), the chance a void, not yet filled, is filled by a packet in the next infinitesimally small time interval $dt$ is given by $\lambda \cdot \text{Prob}[B \leq L_{\text{max}}(t)] \cdot dt$. From a point of view of a void, arrivals of packets that fit into the void occur according to an inhomogeneous Poisson process with arrival rate $\lambda(t) = \lambda \cdot \text{Prob}[B \leq L_{\text{max}}(t)]$. Note that this Poisson process does not describe the packets that actually fill the void but the packets that would fit in the void under the hypothesis that no prior arrival already filled it. The overall expected number of packets to arrive in the life cycle of the void that are able to fill the void ($\Lambda$) is calculated by integrating $\lambda(t)$ over the lifetime of a void.

$$\Lambda = \int_{\text{lifetime}} \lambda(t) dt = \frac{\rho}{D} \int_{0}^{\infty} \text{Prob}[B \leq L_{\text{max}}(t)] dt. \quad (1)$$

Due to the nature of the Poisson process the chance no packet arrives during its lifetime is given by $e^{-\Lambda}$. The overall normalized expected packet size to arrive in the system that is able to fill the void ($\overline{\Lambda}$) is calculated by a similar integral:

$$\overline{\Lambda} = \frac{\rho}{D} \cdot \int_{0}^{\infty} \text{Prob}[B \leq L_{\text{max}}(t)] \cdot \frac{E[B|B \leq L_{\text{max}}(t)]}{D} dt. \quad (2)$$

It is this $\overline{\Lambda}$ that is used as the basis for the void value function. Hereby, the theoretical value of scheduling on the dot will be confronted with that of scheduling on the triangle in terms of a theoretical (added) value function: $\overline{\Lambda}(\triangle) - \overline{\Lambda}(\bullet)$. With the given definition of $\overline{\Lambda}$, $E[B|B \leq L_{\text{max}}(t)]$ was evaluated for a variety of packet size distributions in [22]. A fixed packet scheduled on the dot size yields: $\overline{\Lambda}(\bullet) = 0$.

When scheduling on the second FDL after the horizon (triangle) $\overline{\Lambda}$ becomes:

$$\overline{\Lambda}(\triangle) = \rho \cdot n \cdot g.$$

Where $g$ denotes $\text{gap}/D$ and $n$ denotes the horizon index. A uniform distribution of the packet size on the interval $[0, 2 \cdot D]$ results in the following $\overline{\Lambda}$:

$$\overline{\Lambda}(\bullet) = \frac{\rho}{12} \cdot n \cdot g^3.$$

$$\overline{\Lambda}(\triangle) = \frac{\rho}{12} \cdot [n \cdot (1 + 3g + 3g^2) + 1].$$

Figure 4. Life cycles of the voids created by scheduling on the $\bullet$ and $\triangle$ of Fig. 2.
For a uniform distribution of the packet size on the interval $[0.5 \cdot D, 1.5 \cdot D]$ there are two different cases. If $g \leq 0.5$, $\bar{\Lambda}$ becomes:

$$\Lambda(\bullet) = 0.$$ 

$$\bar{\Lambda}(\bigtriangleup) = \frac{\rho}{24} \cdot \left[n \cdot (2 + 9g + 12g^2 + 4g^3) + 2\right].$$

If $0.5 < g \leq 1$ $\bar{\Lambda}$ becomes:

$$\Lambda(\bullet) = \frac{\rho}{24} \cdot n \cdot (1 - 3g + 4g^3).$$

$$\bar{\Lambda}(\bigtriangleup) = \frac{\rho}{24} \cdot \left[n \cdot (-3 + 27g - 4g^3) + 2\right].$$

For an exponentially distributed packet size this $\bar{\Lambda}$ is obtained:

$$\Lambda(\bullet) = \rho \cdot n \cdot \left[g - 2 + (2 + g) \cdot e^{-g}\right].$$

$$\bar{\Lambda}(\bigtriangleup) = \rho \cdot \left[n \cdot \left(1 + e^{-g} \cdot \left(\frac{3 + g}{e} - g - 2\right)\right) + \frac{3}{e} - 1\right].$$

Based on these results a void value threshold algorithm can be created for which void creation is employed if the difference in theoretical void value between scheduling on the dot and the triangle ($\bar{\Lambda}(\bigtriangleup) - \bar{\Lambda}(\bullet)$) is larger than a certain threshold.

