

Practical Priority Contention Resolution for Slotted Optical Burst Switching Networks

Farid Farahmand, Vinod. M. Vokkarane, and Jason P. Jue
Department of Electrical Engineering and Computer Science
The University of Texas at Dallas, Richardson, TX 75083-0688
{ffarid, vinod, jjue}@utdallas.edu

ABSTRACT

Optical burst switching (OBS) has been proposed as a competitive hybrid switching technology to support the next-generation optical Internet. This paper addresses performance issues in the slotted and unslotted OBS networks and extends a proposed contention resolution algorithm called the Shortest Drop Policy to support service differentiation. In addition, this paper presents a practical scheduler design, which can be suitable for the core switch node in OBS networks. Using hardware simulations the performance of our algorithm in terms of processing speed, scalability, and cost is evaluated.

Keywords: Optical burst switching, slotted and unslotted transmissions, contention resolution, hardware implementation.

1. INTRODUCTION

The amount of raw bandwidth available on fiber optic links has increased dramatically with advances in dense wavelength division multiplexing (DWDM) technology; however, existing optical network architectures are unable to fully utilize this bandwidth to support highly dynamic and bursty traffic. As the amount of bursty Internet traffic continues to grow, it will become increasingly critical to develop new architectures to provide the flexible and dynamic bandwidth allocation to support this traffic. Optical burst switching (OBS) has been proposed as an efficient way to satisfy the future bandwidth requirements of such networks.

In OBS, incoming IP packets are assembled into super-sized packets called data bursts. The edge nodes transmit data bursts following a burst header packet (BHP) after some offset time¹. Each BHP contains routing, scheduling, and packet priority level information and is processed electronically prior to its corresponding data burst arrival. Consequently, when the data burst arrives at the intermediate core switch nodes, the burst can “cut-through” the switch on the pre-assigned path with minimum processing. Thus, in OBS the data plane retains the high capacity of all-optical switching, and the control plane takes advantage of the flexibility of electronics.

Broadly speaking, similar to optical packet-switched networks, OBS networks can be divided into slotted and unslotted categories². In slotted OBS networks, control and data channels are divided into fixed-size time slots. Each control slot is further divided into several BHP slots with fixed duration. The data burst can be as long as a single or a multiple number of data slot. Therefore, in a slotted transmission mechanism, the offset time as well as the duration of the data burst and its BHP, will be expressed in terms of slots. Furthermore, the OBS core node must align all incoming optical data bursts to the slot boundaries prior to allowing them to enter the switch fabric³.

In an unslotted OBS network, data bursts and their BHPs can be transmitted at any time and do not have to be delayed until the next slot time boundary. However, in such networks, the start and end time of the BHP's associated data burst must still be specified using some arbitrary units of time. These units of time have a much finer granularity compared to data time slots and are typically on the order of clock cycle.

In terms of performance, slotted and unslotted OBS networks behave differently. For example, better bandwidth efficiency can be obtained in unslotted OBS networks, however, their burst loss probability will be higher due to unpredictable burst arrival characteristics.

A major concern in both slotted and unslotted OBS networks is contention on outgoing data channels, which can result in burst loss. The scheduler block is the primary component of the electronic control plane in the core switch that manages data burst contentions and their reservation. Developing effective scheduling algorithms and contention resolution strategies is critical in order to efficiently use the bandwidth and reduce data burst loss. Various scheduling algorithms have been proposed to effectively use the available bandwidth. For example, in the Latest Available Unscheduled Channel with Void Filling (LAUC-VF) algorithm⁴ a burst chooses the unused channel that becomes available at the latest time and gaps between scheduled data bursts can be utilized. Contention resolution strategies can also protect high priority bursts while reducing the overall burst loss rate. A survey of different contention resolution algorithms in OBS, including the Shortest Drop Policy (SDP), has been provided in⁵. In the SDP, the lowest priority data burst with the shortest duration will be dropped in the case of a contention.

Furthermore, the scheduler block in the core switch must be practical for high-speed hardware implementation in terms of processing time, scalability, and cost. The scheduler's processing time is directly proportional to the complexity of the scheduling algorithm and directly impacts the end-to-end packet delay. Shorter burst header processing time can also affect the optical buffering requirements. The scheduler must be scalable in order to sustain system growth as new ingress/egress ports and channels are added. The primary cost of the scheduler's implementation is its memory requirements, which highly depends on the complexity of reservation algorithms. Many efforts have been made to propose practical design solutions to OBS signaling protocols⁶ and channel scheduling algorithms⁷.

The design solution provided in this paper is different with previous works in terms of both scheduling technique and implementation objectives. The contribution of this paper is to present a practical design of the scheduler block suitable for the core node. We focus on three objectives: first, the design must be generic such that it can operate with any burst reservation algorithms; second, the design must be such that it can fully be implemented in hardware and involves no software; finally, the design must be realizable on an available off-the-shelf reconfigurable hardware device, i.e. field programmable gate array (FPGA) operating at hundreds of MHz.

The development of the scheduler block is carried out in three steps. (1) We first study alternative scheduler architecture. This study involves the method in which BHPs are received and processed, and how egress ports are controlled. We propose two specific architectures, namely *centralized* (pipelined) and *distributed* (parallel). In centralized scheduler architecture, a single scheduler controls all egress ports on the switch fabric. On the other hand, in the distributed scheduler architecture, each egress port on the switch fabric has its own independent scheduling block. (2) We utilize an effective reservation algorithm including the data burst scheduling and contention resolution strategy. In this step we consider a well-defined scheduling technique and focus on extending our previously introduced contention resolution strategy⁵, SDP, to slotted and unslotted OBS networks and to support quality-of-service (QoS). (3) We model the scheduler design using hardware description language, and test its functionalities using hardware simulation.

