Batch Scheduling Algorithms for Optical Burst Switching Networks

Ayman Kaheel and Hussein Alnuweiri

University of British Columbia, Vancouver BC V6T 1Z4, Canada
{aymank, hussein}@ece.ubc.ca

Abstract. Previously proposed wavelength scheduling algorithms in optical burst switching networks process each reservation request individually and in a greedy manner. In this paper we propose a new family of wavelength scheduling algorithms that process a batch of reservation requests together instead of processing them one by one. When a control burst with a reservation request arrives to a free batch scheduler, the scheduler waits for a small amount of time, called the *acceptance delay*, before deciding to accept or reject the reservation request. After the acceptance delay has passed, the scheduler processes all the reservation requests that have arrived during the acceptance delay, then it accepts the requests that will maximize the utilization of the wavelength channels. We describe an optimal batch scheduler that serves as an upper bound on the performance of batch scheduling algorithms. Furthermore, we introduce two heuristic batch scheduling algorithms. The performance of the proposed algorithms is evaluated using a discrete-event simulation model. Simulation results suggest that batch schedulers could decrease the blocking probability by 25% compared to the best previously known wavelength scheduling algorithm.

1 Introduction

Optical Burst Switching (OBS) [1] has been proposed as an optical switching technique that combines the advantages of both Wavelength-Routed (WR) networks and optical packet switching networks. As in WR networks, there is no need for buffering and electronic processing for data at intermediate nodes. At the same time, OBS increases the network utilization by reserving the optical channel for a limited time period. The basic switching entity in OBS is a burst. A burst is a train of packets moving together from one ingress node to one egress node and switched together at intermediate nodes. An optical burst consists of two parts, a header and a data burst. The header is called the control burst (CB) and is transmitted separately from the data, which is called the data burst (DB). The CB is transmitted first on a separate signaling channel to reserve the bandwidth along the path for the corresponding DB. After a given delay, the CB is followed by the DB, which travels over the same path reserved by the CB. The delay between sending the CB and the DB is called the *burst offset time*. The value of the offset time is chosen to be greater than or equal to the total

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 90–101, 2005.

processing delay encountered by the CB. Consequently, no buffering is needed for the DB at intermediate nodes.

Several signaling protocols have been proposed for OBS [1,2]. Qiao et. al have proposed a protocol called Just-Enough-Time (JET) [1]. The JET protocol is reserve-a-fixed duration scheme that reserves resources exactly for the transmission time of the burst. In JET, the CB contains the destination address, the DB length, the wavelength on which the associated DB will arrive, and the burst offset time. When a CB arrives at a core node, a wavelength scheduling algorithm is invoked to find a suitable wavelength channel on the outgoing link for the corresponding DB. The wavelength channel is reserved for a duration equal to the burst length starting from the arrival time of the DB. The information required by the scheduler such as the burst arrival time and its duration are obtained from the CB. The scheduler keeps track of the availability of free time intervals (voids) on every wavelength channel.

The absence of the concept of "packet queues" in OBS nodes, coupled with the one way nature of the JET reservation protocol, drives the blocking probability to become the main performance measure in OBS networks. The burst blocking probability is the probability that a wavelength reservation request will not be granted due to the unavailability of a free wavelength. The blocking probability at an OBS node depends to a certain degree on how efficiently can the wavelength scheduling algorithm handle voids on wavelength channels. This fact has led to a growing interest in the area of wavelength scheduling in OBS networks. A number of wavelength scheduling algorithms have been proposed in the literature (for a survey see [3]).The main differentiating factor between previously proposed algorithms is the wavelength selection criteria. If there is more than one wavelength that can be assigned to the incoming burst, each algorithm selects a wavelength according to different criteria.

In this paper we propose a new family of wavelength schedulers. The proposed schedulers process a batch of bursts together instead of processing them one by one as in the case of previously proposed algorithms. The rest of this paper is organized as follows. Section 2 gives a brief overview of wavelength scheduling algorithms. We introduce the batch scheduling class of schedulers in Section 3. Section 4 describes the optimal batch scheduler. In Section 5 we introduce two new heuristic batch scheduling algorithms. The proposed algorithms are evaluated in Section 6 using a discrete event simulation model. We conclude the paper in Section 7.