### 3.3. Fixed size packets on multiple wavelengths.

To extend the analysis of [22] to multiple wavelengths a setting with fixed size packets $B = D$ is considered. Although the D-VF algorithm was the benchmark algorithm in the single wavelength setting this has to be reassessed for the multiple wavelength setting. More specifically as multiple wavelengths are present, scheduling according to an algorithm that focuses on the gaps in between packets can significantly improve performance. However as in the fixed size packets setting no fillable voids are created, the algorithm will only have to choose which wavelength to join at the back. With this reduced set of options, all gap focused algorithms will schedule an arriving packet at the back of the wavelength resulting in the smallest starting void. We therefore choose the G-VF (gap void-filling) algorithm as the benchmark algorithm to which performance of our void-creating algorithm will be compared. This G-VF algorithm is named according to the same scheme as in [18, 19, 20, 22, 21] but is equivalent to most variation of the min-SV and LAUC-VF algorithms used in other literature. The operation of this algorithm is illustrated in the example of Fig. 5, showing a provisional schedule for a setting with $c = 4$ wavelengths. Similar to the provisional schedule for a single wavelength setting in Fig. 2 the vertical lines represent the delays of the FDLs ($j = 0 \ldots N = 5$) and the granularity in the example is assumed as unity ($D = B = 1$) to ease notation. For a multiple wavelength setting each outgoing wavelength ($i = 1 \ldots c = 4$) is represented horizontally. In this case the G-VF algorithm will schedule an arriving packet on the first wavelength (position $a$), minimizing the gap in between the new packet and the already scheduled packet in front of it. Although a lower delay could be chosen for this packet by scheduling it on the second or fourth wavelength (position $a$ or $d$), making the stacking more dense by minimizing the gap improves the overall performance in the long run.

As our reference algorithm for this setting is the G-VF algorithm, our void-creating algorithm will be based on this algorithm as well. The choices and order in which this void-creating algorithm operates are, according to the authors, done in the most logical and simple way. Although different choices can be made, they will,
VOID CREATION FOR MULTIPLE WAVELENGTHS

Figure 5. An example of a provisional schedule for fixed packet size and multiple wavelengths \((c = 4)\). The G-VF algorithm will choose to minimize the gap by scheduling on position \(a\).

Figure 6. An example of a provisional schedule for fixed packet size and multiple wavelengths \((c = 4)\) resulting in a possible void creation on the second wavelength.

at least in this setting, make little difference in the overall obtainable performance improvement. A first difference for the void-creating algorithm compared to the G-VF algorithm is that fillable voids, created by the void creating mechanism upon arrivals of previous packets, can be present. When a single or multiple fillable voids are present, the void-creating algorithm chooses to fill the void that results in the lowest delay for the packet. This procedure is sustained even if a lower gap can be obtained by filling a different void or by joining a wavelength at the back. The reasoning behind this is that an artificially created void is only advantageous when it is actually filled afterwards. Moreover, as these voids only have a limited lifetime, the void set to expire the quickest (corresponding to the lowest delay) should get filled first. When no fillable voids are present, and the algorithm thus has to decide which wavelength to join at the back, the void creating algorithm, just like the G-VF algorithm, chooses to schedule on the wavelength resulting in the smallest gap. Once, based on this criterion, the wavelength is chosen, the void creating algorithm now decides upon void creation based on the void value function. Just like in the setting of a single wavelength this value function is based on the overall normalized expected packet size to arrive in the system that is able to fill the void \((\bar{\Lambda})\). Hereby, the theoretical value of scheduling on the dot is confronted with that of scheduling on the triangle in terms of a theoretical (added) value function: \(\bar{\Lambda}(\triangle) - \bar{\Lambda}(\bullet)\). If this difference in theoretical void value between scheduling on the dot and triangle is larger than a certain threshold, the algorithm decides to create a larger void by scheduling on the triangle position. In the example of Fig. 6 void creation thus corresponds to scheduling on position \(c\) instead of position \(b\) when this is triggered by the value function. Note that in this case the void-creating position \(c\) is also preferred over both positions \(d\) and \(e\) with the same delay.
By using the same value function as in the single wavelength setting, the void creating algorithm does not take into account any other voids already present in the system, also, and especially, not those on other wavelengths. Extending the algorithm to take into account the value of other voids could be useful but should be done wisely. Indeed, with the threshold value the algorithm still has a parameter that can tune the algorithm to be more or less strict in creating voids, possibly resulting in a shift of the optimal threshold parameter when values of other wavelengths are taken into account in the value function. Extensively investigating the inclusion of the values of others voids in the value function is thus considered out of the scope of this paper and is kept for future work.