Each of the above steps is described in further detail in the subsequent sections. Section II describes the basic concepts, system architecture, and the core node complexity of the slotted and unslotted OBS networks. In Section III we focus on the architecture of the scheduler block embedded in the core switch node. We consider alternative scheduling architecture approaches, namely centralized and distributed, which can be used to realize the contention resolution and scheduling algorithms. We also compare various contention resolutions and scheduling algorithms that can be implemented in the scheduler block. In Section IV we describe the actual hardware prototype design of our proposed scheduler. This includes the formal description of the scheduler's internal blocks and hardware implementation of the packet scheduling algorithms.

2. OBS SYSTEM ARCHITECTURE

Figure 1 shows an OBS network with DWDM links. The OBS network consists of a set of edge and core nodes. The traffic from multiple client networks is accumulated at the ingress edge nodes and transmitted through high capacity DWDM links over the core. Edge nodes provide legacy interfaces such as Gigabit Ethernet, ATM, and IP, and are

responsible for data burst assembly and disassembly. Data burst assembly refers to the process by which incoming packets are aggregated into data bursts. The reverse operation is referred to as the disassembly process.

2.1. Slotted and Unslotted OBS Networks

OBS networks can be divided into two broad categories: slotted and unslotted. In synchronous slotted OBS networks, as shown in Figure 2(a), data bursts and BHPs are only transmitted on their slot boundaries. In this transmission scheme, control and data channels are divided into time slots with fixed duration. Each control slot is further divided into several BHP slots with fixed duration. When a data burst is ready for transmission, its header packet is first transmitted on an available BHP slot on the control channel. After an offset time, at the start of a new data slot, the associated data burst is transmitted. It is therefore convenient to represent offset time and burst duration in terms of slots. Note that data bursts in slotted transmission can have variable or fixed durations.

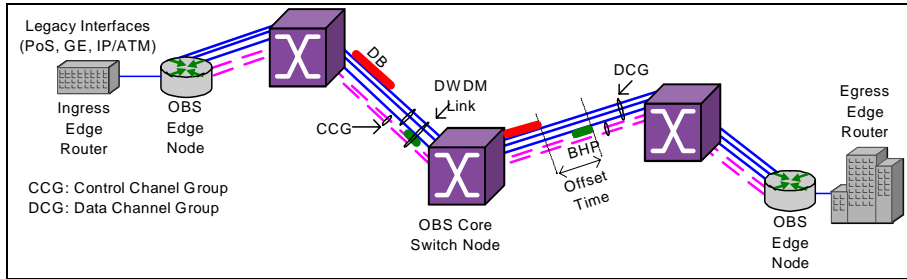


Figure 1. Optical burst-switched network.

Figure 2(b) shows an asynchronous unslotted transmission of data bursts and BHPs. In such a network there is no need to delay a data burst and its BHP until the appropriate slot boundaries have arrived. Although data bursts and their BHPs can be transmitted at any time, the start and end of the BHP's associated data burst must still be identified in terms of some predefined time units. Even though, in theory, data bursts in unslotted OBS networks can have fixed or variable lengths, for practical reasons, we only consider the latter case.

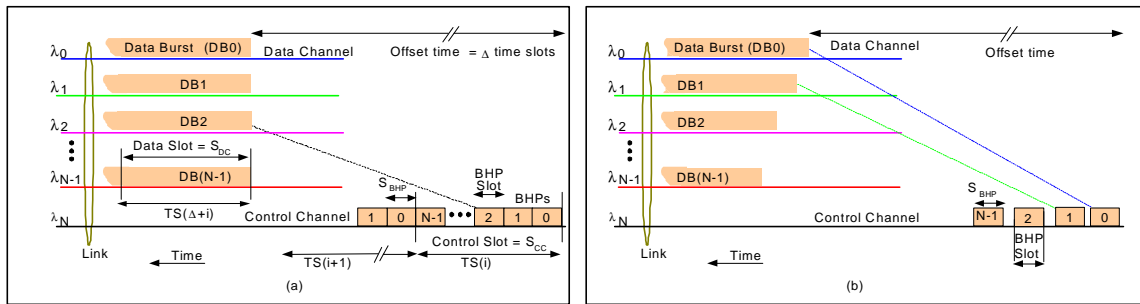


Figure 2. Data bursts and their BHPs in (a) the synchronous slotted and (b) the asynchronous unslotted transmission networks.

In general, since data bursts in slotted networks are transmitted in discrete sizes, bandwidth efficiency (the ratio between the number of data bytes over the total number of data and overhead bytes)⁸ reduces at low traffic loads because data slots are not fully utilized. On the other hand, because arbitrary time units are used to represent data burst duration and offset time, in unslotted OBS networks the size of the header packet may be longer. In addition, the lack of slot boundaries in unslotted networks, eliminates synchronization time which may lead to lower average end-to-end packet delay.

In terms of data burst loss rate, simulation results for photonic packet switching indicate that the performance of slotted transmission with fixed-size packet length is superior to an unslotted system with variable size packets⁹. This is assuming that the average packet duration is equivalent to the fixed-size packet. Generally speaking, having smaller size data bursts can lower the loss rate. Clearly, the tradeoff to this is lowering the bandwidth efficiency.

The implementation complexity of slotted and unslotted packet switching has been described in³. However, in OBS systems, the unslotted transmission mechanism involves some additional complexities. This is mainly due to fact that the core switch nodes must resynchronize and align the data bursts and their associated BHPs.

A quick comparison between slotted OBS networks with fixed and variable size data bursts shows that the former results in lower bandwidth efficiency whereas the latter is slightly more complex to construct and requires more information fields in the header packet. The average end-to-end IP packet delay and the loss rate in variable and fixed sized slotted OBS depends on a number of factors, including the input traffic characteristics, load, and the edge node criteria for determining when bursts can be released. For example, at the edge node, the data bursts can be transmitted when a time threshold has been reached or when a specific length requirement has been met^{10,11}.