2 Wavelength Scheduling

In OBS networks employing the JET protocol (OBS-JET networks), a CB arriving at a node represents a wavelength reservation request. The reservation request consists of a pair of values (t_a, t_e). The value t_a is determined based on the offset time, and t_e is determined based on t_a and the length of the burst b, as shown in Figure 1. A wavelength scheduling algorithm X is more efficient than wavelength scheduling algorithm Y, if it increases the utilization of the

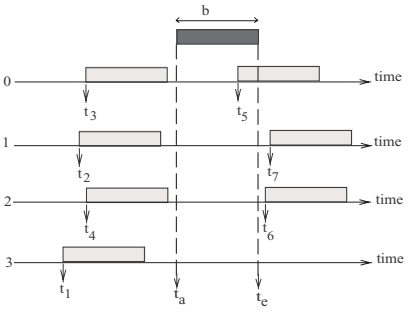

Fig. 1. Wavelength Scheduling

wavelength channels, and hence decreases the burst blocking probability. The blocking probability is calculated as the ratio of bits blocked to bits sent.

A wavelength channel is said to be unscheduled at time t when no burst is using the channel at or after time t. A channel is said to be unused for the duration of voids between successive bursts and after the last burst assigned to the channel.

Xiong *et* al. [4] proposed a channel algorithm called *LAUC-VF* (Latest Available Unused Channel with Void Filling). The basic idea of the LAUC-VF algorithm is to minimize voids by selecting the latest available unused data channel for each arriving DB. Given the arrival time t_a of a DB with duration b to the optical switch, the scheduler first finds the outgoing data channels that are available for the time period of $(t_a; t_a + b)$. If there is at least one such data channel, the scheduler selects the latest available data channel, i.e., the channel having the smallest gap between t_a and the end of the last DB just before t_a. The LAUC-VF algorithm is widely considered to be among the best wavelength scheduling algorithms in terms of its blocking probability.

3 Batch Scheduling

Previously proposed wavelength schedulers are considered to be greedy algorithms. They are greedy in the sense that they consider every request individually, and make the choice that looks best at the moment. Therefore, if the request can be accepted (unblocked), it will be indeed accepted. When a CB arrive to an OBS node employing a greedy wavelength scheduling algorithm, it is inserted into a FIFO queue. When the wavelength scheduler is free, it dequeues the first packet on the queue and processes it. Figure 2 describes this process.

In this work we pose the following question: What if we defer the acceptance of some unblocked requests until we see more requests? The intuition behind this question is that it may be advisable to reject an initially unblocked request if later requests will make better use of the wavelength channels.

To further illustrate the idea. Let the acceptance delay be d. Consider the case of $d = \infty$. In this case the system will wait until all reservation requests

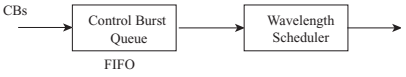

Fig. 2. Control burst scheduling

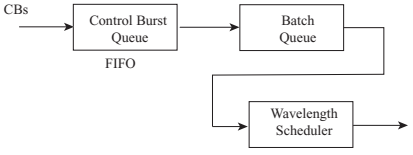

Fig. 3. Control burst scheduling with batch queue

have been received, then it will select requests that will maximize the utilization. Obviously the case of $d = \infty$ is impractical. Our proposal is to use an acceptable value of d, such that we schedule a batch of reservation requests together, instead of using $d = 0$ as in the case of greedy algorithms.

To implement the above concept, we modify the burst scheduling process by inserting a queue before the wavelength scheduler and after the CB queue, we call it the *batch queue*, as shown in Figure 3. When a CB arrives to an OBS node, it gets inserted into the CB queue. When the wavelength scheduler becomes free, it will wait for a small amount of delay, viz. the *acceptance delay* d, then it moves all bursts that are in the CB queue to the batch queue, then processes all the CBs in the batch queue together instead of the previously used greedy approach.

The benefit of the batch queue, as will be seen later, is to limit the delay that a CB can incur during the batch processing.