To evaluate the performance of the proposed void-creating algorithm we employ Monte Carlo simulation. Specifically, the algorithm is programmed in Matlab using a discrete event simulation (DES) in a similar way as in [21, 18]. In a DES, the system is modelled as a sequence of events marked by their particular instant in time. The system state changes from one event to the next and does not change in-between. This is as opposed to continuous simulation in which time is broken into small time slices. Because DES simulations do not simulate every time slice, they are far more efficient in terms of computational resources.

4. Performance results.

4.1. General sized packets on a single wavelength. For the single wavelength setting we first simulate the system without void creation for all packet size distributions, i.e. all packets are scheduled with the D-VF algorithm, our reference algorithm to which the performance of our void creating algorithm will be compared. The number of FDLs is assumed ten \( (N + 1 = 10) \), \( D = E[B] \) and the number of simulated packets is \( 10 \cdot 10^7 \) as this yields confidence intervals narrow enough to draw reliable conclusions. The performance measures are summarized in Table 1 for a load of 60 % and a load of 80 %.

![Table 1](image.png)

Table 1. Performance measures of D-VF for different packet size distributions and a single wavelength.

In the void value threshold algorithm void creation is employed if the difference in theoretical void value between scheduling on the dot and the triangle \( (\bar{\Lambda}(\bigcirc) - \bar{\Lambda}(\bullet)) \) is larger than a certain threshold. Table 2 shows the optimal thresholds and performance improvement (both obtained by simulation) for each combination of performance measure, load and packet size distribution. The obtainable improvements for the fixed packet size from Table 2 are (up to 5 %) larger than the improvements obtained in [21, 18]. The void value threshold algorithm, as opposed to the algorithm from [21, 18], can thus be used for different packet size distributions and moreover outperforms the algorithm from [21, 18] under the same circumstances. From Table 2 it is clear however that obtainable performance improvement strongly
TABLE 2. Optimal thresholds and corresponding performance improvements of the void value threshold algorithm for a single wavelength.

| Optimal threshold | Fixed $B = E[B] = D$ | Exponential $E[B] = D$ | Uniform on $[0, 2D]$ | Uniform on $[0.5D, 1.5D]$ |
|-------------------|-----------------------|------------------------|----------------------|-------------------------|
| $\rho = 0.6$     | $p = 0.6$             | $p = 0.6$              | $p = 0.6$            | $p = 0.6$               |
| $\rho = 0.8$     | $p = 0.8$             | $p = 0.8$              | $p = 0.8$            | $p = 0.8$               |

|        | LP  | LPlength | Packet delay | Packet gap |
|--------|-----|----------|--------------|------------|
| $p = 0.6$ | 1.0 | 1.2      | 1.1          | 1.4        |
| $p = 0.8$ | 1.2 | 1.0      | 1.4          | 1.6        |
| $p = 0.6$ | 1.0 | 1.2      | 1.1          | 1.4        |
| $p = 0.8$ | 1.2 | 1.0      | 1.4          | 1.6        |

| Maximum gain | Fixed $B = E[B] = D$ | Exponential $E[B] = D$ | Uniform on $[0, 2D]$ | Uniform on $[0.5D, 1.5D]$ |
|--------------|-----------------------|------------------------|----------------------|-------------------------|
| $\rho = 0.6$ | $p = 0.6$             | $p = 0.8$              | $p = 0.6$            | $p = 0.8$               |
| $\rho = 0.8$ | $p = 0.8$             | $p = 0.6$              | $p = 0.8$            | $p = 0.6$               |

|        | LP  | LPlength | Packet delay | Packet gap |
|--------|-----|----------|--------------|------------|
| $p = 0.6$ | 54.1% | 36.1%   | 4.5%         | 6.4%       |
| $p = 0.8$ | 54.1% | 36.1%   | 0.2%         | 0.1%       |
| $p = 0.6$ | 54.1% | 36.1%   | 0.6%         | 1.9%       |
| $p = 0.8$ | 54.1% | 36.1%   | 0.6%         | 1.9%       |