2.2. Core Node Architecture

A general description of edge nodes in OBS networks has been provided in⁴. In this subsection we briefly describe the overall core switch node architecture. Figure 3 shows the generic core switch node architecture in OBS. In this hybrid architecture the core switch is fully transparent to optical data bursts, while the control channels are converted into electrical signals. Each ingress link is initially demultiplexed and all data and control channels are separated. Individual optical channels are examined for optical characteristics, including the optical power level and signal-to-noise ratio.

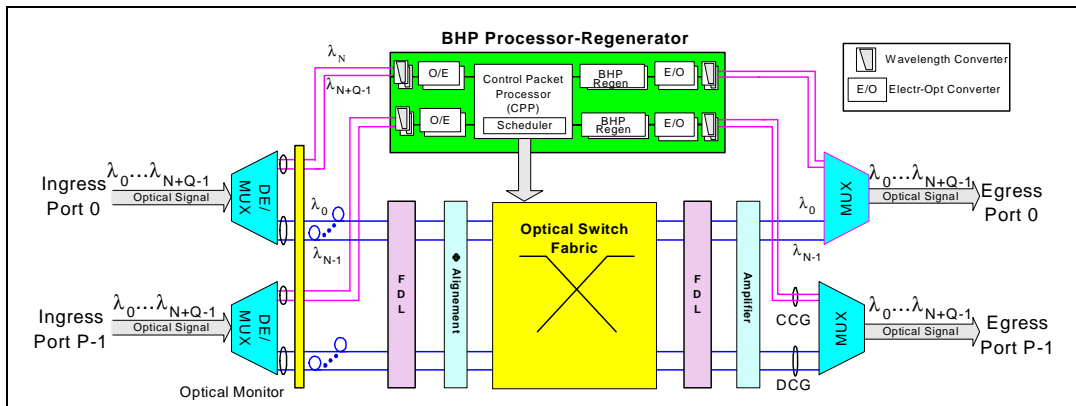


Figure 3. Typical architecture of the OBS core switch node with optical data burst alignment capacity.

Fiber delay line (FDL) blocks can be used for variety of reasons. One potential application is to use FDLs as input or output optical buffers to delay data bursts when multiple bursts are contending for the same egress port in the *switch fabric*. Another possible application of FDLs is to compensate for BHP processing time delay in which data bursts are deliberately delayed in order to maintain the necessary offset times.

In slotted transmission, before data bursts go through the switch fabric, they must be aligned to time slots¹². This operation can be performed using incremental fiber delay blocks, tunable converters, or more exotic approaches such as optical phase locked loops.

Many researchers have proposed various switch fabric architectures^{13,14}. An important issue in the switch fabric design is its cost and scalability¹⁵. In order to improve performance, many switch fabric architectures include wavelength converters. The optical couplers and wavelength converters in the switch fabric, along with the FDL blocks all cause optical loss of energy on the outgoing signals. Therefore, use of optical amplifiers prior to data burst transmission may be required.

Incoming BHPs on control channels are processed and regenerated in the *BHP processor-regenerator* block (BPRB). In the BPRB the BHPs are first converted into electrical signals and then sent to a *control packet processor* (CPP), where

they are processed and scheduled if proper resources are available. If a BHP request was successfully reserved, the switch fabric setup needs to be updated as the corresponding data burst arrives and leaves the switch. Furthermore, each accepted BHP must be regenerated with the updated information and transmitted to the downstream core node. The control packet processor is considered as the main part of the core node's BPRB and contains the data burst reservation algorithm.

3. OBS CONTROL PLANE

In the following sections we describe details of the CPP shown in Figure 3. Our discussion involves three specific areas: alternative architectures for the CPP, performance of the CPP reservation algorithm, and hardware prototyping of the CPP. We provide the formal description of the prototype model in the next section.

3.1. Architectures

Figure 4 shows two different design architectures for the CPP block: centralized and distributed. In the centralized (pipelined) architecture, as shown in Figure 4(a), each BHP is initially received by the receiver block, and its payload, including data burst length, destination, offset, QoS, is extracted. Then, the reformatted BHP request will be stored in the priority queue in which requests with higher QoS are given service priority. The scheduler block processes individual requests from each receiver queue based on their destinations. Upon acceptance of the request, the reservation is stored in the scheduler block until its associated data burst is serviced. The *switch control* block provides an interface between the scheduler and the switch fabric, updating ingress-egress channel connections.

In the distributed (parallel) architecture of the CPP, as shown in Figure 4(b), egress ports have their own independent scheduler blocks. Each incoming BHP is decoded and checked for its destination. Then, the BHP is forwarded to one of the P destination queues connected to the BHP receiver block. Destination queues with similar index numbers are interfaced to the same scheduler block. The scheduler block processes the request, and upon making a reservation, the reservation will be stored until its associated data burst is serviced.

The centralized and distributed architectures of the CPP, to a large extent, resemble input queuing and virtual output queuing systems¹⁶, respectively. In the distributed architecture, by arranging input buffers according to egress ports, we can achieve parallel processing of BHP requests with different destinations. The distributed architecture also minimizes the problem of head-of-queue blocking. Furthermore, the distributed architecture is more reliable and provides better scalability in terms of modularity and adding new control channels. However, one important disadvantage of the distributed scheme is its high relative memory requirement and RAM usage. Assuming P represents the number of control channels entering the CPP, there will be P^2 destination queues, each of which must be dimensioned for the worst-case traffic condition. This is P times more than the total number of priority queues used in the centralized scheme.