3.1 Definitions

A graph is $G = (V, E)$ consists of a finite set V of elements called *vertices*, and a set E of pairs of vertices called *edges*. Let $V(G)$ represent the vertex set of G, and $E(G)$ represent the edge set of G. We call $adj(v)$ the *adjacency set* of vertex v, and we call the pair (v, w) an edge. Clearly $(v, w) \in E$ if and only if $w \in adj(v)$.

A subgraph H of G, denoted $H \subset G$, is a graph with $V(H) \subset V(G)$ and $E(H) \subset E(G)$. The degree of v in the subgraph H for any $v \in V(H)$, denoted $deg(v|H)$, is the number of vertices of H adjacent to v, i.e. $deg(v|H) = |adj(v|H)|$. An undirected graph G is called an *interval graph* if its vertices can be put into one-to-one correspondence with a set of intervals on the real line, such that two vertices are connected by an edge of G if and only if their corresponding intervals have nonempty intersection (have a common point). In other words, G is an interval graph provided that one can assign to each $v \in V(G)$ an interval I_v such that $I_u \cap I_v$ is nonempty if and only if $(u, v) \in E$.

An interval is defined by two points: left point (start of the interval) and right point (end of the interval). Let $l(I)$ and $r(I)$ correspond to the left and right

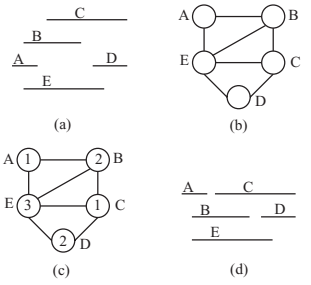

Fig. 4. Mapping batch scheduling to graph coloring

points of interval I respectively. Intervals I_v and I_u overlap if $l(I_v) \leq l(I_u) < r(I_v)$ or if $l(I_u) \leq l(I_v) < r(I_u)$.

A subgraph is called a *clique*, if every pair of vertices in the subgraph is connected by an edge. A clique C is maximal if there is no clique of G that properly contains C as a subset. Let $size(C)$ be the number of vertices in the clique, the maximum clique is the clique of greatest size among all maximal cliques.

One of the most important applications of interval graphs is job scheduling. Consider a set of n jobs to be scheduled on k servers. Finding a feasible schedule is equivalent to finding a proper k-coloring of the corresponding interval graph, such that no two adjacent vertices can have the same color (overlapping jobs are assigned to different servers). Interval graphs and graph coloring problems have been studied intensively in the literature (see [5,6] and references within).

Batch scheduling in OBS networks can be treated as a subset of the interval graph coloring problem. We have a set of DBs that we want to assign to a number of wavelength channels. Each DB corresponds directly to an interval on the real line, and assigning wavelength channels to bursts is similar to assigning colors to intervals, where no two overlapping bursts can be assigned to the same wavelength channel. Figure 4 gives an example for mapping batch scheduling to graph coloring. In Figure 4a, a batch consisting of five bursts to be assigned to three wavelength channels. Figure 4b shows the corresponding interval graph. In Figure 4c, the interval graph is colored using three colors: 1, 2 and 3. Figure 4d shows the wavelength assignment of the batch.

Each CB represents a reservation request for an interval. The reservation request for DB b_i specifies a start time of the reservation $l(b_i)$, corresponding to the arrival time of the DB, and an end time of the reservation $r(b_i)$. The weight $w(b_i)$ of burst b_i is equal to $(r(b_i) - l(b_i))$.

4 Optimal Batch Scheduling Algorithm

The batch scheduling problem can be stated as follows: Given k wavelengths and a set of n bursts $\{b_1, .., b_n\}$ in the batch, find a feasible schedule that maximizes the value of the DBs accepted on the k wavelength channels. Note that n varies

from one batch to another depending on the arrivals to the CB queue. The value of the accepted bursts is simply the sum of their weights. The weight of the burst is defined as the length of its corresponding interval.

