A SCHEDULING ALGORITHM IN A CORE OPTICAL ROUTER WITH HETEROGENEOUS TRAFFIC

Rose Q. Hu
Department of Electrical and Computer Engineering
Mississippi State University
Mississippi State, MS 39762-9571, USA
Email: hu@ece.msstate.edu
Phone: 662-325-1529, Fax: 662-325-2298

Robert Best
Award Solutions, Inc.
Richardson, TX 75081, USA

Yi Qian
Department of Electrical and Computer Engineering
University of Puerto Rico at Mayagüez
Mayagüez, Puerto Rico 00680
Email: yqian@ece.uprm.edu
Phone: 787-833-3338, Fax: 787-833-3331

Mingzhou Jin
Department of Industrial Engineering
Mississippi State University
Mississippi State, MS 3976, USA

Abstract – Recent advances in optical networking reveal that large-scale optical networks supporting heterogeneous traffic may soon become economical as the underlying backbone in wide area networks, in which optical routers play a key role. One big challenge in designing future large-scale optical systems is packet scheduling for the core optical routers. The optical router is a delay system with packets waiting at its ingress queues. A scheduler is necessary to allocate resources so that delay and jitter sensitive real-time traffic can be served with higher priority than the non-delay sensitive traffic. The system capacity should also be efficiently utilized. This is achieved by a prioritised non-blocking scalable scheduling algorithm developed in this paper. The proposed algorithm is based on a heuristic approximation of a Linear Integer Programming model. The performance evaluations in a multi-service high capacity core optical router show that the heuristic solution is close to the optimal solution most of the time, yet it is much easier to implement.

Keywords: Scheduling, optical router, integer linear programming, QoS, heterogeneous traffic.

1. Introduction

Internet traffic has explosively grown in the past few years. It has triggered significant research in designing large-scale optical systems with very high-speed core optical switches and routers (e.g., [1–3]). One big challenge in designing a large-scale high-speed optical system is packet scheduling for core routers. A dynamic scheduling mechanism is necessary to control the switching fabric of the optical router, for the purpose of providing non-blocking transmissions and dynamic adaptations to varying traffic patterns and volumes over time. The adaptation must be fast enough to support fairness and Quality of Service (QoS) requirements as measured in terms of delay, Bit Error Rate (BER), throughput, etc. On the other hand, frequent schedule changes may cause network instability on bandwidth control. An effective and ideal scheduling design is necessary to offer a good balance among these factors.

Research work has been conducted on scheduling optical switches and routers. A scheduling algorithm is proposed in [4, 5] to provide best-effect services in the Birkhoff-von Neumann switch. The problem is formulated as a resource sharing problem that optimizes efficiency and fairness. Since it deals with the best-effort services only, the service differentiation issues are not addressed in [4, 5]. A fair scheduler presented in [6] is suitable in buffer-less circuit-switched blocking networks operating with distributed, asynchronous controllers and variable length messages. The tradeoffs and performance limitations of the fair scheduler are discussed. The circuit-switched optical networks, rather than the packet-switched or IP based networks, are studied in that paper. In [7], a hierarchical scheduling framework is introduced in a class of photonic packet switching systems based on WDM, in which the flow scheduling is separated from the transmission scheduling. As stated in [7], existing work in wide-area optical networks largely focuses on the support of end-to-end virtual connections; relatively less attention has been paid to support heterogeneous traffic types and to satisfy the potentially different QoS requirements of different traffic types. An iterative algorithm that achieves high throughput on virtual output queues (VOQs) is introduced in [8]. The algorithm provides fair access to output lines and prevents starvation of input queues. However, the algorithm is not practical because of its high computational complexity [9]. Although Weighted Bipartite Matching (Assignment) algorithms are proven to achieve 100% throughput for all admissible independently and identically distributed arrivals [9], their computational complexity is too high for high speed implementation [10]. A scheduling algorithm for a combined input and output queuing switch with space division multiplexing expansion and grouped inputs/outputs (SDMG CIOQ switch for short) is proposed in [10]. The scheduling problem for the SDMG CIOQ switch is abstracted as a maximum bipartite *k*-matching problem. Using the fluid model, it is proven that any maximum *k*-matching algorithms on an SDMG CIOQ switch with an expansion factor 2 can achieve 100% throughput if input arrivals satisfy the strong law of large numbers and no inputs/outputs are oversubscribed. That study uses uniform traffic and polarized traffic. [11] and [12] address the problem of the existence of local scheduling policies that guarantee 100% throughput in a network of Input Queued and CIOQ switches. The strong law of large numbers and no oversubscription on any link in the network are the only assumptions on the input traffic. While 100% throughput can be achieved in [10–12], it is not clear whether service can be differentiated for different traffic types in all these scheduling algorithms.