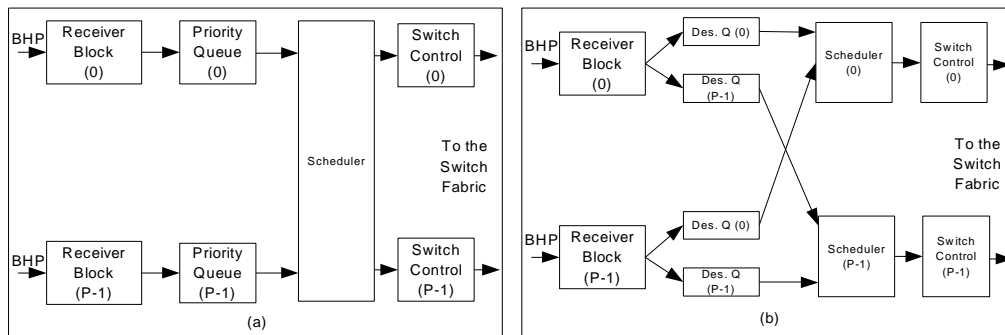


Figure 4. Control packet processor (CPP) architectures: (a) centralized (pipelined) and (b) distributed (parallel).

3.2. Reservation Algorithms

The scheduler is the heart of the control packet processor block. Incoming requests to the scheduler will be processed according to a reservation algorithm. Therefore, the efficiency of the reservation algorithm is very critical. A reservation

algorithm addresses two main issues: (1) how to use available resources to schedule incoming data bursts; and (2) how to resolve contention when the total number of data bursts going to the same output port at a given time is larger than the available channels on that port. The first issue is referred as scheduling and the second issue is known as contention resolution. Various scheduling techniques for OBS networks have been proposed in earlier literature^{4,17}.

When no available unscheduled channel can be found for a BHP request, contention resolution schemes may be invoked. Different types of contention resolution policies have been proposed including, time deflection (using buffering)¹⁸, space deflection (using deflection routing)¹⁹, wavelength conversion (using wavelength converters)²⁰, or a combination of these approaches. When a contention cannot be resolved by any one of these techniques, one or more bursts must be dropped. The policy for selecting which bursts to drop is referred to as the soft contention resolution policy. In the remainder of this section we briefly review a few such algorithms.

3.2.1 Latest Drop Policy (LDP)

The simplest soft contention resolution policy is the latest-arrival drop policy (LDP). In LDP, the algorithm searches for an available unscheduled channel (as in LAUC-VF), and if no such channel is found, the latest incoming data burst will be discarded. Although processing speed of BHPs in LDP is attractive, the main disadvantage of this technique is that it has relatively poor performance with respect to data loss when no optical buffers are utilized.

Inherently, LDP is not capable of differentiating packets with different priority types. A novel scheme proposed in²¹ suggests that giving extra offset time to high priority data bursts can ensure their early reservations. This approach is known as offset-time-based QoS.

3.2.2 Segmentation Drop Policy (SEG)

The basic assumption in this scheme is that each transmitted data burst consists of individual independent segments such as slots. Therefore, if contention occurs, only the segments of the lower priority burst involved in the contention will be removed. Details of this mechanism, known as Segmentation, along with its variations are described in²².

Although the QoS-enabled Segmentation algorithm appears to be straightforward, the hardware implementation in terms of burst assembly and disassembly, as well as overhead insertion and extraction, can be involved. Furthermore, implementing SEG requires a complex data burst alignment stage. This is due to the fact that, as data bursts go through more intermediate core nodes, they may be subject to length change.

3.2.3 Shortest Drop Policy (SDP)

In the SDP upon detecting contention, the lower priority burst with the shortest duration and latest arrival time will preferentially be dropped. One drawback of the SDP is its potential over-reserving of resources. Soon after a data burst is reserved, its BHP is retransmitted. However, the reservation can potentially be cancelled within the next Δ time units before the associated data burst, arrives. The cancellation can be due to arrival of a data burst with higher priority or longer duration. One option to minimize over-reservation is to send a burst cancellation control packet to the downstream nodes. This can occur immediately after an old reservation is replaced by a new reservation. Note that the replaced reservations are permanently removed in the SDP.

In terms of supporting class differentiation, the SDP can support an unlimited number of priority levels and requires no extra offset assignments for bursts with higher service requirements. The SDP mechanism can also offer absolute as well as proportional class isolation. In absolute isolation the possibility of a high-priority burst being blocked by any lower priority burst is eliminated. On the other hand, in proportional isolation the dropping criteria will be based on the relative length and priority level of data bursts. In such a scheme, it is possible that, in a contention between a short duration high-priority burst and a long duration low-priority burst, the burst with higher priority is discarded.

3.2.4 Simulation Results

Through simulation we compare the performance of different contention resolution policies. The simulation results are obtained with the following assumptions: (1) The network consists of a single bufferless core switch with 4 ingress/egress ports. Each port consists of a single control channel and 7 data channels, each equipped with a full wavelength converter. (2) In the case of synchronous slotted transmission with fixed-size bursts, the burst duration was set to be 20,000 bytes. In the asynchronous unslotted transmission, the maximum data burst duration was assumed to be

40,000 bytes. This is equivalent to 20 data slots. (3) Data bursts can have three distinct priority levels, $c=1, 2, 3$, with a distribution ratio of 10%, 30%, and 60%, respectively.

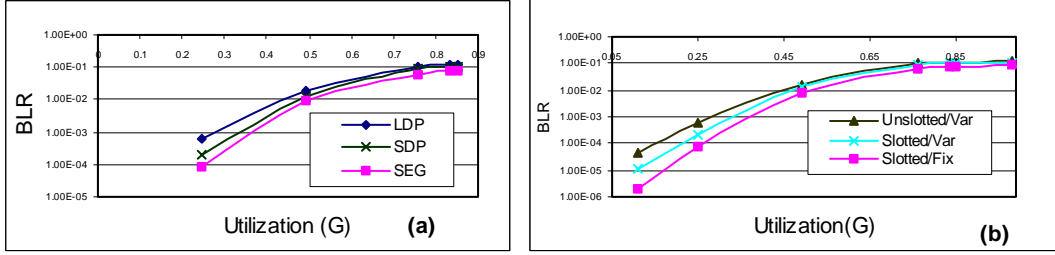


Figure 5. (a) Overall BLR performance using different contention resolution schemes assuming slotted transmission with variable length. (b) Comparing slotted and unslotted transmission using the SDP.