A number of algorithms exist in the literature for finding an optimal feasible schedule of intervals while maximizing the total weight of the selected intervals [6, 7]. In [6], the authors formulated the problem as an instance of the interval graph coloring problem as explained next. Given k colors and n intervals, the objective is to maximize the value of the legally colored graph. The problem is then reformulated as the following binary integer linear program (ILP):

$$\begin{aligned} &\text{Maximize } w^T x \\ &\text{subject to } Ax \leq e^T k, \\ &\qquad\qquad x \in \{0,1\}, \end{aligned}$$

where w is an n-vector representing the weights of the intervals, x is a binary n-vector in which $x_j = 1$ implies that interval j is to be selected, $x_j = 0$ otherwise. A is $m \times n$ clique matrix [8], in which m is the number of the maximal cliques in the interval graph, where $A_{ij} = 1$ if interval j is in the i^{th} maximal clique, and $A_{ij} = 0$ otherwise. Also e is an m-vector of 1's. ILP problems are generally NP-hard. However, for our particular problem the A matrix has the *consecutive ones* property, and A is thus *totally unimodular* [9]. Therefore, we can relax the integrality constraint and solve the problem as a linear program in polynomial time.

This ILP formulation can be used to find an offline optimal schedule of bursts after receiving all the reservation requests. This optimal schedule can serve as a reference against which the performance of other algorithms can be compared.

Although it is tempting to use the same ILP formulation for finding the optimal schedule for a batch of bursts, it is not possible. The reason is that zero of more of the n bursts may not be assignable to one or more of the k wavelength channels due to already existing assignments of bursts belonging to previous batches. This constitute extra constraints in the problem.

To find an optimal batch schedule, we formulate our problem as an instance of the *scheduling with non-identical machines problem* [7], in which we associate with each interval (burst) a subset of servers (wavelength channels) on which it can be processed. Some bursts can not be processed on any wavelength channel, because all are busy in the time intervals corresponding to these bursts, these bursts will be dropped and not included in the optimization process. The authors in [7] have given an algorithm that finds the optimal feasible schedule for the non-identical machines problem. The algorithm works as follows. First build a list of events corresponding to the start and end times of the intervals being optimized. Then, for each event construct the vertices corresponding to legal combination of intervals and servers at the time of the event. The constructed vertices are used to build a directed graph. It then can be shown that the interval assignments along the longest path in the directed graph represent the optimal interval assignment to servers.

Unfortunately, the above algorithm has a $O(n^{k+1})$ computational complexity, which is high when k is large, rendering this algorithm impractical for use as an online batch scheduler. However, the algorithm is useful for comparison purposes since it serves as an upper bound on the performance of batch scheduling algorithms in terms of blocking probability.

5 Heuristic Batch Scheduling Algorithms

Because finding the optimal batch schedule has high computational complexity, we have to turn our attention to heuristic algorithms. In this section we propose a number of batch heuristic algorithms for wavelength channel scheduling in OBS networks. All of the proposed heuristic algorithms are based on performing the following two steps:

1. Impose a certain linear order on the control bursts in the batch queue.
2. Traverse the bursts in this linear order, and assign the corresponding data bursts to wavelength channels using a greedy void filling wavelength scheduling algorithm.

Although the ordering process imposes a small additional delay on the bursts in the batch queue, this delay is limited since the number of bursts in the batch queue is bounded by the acceptance delay value. If we were to order the bursts directly in the FIFO queue shown in Figure 2, then it is possible that a CB will be delayed past the arrival time of the corresponding DB because the number of bursts involved in the ordering process will not be bounded, and this shows the importance of the batch queue.

In the following we propose two batch ordering algorithms: *Smallest-Last Vertex Ordering,* and *Maximal Cliques First Ordering.*

5.1 Smallest-Last Vertex Ordering (SLV)

The SLV heuristic algorithm orders the vertices of the interval graph according to the smallest-last ordering [5]. The vertices $v_1, v_2, \ldots, v_n$ of a graph are said to be in smallest last order whenever v_i has minimum degree in the maximal subgraph on the vertices $v_1, v_2, \ldots, v_i$ for all i. Let $\delta(H) = min_{v \in V(H)}\{deg(v|H)\}$ be the minimum degree of graph H. A formal description of the SLV algorithm is given in Algorithm 1.