This paper covers the design of a heuristic scheduling algorithm for a core optical router with the objective to support heterogeneous traffic types and to achieve service differentiation requirements of different traffic types. We address this issue by developing a fast and effective heuristic approximation to the optimization model, which aims to maximize the QoS value of the whole system by giving priority to the flows with higher QoS values. The proposed heuristic scheme is both starvation free and lock-out free. The heuristic algorithm, in general, results in a solution close to the optimal Linear Integer Programming solution, but the heuristic is much faster and easier to implement. The performance of the heuristic scheduling algorithm for a multi-service high capacity optical core router is verified by numerical experiments.

The rest of the paper is organized as follows: Section 2 briefly describes the core optical router architecture for which the algorithm can be deployed; Section 3 provides the details of the scheduler designs for the core optical router; Section 4 presents traffic modeling and simulation results of the proposed scheduling algorithm with discussions; Section 5 concludes this study.

2. Optical Switch Architecture

The proposed scheduling algorithm uses the core optical router architecture as described in Stanford's "OR (Optical Router) Project" [1]. The base model of the proposed system architecture is characterized as a core optical router surrounded by remote edge routers. The edge routers are connected to the core using wavelength division multiplexing (WDM) links. In particular, we use the configurations in Figure 1 to describe the proposed scheduling algorithm. However, our solution is not limited to these particular architecture configurations and parameters shown.

The core optical router consists of three stages: 1 core switch fabric, 4 (ingress, egress) edges, and 4 (ingress, egress) ports or Line Cards (LC) per edge. Each port carries OC-48 or 2.5 Gbps traffic, leading to an edge capacity at OC-192 or 10 Gbps. A port generates 16 virtual waves, each at a bandwidth of OC-3. Each edge is connected to the core using WDM links with 64 (16x4) virtual waves, for an aggregated capacity of 10 Gbps from each edge and a combined 40 Gbps for 4 edges. The core scheduler determines the scheduling patterns to grant, which changes as a function of the input traffic characteristics. For this particular optical switch, a fixed length scheduling cycle consists of 64 wave slots, with each wave slot at 1 μs. During each of 64 wave slots, the core switch fabric is capable of establishing a different mesh connectivity pattern from the ingress edges to the egress edges. With the given capacity and the size of the scheduling cycle, each wave slot will be able to switch a payload of 1250 bytes (=OC-192 * 1 μs). The core switching granularity is OC-3 so bandwidth for port-to-port connection has to be allocated in increments or multiples of OC-3. The core dynamic scheduler remains in effect until a new schedule is deployed.