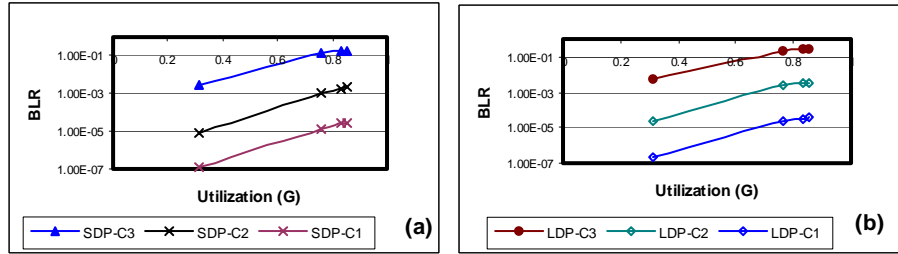


Figure 6. BLR performance for all three classes, assuming slotted transmission mechanism with variable length bursts: (a) SDP and (b) LDP. C1 indicates the highest priority level.

We represent the simulation results in terms of utilization and burst loss rate. Utilization, G , is the ratio of the total number of burst slots generated in the network per unit time. The burst loss rate, BLR, is the percentage of burst payload that was sent by the source but never received by the destination.

Figure 5(a) shows the overall performance of SDP compared to the Segmentation (SEG) and Latest Drop Policy (LDP) algorithms under slotted transmission mechanism with variable length bursts. Note that, SEG and LDP provide the upper and lower bounds on performance, respectively. On average the SDP performs about 20% better than LDP.

The performance comparison between the SDP algorithm in slotted and unslotted networks is shown in Figure 5(b). Note that the slotted transmission with fixed-size data bursts performs about 15% better than the unslotted transmission. The resulting burst loss rate for all three classes of service using SDP is shown in Figure 6(a). These results can be compared with Figure 6(b), which depicts the BLR for individual classes using LDP. Note that the BLR for the highest-class priority is significantly improved in SDP compared to LDP.

4. HARDWARE IMPLEMENTATION

In this section we provide a formal description of the control packet processor prototype using the proposed distributed architecture, as shown in Figure 4(b). We also describe the hardware implementation of the SDP contention resolution algorithm, focusing on loss performance, cost, and scalability. In our prototype we assume a bufferless OBS network, where the control and data slots have the same duration. In addition, we assume each core node has P ingress/egress ports each having N data channels and a single control channel. For simplicity, we limit control packets to BHPs and cancellation packets.

As shown in Figure 3, prior to arriving to the CPP, each BHP is demultiplexed and converted from optical to digital signals. BHPs are then processed through the BHP receiver block, as shown in Figure 4(b). The receiver block first checks each BHP for proper framing and parity information to ensure the packet validity. Then, upon detecting a valid BHP frame, the required information fields (such as destination, burst length, offset, QoS, ingress channel) are extracted for further processing.

The receiver block also generates a timestamp count, CS_CNT , representing the control slot in which a BHP arrived. The timestamp generator can be a periodic counter with a maximum value greater than the sum of the maximum allowable data burst length and offset time. Therefore, each request is time stamped based on its corresponding data burst arrival time and the end of the data burst service time. The beginning of the burst arrival time (TS) is equivalent to the latest timestamp count value added to the offset time (Δ). The end of the scheduling time for data burst i , TE_i , is represented by

$$TE_i = TS_i + L(B_i) = (CS_CNT + \Delta_i) + L(B_i),$$

where $L(B_i)$ is the length of the data burst i . Therefore, each request, BHP_i , requires a reservation from TS_i to TE_i and $L(B_i) = TE_i - TS_i$. The receiver block reformats each incoming BHP to include only the required payload and the scheduling timing information. Note that, in slotted transmission, TS and TE can be expressed in terms of data burst slots.

Depending on the BHP's destination, the receiver sends the reformatted BHP request to one of its P destination queues shown in Figure 4(b). The destination queue is a prioritized digital queue that services requests according to the start of their scheduling time and their priority. If the offset times between requests (with the same priority) vary, the request with the earliest data burst arrival time must be serviced first. On the other hand, the destination queue must also ensure service priority to requests with higher QoS. This is critical because of possible overflow that may occur in the destination queues due to having persistent traffic going to the same destination port. In addition, service starvation for low-priority requests²³ must be avoided in the destination queue.

All destination queues with the same index (0 through $P-1$) are interfaced to the same scheduler block, each of which schedules requests for one of the egress ports on the switch fabric. Thus, requests with different destinations can be scheduled concurrently. Details of a generic scheduler block and its interfaces are shown in Figure 7. The scheduler block contains the scheduling algorithm and is divided into four hardware blocks: *arbiter*, *processor*, *channel manager*, and *statistics accumulator*.

The scheduler's arbiter is required to control request flows into the scheduler. The arbiter ensures that requests with earlier scheduling time and higher priority levels are serviced first. When dealing with service differentiation, two critical issues must be addressed: possible service starvation for low-priority requests and fairness between requests coming from different control channels with the same destination. Various arbitration schemes can be considered to ensure service differentiation. In our design, we implement a priority Binary Tree Arbiter (BTA) scheme to control the packet flow from destination queues into each scheduler. The BTA²⁴ is a Mealy machine in which decisions are made based on the previous states. Studies have shown that such arbitration schemes can provide fair allocation of resources among all requests while avoiding service starvation. Other advantages of BTAs include their fast processing time and scalability.