The SLV algorithm works as follows. Let v_n be chosen to have minimum degree in G. For $i = n-1, n-2, \ldots, 2, 1$, let v_i be chosen to have minimum degree in $H_i = \langle V(G) - v_n, v_{n-1}, \ldots, v_{i+1} \rangle$. From the resulting sequence $v_1, v_2, \ldots, v_n$, where v_n has the minimum degree in G, vertex v_1 will be colored first. Which means that the burst corresponding to v_1 will be assigned first to the first available wavelength. The process repeats until each burst is either assigned to a wavelength channel or dropped.

Choosing the largest degree vertices first for coloring would minimize the total number of colors required to produce proper coloring of the graph. This means

input : G on n vertices
output : An ordering $v_1, v_2, \ldots, v_n$ of vertices of G, where $deg(v_i|H_i) = \delta(H_i)$ for $1 \leq i \leq n$;
initialize $i \leftarrow n, H \leftarrow G$

while $i \geq 1$ **do**
    Let v_i be a vertex of minimum degree in H;
    $H \leftarrow H - v_i$, $i \leftarrow i - 1$;
end
Report sequence $v_1, v_2, \ldots, v_n$;

Algorithm 1: SLV algorithm

that bursts which overlap the largest number of other bursts will be assigned first to wavelengths, which will lead to minimizing the number of bursts to be dropped. The algorithm does not consider the weight of the bursts, and it only attempts to minimize the *number* of dropped bursts.

5.2 Maximal Cliques First (MCF)

The basic idea behind the MCF algorithm is that since the maximum clique that can be colored is of size k, then our problem is equivalent to that of deleting a subset of intervals such that all remaining cliques are of size k or less. To this end, the MCF algorithm finds all the maximal cliques in the interval graph, then orders them in an increasing order according to time. Let $\{C_1, C_2 ... C_m\}$ be the set of maximal cliques in G ordered such that $C_i \prec C_j$ for $i < j$. The algorithm processes intervals belonging to C_i before intervals belonging to C_j for $i > j$. A formal description of the algorithm is given in Algorithm 2.

The MCF algorithm works as follows. It finds the list of maximal cliques in the interval graph with size larger than the number of available wavelengths. The algorithm then finds the clique with the latest occurrence time among this list. Thereafter, the algorithm removes the intervals with the smallest finish time from the maximal clique (and from the graph) such that the size of the maximal clique becomes equal to the number of the available wavelengths. This process is repeated until the size of all maximal cliques in the graph are less than or equal to the number of wavelengths. All vertices remaining in G are appended to the output list first, then the vertices removed by the algorithm are appended to the end of the output list.

6 Performance Evaluation

This section presents experimental results on the proposed algorithms: SLV and MCF, and their comparisons with the optimal batch algorithm (BATCH-OPT), and the greedy LAUC-VF algorithm. Our main concern in the following simulations is the blocking probability.

input : G on n vertices, k
output : A list of vertices L
initialize $H \leftarrow G$
while *true* **do**
  Let C = list of all maximal cliques in H with size $> k$;
  if C *is empty* **then**
    break;
  end
  Let c_{max} = clique with latest occurrence time in C;
  $z = size(c_{max}) - k$;
  for $i \leftarrow 0$ ***to*** $z - 1$ **do**
    v_i = vertex with smallest finish time in c_{max};
    $H \leftarrow H - v_i$;
    $c_{max} \leftarrow c_{max} - v_i$;
    $S \leftarrow S \cup v_i$;
  end
end
$L \leftarrow L \cup v_j, \forall v_j$ remaining in G;
Append S to the end of L;
Report sequence L;

Algorithm 2: MCF algorithm

6.1 Simulation Setup

Figure 5 shows the simulation model used in this section, which is based on the OPNET [10] simulation tool. It consists of a single OBS node connected to traffic sources and a sink node. Each input link carries two separate wavelengths, one for data and one for control. The output link carries five wavelengths, four for data and one for control, and each wavelength has transmission speed of approximately 2.5 Gbps (OC-48). We assume that sources generate control and data bursts. Sources generate bursts according to Poisson process, and burst size has a mean value equal to B bits. The offset time has a uniform distribution over $[A_{min}, A_{max}]$. Let Δ be the transmission time of 1024 bits on one of the wavelengths, i.e. $\Delta = 1024/2377728000 = 4.3\text{e-}7$ seconds. We express the acceptance delay and the offset time in terms of Δ. In the following simulations, unless otherwise stated, we set the acceptance delay d to an arbitrary value of 100Δ second which is approximately 40μsec. The greedy wavelength algorithm used in conjunction with the batch algorithms is the LAUC-VF algorithm.