4.2. Fixed size packets on multiple wavelengths. Similar to the single wavelength case we first simulate the system without void creation, i.e. all packets are scheduled with the G-VF algorithm as defined above, our reference algorithm to which the performance of our void creating algorithm will be compared. The number of FDLs is assumed ten ($N + 1 = 10$), $D = E[B] = B$ and the number of
simulated packets is $10 \cdot 10^7$ as this yields confidence intervals narrow enough to draw reliable conclusions. Both the number of wavelengths ($c = 1, 2, 4, 6$ and $8$) and the load are varied ($\rho = 60\%$, $80\%$ and $90\%$) in simulations. As even lower values of the load and a higher number of wavelengths would require a too a high number of simulated packets and accompanying long simulation time, the simulated values are limited to combinations of these sets. For all combinations of these values the performance measures (LP, LPsize, delay, gap) are determined. Table 3 shows the results for all performance measures and combinations of parameters (load and number of wavelengths) for the G-VF algorithm. In Table 3 an ‘x’ indicates when too little packets (or even none at all) are lost, resulting in unreliable and thus omitted results for this parameter combination.

Table 3. Performance measures of G-VF for a fixed packet size distribution and a varying number of wavelength ($c = 1, 2, 4, 6$ and $8$).

| G-VF | c=1 | c=2 | c=4 | c=6 | c=8 |
|------|-----|-----|-----|-----|-----|
| $\rho=0.6$ | $\rho=0.8$ | $\rho=0.9$ | $\rho=0.6$ | $\rho=0.8$ | $\rho=0.9$ | $\rho=0.6$ | $\rho=0.8$ | $\rho=0.9$ |
| LP | 2.1 | 14.5 | 22.5 | 0.0 | 0% | 63.8 | 0.1% | 5.7% | x | 0.0% | 1.7% | x | x | 0.3% |
| LP_length | 2.1 | 14.5 | 22.5 | 0.0% | 63.8 | 0.1% | 5.7% | x | 0.0% | 1.7% | x | x | 0.3% |
| Packet delay | 3.0 | 6.1 | 7.1 | 1.0 | 4.9 | 7.1 | 0.4 | 1.9 | 6.2 | 0.2 | 1.1 | 4.5 | 0.1 | 0.8 | 2.8 |
| Packet gap | 0.36 | 0.42 | 0.42 | 0.13 | 0.25 | 0.28 | 0.03 | 0.08 | 0.16 | 0.01 | 0.04 | 0.09 | 0.01 | 0.02 | 0.05 |

Similar as Table 2 does for a single wavelength setting, Table 4 shows the optimal thresholds and performance improvements (both obtained by simulation) for each performance measure, load and number of wavelengths. Note that rather than actual performance measure values Table 4 thus shows performance improvements relative to the performance measure values from Table 3. Whereas in Table 3 it is natural that performance measures decrease with an increasing number of wavelengths and decreasing load, this, although observed, is not necessarily true for the values in Table 4 as they are mere relative performance improvements. Similarly as in Table 3 an ‘x’ indicates when too little packets (or even none at all) are lost, resulting in unreliable and thus omitted results for the LP and LP_length values. An $\infty$ symbol in Table 4 (corresponding to a 0% obtainable performance improvement) on the other hand marks the cases when no improvement can be achieved by the void creating algorithm. This results in never applying void creation and thus an infinitely high value threshold in the void-creating value threshold algorithm.

From Table 4 it is clear that the optimal threshold decreases as the number of wavelengths ($c$) increases. This continues up until a certain number wavelengths at which point no performance improvement is achieved by the void-creating algorithm and the optimal threshold thus jumps to infinity. This is also clear in the graphs of Fig. 7, which show the maximum gain in performance improvement in relation to the number of wavelengths ($c$) and the load. Although not a direct consequence from the changing parameters, the obtainable performance improvements in general do tend to decrease with an increasing number of wavelengths and lower load. This decrease however is not always monotonically as for some performance measures and values of the load an intermediate maximum in obtainable performance improvement can be observed.
(a) Optimal thresholds

|                | c=1 | c=2 | c=4 | c=6 | c=8 |
|----------------|-----|-----|-----|-----|-----|
| **optimal** threshold |      |     |    |     |     |
| **ρ**=0.6      | 0.9 | 1.2 | 1.3 | 0.7 | 1.0 | 0.9 |
| **ρ**=0.8      | p=0 | p=0 | p=0 | p=0 | p=0 | p=0 |
| **ρ**=0.9      | x   | x   | x   | 2.3 | 0.5 | x   |
| **ρ**=0.6      | x   | x   | x   | 1.4 |     |     |

(b) Corresponding performance improvements

|                | c=1 | c=2 | c=4 | c=6 | c=8 |
|----------------|-----|-----|-----|-----|-----|
| **maximum** gain |      |     |     |     |     |
| **LP** 53.5%  | 50.9% | 47.2% | 43.6% | 26.6% | 22.1% |     |
| **LPLength** 53.5% | 50.9% | 47.2% | 43.6% | 26.6% | 22.1% |   |
| **Packet delay** | 16.1% | 16.1% | 12.9% | 0% | 0% | 8.3% |
| **Packet gap** | 11.4% | 22.7% | 24.8% | 3.5% | 5.2% | 14.3% |

Table 4. Optimal thresholds and corresponding performance improvements of the void value threshold algorithm for a fixed packet size distribution and a varying number of wavelength (c = 1, 2, 4, 6 and 8).