2.1. *Ingress Processing*

We assume five traffic types are supported in the proposed optical core router:

- TDM traffic – the equivalent of "circuit-switched" or constant bit rate traffic;
- MPLS traffic – flow based core IP traffic for traffic engineering;
- DiffServ Class of Service 1 (DFS1)
- DiffServ Class of Service 2 (DFS2)
- Best Efforts (BE)

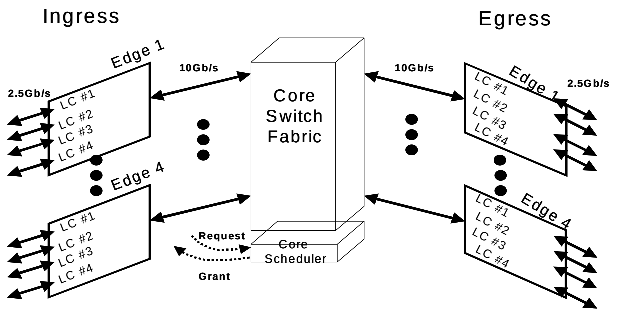


Figure 1. Optical router architecture

Traffic enters the optical router through a port in the ingress edge. An ingress port will either support TDM traffic (TDM port) or IP traffic (Packet over SONET or POS port), but not both. Each ingress port maintains a set of input queues, one for each ordered pair (egress port, QoS class). There are four IP QoS classes, one each for MPLS, DFS1, DFS2, BE traffic. Under these assumptions, there are in total 64 (16 egress ports * number of QoS/port (e.g. 4)) input queues for each POS port. For a TDM port, only 4 input queues are maintained, which enables the TDM ingress port to transport to any of the 4 egress TDM ports. The incoming packets are inserted into one of the input queues for that ingress port, based on its egress edge/port address and its QoS index. Figure 2 is the conceptualized diagram for the ingress POS port, which has three key elements: input queues, traffic manager and port scheduler. Figure 3 gives the picture for ingress edge processing. For MPLS and IP traffic, the Packet Classifier identifies the destination edge and port, the QoS queue, and QoS parameters based on the built-in MPLS/OSPF routing tables. The MPLS and IP packets are then inserted into appropriate input queues based on the routing information and QoS index, while waiting to be scheduled through the core. The traffic manager periodically monitors all the input queues and collects necessary statistics for the port scheduler. Port scheduler consolidates the port statistics and sends them to the edge and core schedulers.

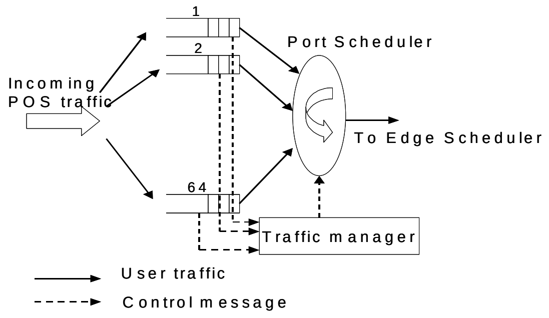


Figure 2. Ingress POS port processing

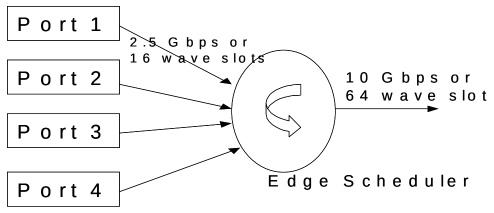


Figure 3. Ingress edge processing

2.2. *Optical switch fabric*

The optical core is essentially a fast switching fabric using a 4-by-4 crossbar Time Division Multiplexing (TDM) switch with a fixed duration wave slot at 1 μs. It creates a virtual fully connected mesh between ports by periodically reconfiguring the core to allow exchange of data from one ingress port/edge to another egress port/edge, as shown in Figure 4. Each edge sends a payload of no more than 1250 bytes on every wave slot. The packet transmission from all ingress ports/edges is synchronized with the switching cycle of the space switch fabric in the core so that the data is switched to the appropriate egress ports without any contention in the core. Each egress edge has a copy of the current schedule and uses it to route all received traffic to the appropriate egress port within that edge.