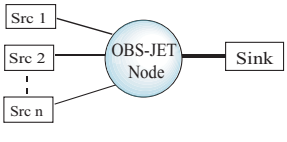

Fig. 5. Simulation model

6.2 Blocking Probability Versus Offered Load

In this section we study the effect of the proposed algorithms while varying the OBS node load. We study two scenarios. In the first scenario we use exponentially distributed burst sizes with mean value $B = 81920$ bits. The values of A_{max} and A_{min} are set to 150Δ second and 130Δ sec. respectively.

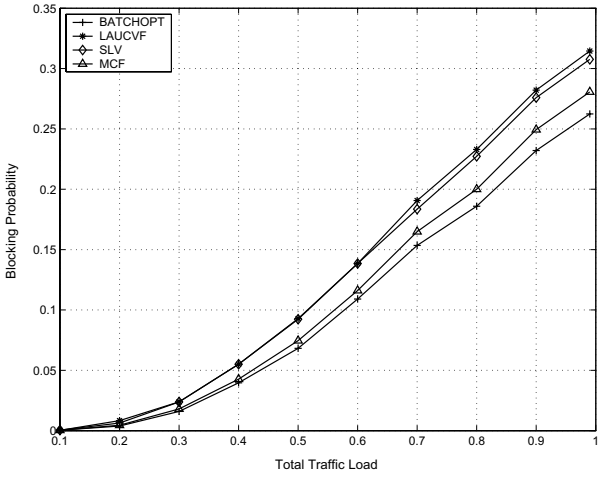

Fig. 6. Blocking Probability vs. Offered Load with exponential burst size

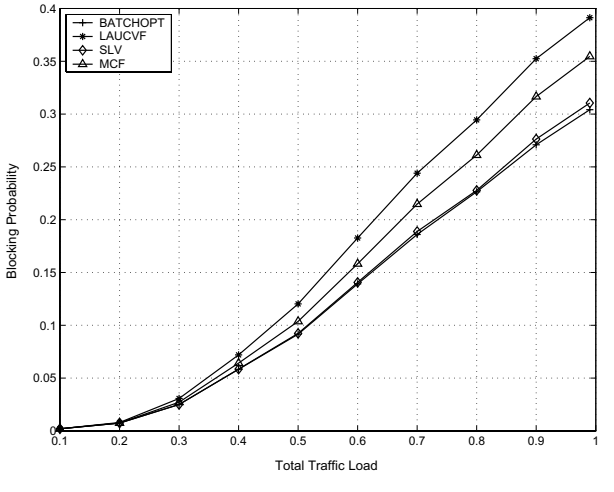

Fig. 7. Blocking Probability vs. Offered Load with constant burst size

Figure 6 shows that the performance of batch algorithms is upper bounded by the optimum batch algorithm as expected, and lower bounded by the greedy LAUC-VF algorithm. The figure shows that the MCF algorithm performs better than the SLV algorithm in this scenario, and its performance is close to the BATCH-OPT algorithm. In addition, the figure shows that the SLV algorithm performs slightly better than the LAUC-VF algorithm, however its results are not as significant when compared to the MCF algorithm.

In the second scenario we use constant burst size equal to 81920 bits. Figure 7 shows that the SLV algorithm is the best performing algorithms in this scenario. Moreover, the MCF algorithm in this scenario performs significantly better than the LAUC-VF algorithm, but not as good as the SLV algorithm.