From these results it is clear that the void-creating algorithm, using the same threshold based method, can indeed be applied on multiple wavelength settings for a fixed packet size. The resulting performance improvements, which are more than significant, highlight that a basic extension of the algorithm already leads to significant performance gain. Although gap focused algorithms as G-VF operate fundamentally different than delay-based algorithms in this setting, our void creating algorithm still manages to improve performance in the same order of magnitude. This clearly shows that void creation is not a competing mechanism for more advanced void filling and can thus be used mostly in a complementary manner, to improve the performance of a variety of algorithms. In this setting the performance of this algorithm however does tend to decrease with an increasing number of wavelengths and decreasing load. In the settings that were studied, performance improvement seemed to decrease along with the (absolute) value of the performance measure considered. Whether this observation also holds true for different packet size distributions is a question yet to be answered. While it may be reasonably expected that taking into account the values of other voids in the value function might improve performance, such considerations are open research questions and part of intended future work.

5. Conclusions. In this paper we proposed a new void-creating algorithm which can significantly improve the performance of existing void-filling algorithms for different packet size distributions in a single wavelength setting. Based on the current system state the algorithm decides to create a fillable void or not based on a method that uses a threshold on the added void value. Using Monte Carlo simulation, this
algorithm was validated for a variety of settings and parameter variations. It was shown that this algorithm can achieve performance improvements in all performance measures, i.e. packet loss, packet size loss, packet delay and packet gap. The achievable improvement depends on the packet size distribution and, in general, decreases as the variance of the packet size distribution increases. It was shown that our choice of theoretical void values makes it possible to decide upon void creation once
the packet size distribution, performance measure and load are known. The result is a scheduling algorithm that is as simple to implement as a classic scheduling algorithm, deciding upon void creation using a closed-form expression derived from the packet size distribution. The void value threshold algorithm outperforms existing void-filling and void-creating algorithms and, above all, can be used for different packet size distributions. Moreover it was shown that the use of this algorithm for a fixed packet size can be extended to settings with multiple wavelengths with only little modifications. The original void creating mechanism thus proves itself to be subtle and effective enough by design to be extended to other settings in a rather straightforward way. Particularly, the adapted algorithm performs very well for a modest number of wavelengths like 2 or 4. As the obtainable performance improvement for the adapted algorithm does tend to decrease with an increasing number of wavelengths of 6,8 or more, future work should focus on extending the void-creating algorithm to take into account the values of other voids in the threshold value function. Moreover, as the threshold value function is already based on a semi-analytical calculation of the future probability of the void being filled, an extension of this value function could lead to a deepened analytical insight, and possibly completely analytical solution of the performance of the proposed algorithm.

Acknowledgement. Part of this research has been funded by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office.