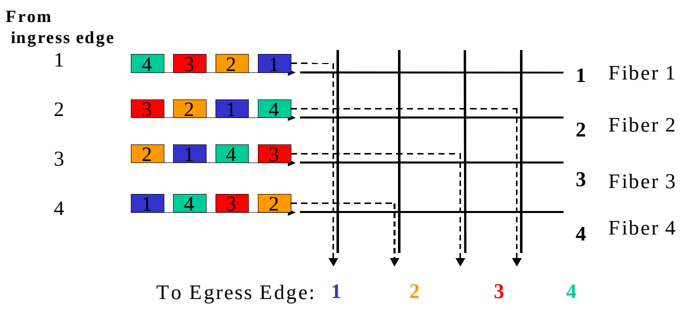


Figure 4. Optical core switching

3. Scheduler Design

The scheduler is the central brain of the entire system. The proposed scheduler provides a schedule that will serve the highest priority traffic available and guarantee that there is no starvation and no lock-out during switching. Since traffic patterns change over time, the scheduler must also adapt to these changes. A new schedule is created when there are new TDM/MPLS connections accepted or when the current schedule pattern does not perform satisfactorily any more. When the core scheduler determines that a new schedule is needed, it solicits the traffic demands from each ingress port. The scheduler computes and deploys a new schedule based on the value and urgency of each port-to-port connection. The proposed dynamic scheduling procedure can be described conceptually as a two-state scheduling operation shown in Figure 5. In steady state, the existing schedule pattern will be repeated as long as the performance of the schedule remains sufficient.

The proposed scheduler design is based on the following two-phase algorithm:

- Step 1: Wave Slot Definition (WSD) determines the 'best' set port-to-port connections to make during a fixed scheduling cycle.
- Step 2: Wave Slot Assignment (WSA) determines the ordering and timing of wave slots for the schedule that guarantees no blocking in the core.

The WSD phase (optimal or heuristic) determines the set of port-to-port connections, while the WSA phase defines the scheduling frame of individual connections.

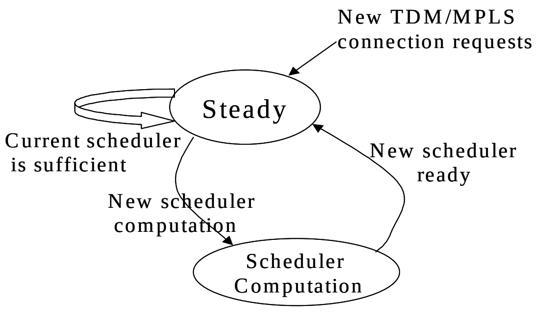

New TDM/MPLS connection requests
Steady
Current scheduler is sufficient
New scheduler ready
New scheduler computation
Scheduler Computation

Figure 5. Scheduling state diagram

3.1. *WSD Optimization Model*

The problem of determining a dynamically changing schedule can be formulated as a Linear Integer Programming problem. At ingress POS port *i*, all the input queues destined to the same egress port *j* (in total there should be 4 such queues per POS card) can be virtually consolidated into one queue, indexed by virtual queue (*i, j*). The virtual queue is computationally re-segmented into units at size of 1250 bytes, with each 1250 bytes traffic defined as a *payload*. Assume the given QoS value for type *m* traffic is q_m, $m=1,...,4$. Then the total QoS value for a type *m* payload is 1250* q_m. Let V_{ijk} represent the QoS value of sending the *k-th* payload of the virtual queue (*i, j*). The optimal schedule with the blocking restrictions can be represented mathematically as a Linear Integer Programming (LIP) problem.

Parameters:
I: the set of ingress ports;
J: the set of egress ports ($|I|=|J|$ *in our model*);
U: the number of wave slots allocated to each port;
K_{ij}: the set of payloads in the virtual queue (*i, j*);
v_{ijk}: QoS value for the *k*-th payload of the virtual queue(*i. j*).
Decision Variables:
$x_{ijk} = 0$ if no connection is made for the *k*-th payload of the virtual queue *(i, j)* during the scheduling cycle.

= 1 otherwise.