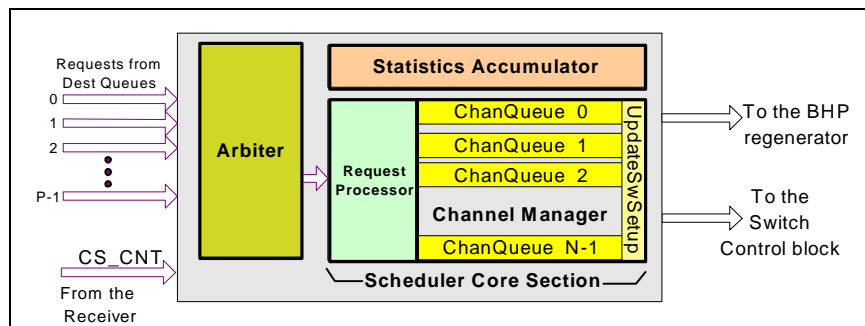


Figure 7. Details of the scheduler block and its interfaces, assuming P ingress/egress ports each having N data channels and a single control channel.

The arbiter passes the requests to the processor one at a time. Based on the request's scheduling time and its duration, the processor searches through all previous reservations on different channels and checks for any available bandwidth to accommodate the new request. If a request was successful, it is passed to the channel manager block. Obviously, an increase in the number of existing reservations for each channel results in longer search time per new request. If there are no available resources, the reservation request is denied and its incoming associated data burst is discarded. Note that different scheduling and contention resolution policies, such as the offset-time-based QoS or SDP can be implemented in the processor block.

The channel manager block contains as many as N channel queues (one for each channel on the egress port) and an update switch setup block. The processor sends an accepted request to the proper channel queue and the request is stored until its corresponding data burst is completely serviced. The storage time for a given request j in a channel queue is equivalent to $(\Delta_j + L(B_j))$ data burst slots. For simplicity, and without loss of generality, in our design, we assume that all incoming requests have constant offset. In this case all requests will be sequentially stored in channel queues in order of their burst arrivals. Furthermore, such a constraint simplifies the search, because upon the arrival of new requests, only the head of each channel queue needs to be examined. Therefore, a channel queue can be constructed using a dual-port FIFO with the appropriate depth to store the requests.

When the CS_CNT changes, a new search for reservations with starting or ending timestamps similar to the current timestamp count begins through all N channel queues. If the starting time of a request (TS) matches the CS_CNT, then the switch control block updates the switch fabric with the new ingress-egress path. On the other hand, if the ending timestamp of an active request (TE) was found to be the same as the CS_CNT, then the switch control block must update the switch fabric and remove the active ingress/egress path. In this case, the data burst has been completely serviced through the switch fabric, and the update switch setup must remove the corresponding request from the channel queue in order to avoid its possible reactivation.

Soon after a reservation request is reserved, the update switch setup block sends a copy of the reserved request to the BHP Regenerator block, as shown in Figure 3, for retransmission to the next core node. The regenerated BHP must reflect the updated information, including the ingress port, and channel. The update switch setup block is also in charge of initiating other types of control packets such as cancellation packets. A cancellation packet may be required in the SDP to notify the downstream core nodes that a previous reservation has been cancelled. The scheduler block also includes of a statistics accumulator block. This block can be used to keep track of the percentage of requests dropped and number of errors detected.

4.1. Illustration of the Scheduler Operation

We now show the scheduler block operation by means of an example. Let us assume that there are three channels on each egress port, packet transmission is slotted, and all incoming BHPs have the same priority and offset time. Figure 8(a) maps out the expected arrival time slots for the incoming data bursts between CS_CNT = 12 and 19. The TS and TE represent the expected arriving and service completion time slots of each incoming data burst, respectively. Note that the duration of each data burst can be expressed in terms of the difference between its TE and TS.

The contents of channel queues (CQ0 through CQ2), as the timestamp count increments, are shown in Figure 8(b). Each channel queue stores the requests for one of the three available channels. Initially, we assume the request for B1 arrives at CS_CNT = i and there are no other previous reservations. The request for data burst B1 is stored in CQ0, indicating that B1 will be passed through channel 0 of the egress port. One time slot later, requests for B2 and B3 are reserved on CQ1, and CQ2, respectively. This will be based on first-available-channel criteria. When the timestamp count reaches $(i+6)$, a request for B4 arrives and is reserved on CQ1. Upon arrival of the request for B5, the reservation for B4 is removed and B5 is reserved on CQ1. This is because B5 has a longer duration than B4. Note that in this example we assume that no data burst is completely serviced prior to arrival of B5's request. Therefore, as shown in Figure 8(b), requests for B1, B2, and B3 will remain at the head of queue (HoQ) until their data bursts are completely serviced. For example, at CS_CNT = 13 the reservation for B2 becomes active and is removed from CQ1 at CS_CNT = 17. Consequently, the request for B5 will be moved to the head of CQ1.

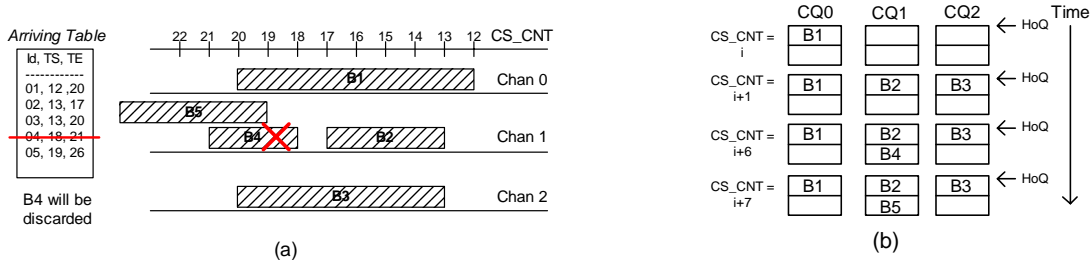


Figure 8. An example of (a) expected data burst arrivals with the same destination and (b) status of channel queues.