6.3 Blocking Probability Versus Offset Time Range

We define the offset time range to be $A_{max} - A_{min}$. In this section, we vary the value of A_{min} to study effect of the offset time range on the performance of the batch scheduling algorithms. We use exponentially distributed burst sizes with mean $B = 81920$ bits. The value of A_{max} is set to 200Δ, and offered load is set to 99% of the link capacity. The value of A_{min} is varied between 10Δ and 150Δ. Figure 8 plots the blocking probability against the offset time range value.

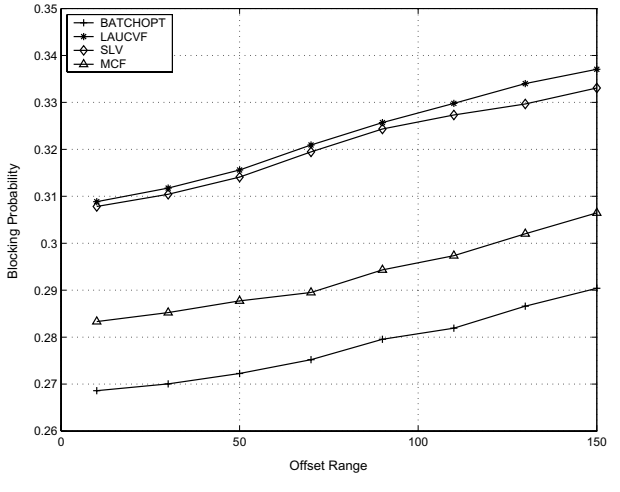

Fig. 8. Blocking Probability vs. Offset Time Range

Figure 8 illustrates that generally the blocking probability increases as the offset time range increases. Additionally, the rate of the increase in the blocking probability is approximately equal for all algorithms. This increase is due to the "retro-blocking" phenomena [11] of the JET signaling protocol, in which, a reservation request can be blocked by another reservation starting after its own

time. Obviously this phenomena becomes more significant as the offset time range becomes larger.

7 Concluding Remarks

In this paper we have introduced a novel class of wavelength scheduling algorithms in OBS networks called batch scheduling algorithms. We have described an optimum batch scheduler that serves as an upper bound for the performance of the batch scheduling algorithms. Moreover, we have introduced two heuristic batch scheduling algorithms: SLV, MCF. The MCF algorithm was shown to be superior in scheduling variable size bursts. In case of constant size bursts, the SLV algorithm was shown to perform better than the MCF algorithm. Both algorithms reduce the blocking probability considerably with respect to greedy scheduling algorithms.

References

1. Qiao, C., Yoo, M.: Optical burst switching (obs) - a new paradigm for an optical internet. Journal of High Speed Networks **8** (1999) 69–84
2. Wei, J.Y., McFarland, R.I.: Just-in-time signaling for wdm optical burst switching networks. Journal of Lightwave Technology **18** (2000) 2019–2037
3. Kaheel, A., Alnuweiri, H.: Wavelength scheduling algorithms in buffer-less optical burst switching networks. In: Proceedings of IASTED International Multi-Conference on Wireless and Optical Communications-WOC'04, Canada (2004)
4. Xiong, Y., Vandenhoute, M., Cankaya, C.: Control architecture in optical burst-switched wdm networks. IEEE Journal on Selected Areas in Communications **18** (2000) 1838–1851
5. Matula, D.W.: Smallest-last ordering and clustering and graph coloring algorithms. Journal of the ACM **30** (1983) 217–427
6. Yannakakis, M.: The maximum k-colorable subgraph problem for chordal graphs. Information Processing Letters **24** (1987) 133–137
7. Arkin, M., Silverberg, E.: Scheduling jobs with fixed start and end times. Discrete Applied Mathematics **18** (1987) 1–8
8. Golumbic, M.C. In: Algorithmic Graph Theory and Perfect Graphs. Academic Press (1980)
9. Papadimitriou, C.H., Steiglitz, K. In: Combinatorial Optimization : Algorithms and Complexity. Dover Publications (1998)
10. OPNET Technologies Inc.: OPNET Modeler. Web page: `http://www.opnet.com/` (2005)
11. Kaheel, A., Alnuweiri, H., Gebali, F.: Analytical evaluation of blocking probability in optical burst switching networks. In: Proceedings of IEEE International Conference on Communications-ICC'04, France (2004)