The ILP model:

$$\max \sum_{i\in I}\sum_{j\in J}\sum_{k\in K_{ij}} v_{ijk} x_{ijk}$$

$$s.t. \sum_{i\in I}\sum_{k\in K_{ij}} x_{ijk} \le U \quad \forall j \in J$$

$$\sum_{j\in J}\sum_{k\in K_{ij}} x_{ijk} \le U \quad \forall i \in I$$

$$x_{ijk} : 0 \text{ or } 1.$$

For the router configuration given in Section 2, we have *I = J = U=16*.

The solution to this LIP model will:

(i) Create port-to-port connectivity in an optimal way such that maximal total QoS values will be achieved.
(ii) Assign at most 16 wave slots to each ingress port, hence 64 wave slots to each ingress edge for a scheduling cycle.
(iii) Assign at most 16 wave slots to each egress port, hence 64 wave slots to each egress edge for a scheduling cycle.

The ILP problem above is a unimodular model that has integrality properties [15]. In other words, there exists an integer optimal solution to the noninterger linear model that relaxes the integer requirements, i.e., $0 \le x_{ijk} \le 1$. The detailed proof of integrality property is provided in the Appendix. Like any other pure Linear Programming models, the WSD optimization model is not NP-Complete. To further study its computational complexity, we have the following Theorem 1.

Theorem 1: The WSD optimization problem is equivalent to a Weighted Bipartite Matching Problem (Assignment Problem in Operations Research) with $2U|I|$ nodes and $(U^2|I|)^2$ acres.

The proof and an example are presented in the Appendix.

Though a polynomial-time algorithm can provide the optimal solution, the computational complexity of the WSD optimization problem is $O(U^3|I|^3)$ if Dijkstra's shortest path algorithm is applied [15] to solve the equivalent Weighted Bipartite Matching Problem. Solving for the optimal schedule is too time-consuming and is not real-time applicable, especially when the router size grows. To address the need for a faster scheduler, we define a heuristic approximation to the ILP model in the following discussion.

3.2. *WSD frozen heuristic algorithm*

This algorithm constructs a schedule that transmits the highest-valued traffic possible on a *port-by-port* basis. It differs from the more complex and time-consuming optimal algorithm, which chooses the overall highest-valued schedule and solves a global maximization problem. Though not yielding an optimal solution, the heuristic algorithm is fast and provides a 'good' schedule most of the time.

The heuristic algorithm uses the same traffic demands to determine a high-priority schedule. At the first iteration, the algorithm considers the 16 highest priority payloads from each ingress port. TDM flows or CAC based MPLS flows are treated with the highest priority. For example of *U=16*, *16* payloads from each ingress port results in an *average* of *16* payloads per egress port. If the first iteration results in exactly 16 payloads for each egress port, the schedule is complete. More likely, however, some egress ports will be assigned more than 16 while others will be assigned fewer than 16. For each egress port with more than 16 payloads, retain only the highest-valued 16 and delete the remaining payloads. Now all ingress and egress ports have either 16 or fewer payloads assigned. Every ingress/egress port with exactly 16 payloads is "frozen": no payload will be added into or removed from a frozen port. Further more, the payloads associated with frozen ingress and egress ports are also frozen. Thus, a frozen payload could be assigned to the following ingress and egress port combination:

(i) a frozen ingress port + an unfrozen egress port;

(ii) an unfrozen ingress port + a frozen egress port;

(iii) a frozen ingress port + a frozen egress port.

Now, the algorithm reaches the end of the first iteration. At the end of each iteration, check if all ports have 16 payloads assigned. If so, the schedule is complete. If not, perform the next iteration. In the new iteration, we add the highest-valued payloads among the non-frozen payloads to the unfrozen ingress ports to bring the total, including the frozen payloads, up to 16. Then repeat the actions for the egress ports introduced before.

Theorem 2: The algorithm must freeze at least one egress port and/or one ingress port after each iteration. Therefore, it is guaranteed to end within finite steps.