4.2. Scheduler Prototyping

We develop a prototype VHDL model for the control packet processor block shown in Figure 4(b). The model is implemented using a reconfigurable hardware device, i.e., field programmable gate array (FPGA), which allows easy upgradeability. In the design, software handling of the scheduling process is avoided and a hardware approach is taken. For practical reasons, we assume a single control channel per link and four ingress and egress ports, each having 16 channels. With a few exceptions, all blocks are modeled directly using VHDL. The scheduler core section, shown in Figure 7, including the processor (which includes the Shortest Drop Policy algorithm) as well as the destination queues, was initially modeled using the C-language. The C source code was then transformed into Handel-C language. Using the Celoxica DK design suite, the Handel-C source code was compiled and translated into a gate-level VHDL format optimized for a given targeted technology, i.e., an Altera FPGA device. All generated VHDL blocks were combined together using a top level VHDL code. The entire design functionality was tested and verified using the Cadence (NcSim) framework. The design synthesis was performed by Synplify and FPGA placement and routing was done using Quartus II.

4.3. Design Performance

The hardware model for the control packet processor protocol was targeted for an Altera APEX 20K FPGA family, namely EP20K400E, offering 2.5 million system gates and operating at clock rates up to 840 MHz. We assumed the operating frequency of the scheduler to be 500MHz. Figure 9(a) shows the number of cycles required to process each incoming request. Note that, as the number of reservations stored in the channels queues becomes larger, the scheduling time for new requests increases. This is intuitive, since there will be more reservations to be verified. Once all channel queues have been saturated, the processing time reaches a steady state. Figure 9(a) also indicates that, as the number of channels on a port becomes larger, the maximum number of cycles required for a new request to be scheduled increases.

Note that the storage time for each request in the channel queue is equivalent to the sum of its offset time and duration. Therefore, the maximum memory requirement for each egress port's scheduler block will be $N \cdot (\Delta_{\max} + L_{\max})$, where L_{\max} and Δ_{\max} are the largest allowable data burst duration and offset, respectively. On the other hand, the cost of each destination queue directly corresponds to the traffic characteristics. Contrary to the case for channel queues, the memory requirement for each destination queue increases when the length of the data bursts is decreased to a single slot. This implies that each control slot will have N requests to be processed. Assuming unbalanced traffic in which all bursts go to the same destination port for S continuous data slots from all ingress channels, there will be $N \cdot P \cdot S / S_r$ requests accumulated in the destination queues, where S_r is the average number of requests serviced per control slot. If the burstiness continues beyond S slots, packet overflow will occur. Therefore, the overall theoretical upper bound cost of the scheduler with distributed architecture in terms of memory requirements will be

$$[P \cdot (N \cdot (\Delta_{\max} + L_{\max})) + P^2 \cdot (\frac{N \cdot P}{S_r \cdot L_{avg}} \cdot S)] \cdot Wd,$$

where Wd is the width of each request (in bits) stored in the channel queue or destination queue. Note that in this case we are ignoring the memory units required for the receiver block.

Figure 9(b) shows the relative hardware cost of implementing the core section of the scheduler in the CPP using the Shortest Drop Policy. Note that, as the number of ports, P , and channels increase, the cost of the scheduler drastically increases.

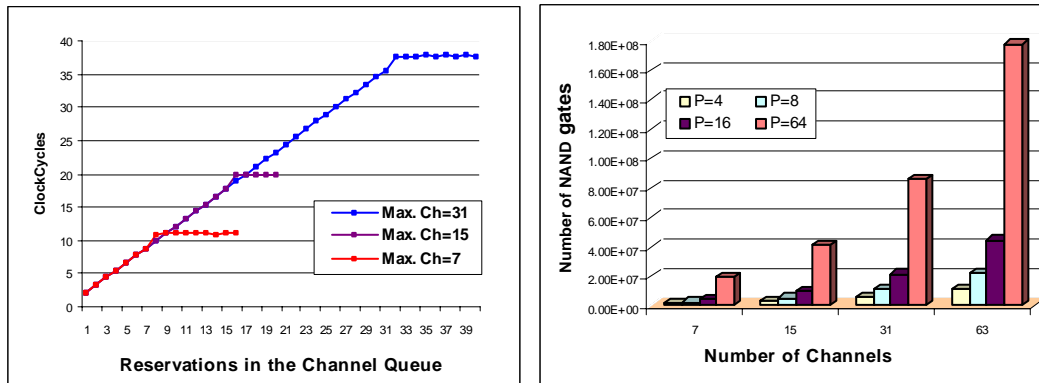


Figure 9. (a) Number of clock cycles required for each new request to be scheduled on an egress port; (b) Relative hardware cost of the scheduler core section in terms of NAND gates for various number of egress ports and embedded channels.

5. CONCLUSION

One important functional block in the OBS system is the control packet processor, which is a part of the control plane in the core switch node. The control packet processor is responsible for receiving BHPs and scheduling data bursts as well as controlling the switch fabric. This paper presented a detailed description of a control packet processor, including its alternative architectures and reservation schemes involving scheduling algorithms and contention resolution strategies. Furthermore, we presented a QoS-enabled contention resolution strategy called the Shortest Drop Policy and investigated its performance for slotted and unslotted OBS networks.

The entire header packet processor block was modeled using VHDL hardware descriptive language and implemented into a hardware programmable device such as FPGA. The performance and functionality of the design was verified using hardware simulations. The proposed design was proved to be cost efficient in terms of memory requirements, highly scalable to the system growth, and capable of operating at hundreds of MHz.

The next step in our study will be testing the control packet processor prototype under live BHP traffic and measuring statistics in terms of the number of requests dropped over a long period of time. This will provide a much deeper insight into the performance of the proposed scheduling and contention resolution algorithms. Further hardware experiments can also provide better understanding with regards to the required storage capacity for the control packet processor block as the traffic characteristic and load vary.