REFERENCES

[1] E. Burmeister, D. Blumenthal and J. Bowers, A comparison of optical buffering technologies, *Optical Switching and Networking*, 5 (2008), 10–18.
[2] F. Callegati, A. Campi and W. Cerroni, Fast and versatile scheduler design for optical packet/burst switching, *Optical Switching and Networking*, 8 (2011), 93–102.
[3] F. Callegati, W. Cerroni and G. S. Pavani, Key parameters for contention resolution in multifiber optical burst/packet switching nodes, in *Proceedings of Broadnets 07*, Raleigh, 2007.
[4] M. Chen, H. Jin, Y. Wen and V. C. M. Leung, Enabling technologies for future data center networking: A primer, *IEEE Network*, 27 (2013), 8–15.
[5] L. G. Dizaji and A. G. Rahbar, Efficient integration of switching mechanisms in all-optical networks, in *proceedings of the 8th International Symposium on Telecommunications (IST)*, 2016, 40–44.
[6] K. Dolzer, C. Gauger, J. Späth and B. Stefan, Evaluation of reservation mechanisms for optical burst switching, *AEU - International Journal of Electronics and Communications*, 55 (2001), 18–26.
[7] D. H. Hailu, G. G. Lema, E. A. Yekun and S. H. Kebede, Unified study of quality of service (QoS) in OPS/OBS networks, *Optical Fiber Technology*, 36 (2017), 394–402.
[8] L. Krull, World data transfer record back in Danish hands, 2014, [http://www.dtu.dk/english/news/2014/07/verdensrekord-1-dataoverfoersel-paa-danske-haender-igen?id=bed76c33-c9da-4214-91f3-c9edf8a0e24](http://www.dtu.dk/english/news/2014/07/verdensrekord-1-dataoverfoersel-paa-danske-haender-igen?id=bed76c33-c9da-4214-91f3-c9edf8a0e24).
[9] M. Nandi, A. K. Turuk, D. K. Puthal and S. Dutta, Best fit void filling algorithm in optical burst switching networks, in *proceedings of the Second International Conference on Emerging Trends in Engineering Technology*, 2009, 609–614.
[10] W. Rogiest, J. Lambert, D. Fiems, B. V. Houdt, H. Bruneel and C. Blondia, A unified model for synchronous and asynchronous FDL buffers allowing closed-form solution, *Performance Evaluation*, 66 (2009), 343–355.
[11] W. Rogiest, D. Fiems and J.-P. Dorsman, Analysis of fibre-loop optical buffers with a void-avoiding schedule, in *Proceedings of Valuetools 2014*, 15 (2015), e5.
[12] L. Tancevski, L. Tamil and F. Callegati, Nondegenerate buffers: An approach for building large optical memories, *IEEE Photonics Technology Letters*, 11 (1999), 1072–1074.
[13] H.-L. To, S.-H. Lee and W.-J. Hwang, A burst loss probability model with impatient customer feature for optical burst switching networks, *International Journal of Communication Systems*, 28 (2015), 1729–1740.
[14] A. Triki, I. Popescu, A. Gravey, X. Cao, T. Tsuritani and P. Gravey, TWIN as a future-proof optical transport technology for next generation metro networks, in proceedings of the 17th IEEE International Conference on High Performance Switching and Routing (HPSR), 2016, 87–92.

[15] R. Tucker, Scalability and energy consumption of optical and electronic packet switching, Journal of Lightwave Technology, 29 (2011), 2410–2421.

[16] J. S. Turner, Terabit burst switching, Journal of High Speed Networks, 8 (1999), 3–16.

[17] T. van der Vorst, Brennenraedts, D. van Kerkhof and R. Bekkers, Fast Forward: How the speed of the Internet will develop between now and 2020, Technical Report 2013.048-1262, Dialogic, Utrecht, 2014.

[18] K. Van Hautegem, W. Rogiest and H. Bruneel, Fill the void: Improved scheduling for optical switching, in proceedings of the 27th International Teletraffic Congress (ITC 27), 2015, 82–88.

[19] K. Van Hautegem, W. Rogiest and H. Bruneel, OPS/OBS scheduling algorithms: Incorporating a wavelength conversion cost in the performance analysis, in Proceedings of the 32nd IEEE International Performance, Computing, and Communication Conference (IPCCC), San Diego, 2013.

[20] K. Van Hautegem, W. Rogiest and H. Bruneel, Scheduling in optical switching: Deploying shared wavelength converters more effectively, in Proceedings of the 2014 IEEE International Conference on Communications (ICC), Sydney, 2014.

[21] K. Van Hautegem, W. Rogiest and H. Bruneel, Void-creating algorithm in OPS/OBS: Mind the gap, AIP Conference Proceedings, 1648 (2015), 170002.

[22] K. Van Hautegem, W. Rogiest and H. Bruneel, Optical switching for variable size packets: Improved void filling through selective void creation, in Proceedings of the 2016 International Conference on Queueing Theory and Network Applications (QTNA), Wellington, 2016.

[23] J. Xu, C. Qiao, J. Li and G. Xu, Efficient channel scheduling algorithms in optical burst switched networks, in proceedings of the 22nd IEEE INFOCOM, 3 (2003), 2268–2278.

[24] Y. Yan, G. M. Saridis, Y. Shu, B. R. Rofoee, S. Yan, M. Arslan, T. Bradley, N. V. Wheeler, N. H.-L. Wong, F. Poletti, M. N. Petrovich, D. J. Richardson, S. Poole, G. Zervas and D. Simeonidou, All-optical programmable disaggregated data centre network realized by FPGA-based switch and interface card, Journal of Lightwave Technology, 34 (2016), 1925–1932.

Received for publication August 2017.