ACKNOWLEDGMENTS

The authors wish to thank Amir Shaikh from Celoxica for his technical support and assistance throughout this project.

REFERENCES

1. M. Yoo and C. Qiao, "Just-Enough-Time (JET): A High Speed Protocol for Bursty Traffic in Optical Networks," *IEEE/LEOS Conf. on Technologies for a Global Information Infrastructure*, pp. 26-27, Aug. 1997.
2. R. Ramaswami and K. N. Sivarajan, *Optical Networks: A Practical Perspective*, Morgan Kaufmann Publishers Inc., second edition, 1998.
3. S. Yao, B. Mukherjee, and S. Dixit, "Advances in Photonic Packet Switching: an Overview," *IEEE Communications Magazine*, vol. 38, no. 2, Feb. 2000.
4. Y. Xiong, M. Vanderhoute, and H.C. Cankaya, "Control Architecture in Optical Burst-Switched WDM Networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1838-1851, Oct. 2000.
5. F. Farahmand and J. P. Jue, "Look-ahead Window Contention Resolution in Optical Burst Switched Networks," *IEEE Workshop on High Performance Switching and Routing (HPSR) 2003*, Torino, Italy, pp. 147-151, June 2003.

6. I. Baldine, G.N. Rouskas, H.G. Perros, and D. Stevenson, "JumpStart: A Just-in-Time Signaling Architecture for WDM Burst-Switched Networks," *IEEE Communications*, vol. 40, no. 2, pp. 82-89, Feb. 2002.
7. S. Zheng, Y. Xiong, M. Vandenhout, and H. C. Cankaya, "Hardware Design of a Channel Scheduling Algorithm for Optical Burst Switching Routers," *Proceedings, SPIE ITCOM 2002*, vol. 4872, pp.199-209, 2002.
8. M. de Prycker, *Asynchronous Transfer Mode: Solution for Broadband ISDN*, Ellis Horwood Publishers, second edition, 1993.
9. S. Yao, S.J. Ben Yoo, and B. Mukherjee, "A Comparison Study between Slotted and Unslotted All-optical Packet-Switched Network with Priority-Based Routing," *Proceedings, Optical Fiber Communication Conference and Exhibit (OFC) 2001*, vol. 2, 2001.
10. V. M. Vokkarane, K. Haridoss, and J. P. Jue, "Threshold-Based Burst Assembly Policies for QoS Support in Optical Burst-Switched Networks," *Proceedings, SPIE Optical Networking and Communication Conference (OptiComm) 2002*, Boston, MA, vol. 4874, pp. 125-136, July-Aug. 2002.
11. X. Yu, Y. Chen, and C. Qiao, "A Study of Traffic Statistics of Assembled Burst Traffic in Optical Burst Switched Networks," *Proceedings, SPIE Optical Networking and Communication Conference (OptiComm) 2002*, Boston, MA, pp. 149-159, July-Aug 2002.
12. B. Bostica, M. Burzio, P. Gambini, and L. Zucchelli, "Synchronization Issues in Optical Packet Switched Networks," *Photonic Networks*, G.Prati, ed., Springer-Verlag, 1997.
13. S.L. Danielsen, et al, "WDM Packet Switch Architectures and Analysis of the Influence of Tunable Wavelength Converters on the Performance," *Journal of Lightwave Technology*, vol. 15, no. 2, pp. 219 –227, Feb. 1997.
14. V. Eramo and M. Listanti, "Packet Loss in a Bufferless Optical WDM Switch Employing Shared Tunable Wavelength Converters," *Journal of Lightwave Technology*, vol. 18, no. 12, pp. 1818 –1833, Dec. 2000.
15. D.J. Blumenthal, P.R. Prucnal, and J.R. Sauer, "Photonic Packet Switches: Architectures and Experimental Implementations," *Proceedings of the IEEE*, vol. 82, no. 11, pp. 1650 –1667, Nov. 1994.
16. C. Minkenberg, *On Packet Switch Design*, Ph.D. Dissertation, Eindhoven University of Technology, The Netherlands, 2001.
17. J.S. Turner, "Terabit Burst Switching," *Journal of High Speed Networks*, vol. 8, no. 1, pp. 3-16, Jan. 1999.
18. C. Gauger, "Dimensioning of FDL Buffers for Optical Burst Switching Nodes," *Proceedings, Optical Network Design and Modeling (ONDM) 2002*, Torino, Italy, 2002.
19. X. Wang, H. Morikawa, and T. Aoyama, "A Deflection Routing Protocol for Optical Bursts in WDM Networks," *Proceedings, Fifth Optoelectronics and Communications Conference (OECC) 2000*, Makuhari, Japan, pp. 94-95, July 2000.
20. S. Yao, B. Mukherjee, S. J. Ben Yoo, and S. Dixit, "All-optical packet switching for Metropolitan Area Networks: Opportunities and Challenges," *IEEE Communication Magazine*, vol. 39, no. 3, pp. 142-148, March 2001.
21. M. Yoo and C. Qiao, "Supporting Multiple Classes of Services in IP over WDM Networks," *Proceedings, IEEE GLOBECOM 1999*, Brazil, pp. 1023-1027, Dec. 1999.
22. V. M. Vokkarane, J. P. Jue, and S. Sitaraman, "Burst Segmentation: An Approach for Reducing Packet Loss In Optical Burst Switched," *Proceedings, IEEE International Conference on Communications (ICC) 2002*, New York, NY, vol. 5, pp. 2673-2677, April 2002.
23. S. Keshav, *An Engineering Approach to Computer Networking*, Addison-Wesley, Reading, Mass., 1997.
24. K. Boahen, "A Throughput-on-Demand 2-D Address-Event Transmitter for Neuromorphic Chips," *Proceedings, The 20th Conference on Advanced Research in VLSI*, Atlanta, GA, 1999.