Proof: We assume input queue always has traffic to send; thus there will be no empty wave slots in the schedule. If this is not true, we just insert empty payloads to fill up the scheduling frame. Assume there are *n* unfrozen ingress ports and *m* unfrozen egress ports at an iteration. Initially, $n=m=16$. The ingress side has the same number of frozen payloads as the egress side at any iteration because any payload is indexed by an ingress port and egress port pair. At the beginning of each iteration, we add new payloads with the highest QoS values to bring the total payloads, including the frozen payloads, up to 16 for each unfrozen ingress port. All the new payloads can only go to non-frozen egress ports. Thus, 16 payloads from each ingress port result in an average of 16 payloads per unfrozen egress port. So at least one unfrozen egress port can be frozen at an iteration. □

The flowchart for the heuristic algorithm is shown in Figure 6.

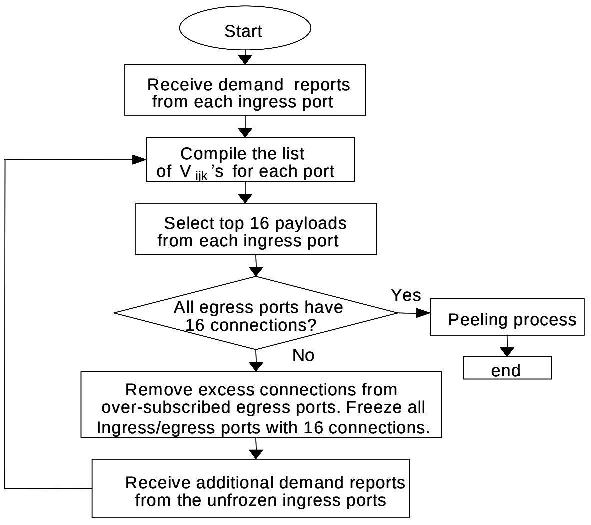

Start
Receive demand reports from each ingress port
Compile the list of V_{ijk}'s for each port
Select top 16 payloads from each ingress port
All egress ports have 16 connections?
Yes
Peeling process
end
No
Remove excess connections from over-subscribed egress ports. Freeze all Ingress/egress ports with 16 connections.
Receive additional demand reports from the unfrozen ingress ports

Figure 6. Frozen Heuristic Algorithm Flow Chart

We provide an example here to further explain how the frozen heuristic algorithm works. Suppose there are three ports, and each port can make a *single* connection during a scheduling cycle (*U=1*). Table 1 identifies the ranked values (Val) and the egress port (EP) for three different offered payloads for ingress ports 1, 2, and 3. Note that the offers are ranked in the order of QoS values from each ingress port.

Ingress Port	Offer 1		Offer 2		Offer 3	
	Val	EP	Val	EP	Val	EP
1	17	1	12	3	12	2
2	20	2	19	1	10	3
3	25	2	23	3	10	1

Table 1. Offered Traffic Demands

The first demand report will offer:

(i) 1 payload from ingress port 1 to egress port 1 with value 17,

(ii) 1 payload from ingress port 2 to egress port 2 with value 20,

(iii) 1 payload from ingress port 3 to egress port 2 with value 25;

The frozen algorithm will accept ingress port 1 to egress port 1 connection and ingress port 3 to egress port 2 connection. It then freezes ingress ports 1 and 3 and egress ports 1 and 2 (Figure 7.a) with all payloads associated with them. The only remaining available connection is port 2 to port 3 with a value of 10 (Offer 3) for a total value of 52 (Figure 7.b)

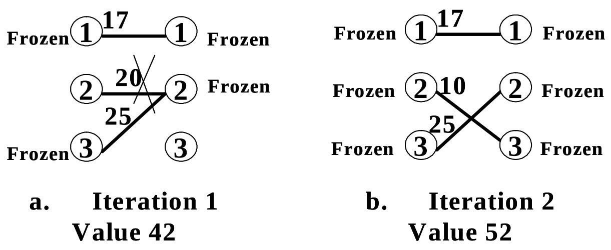


Figure 7. An example of frozen algorithm

Since WSD frozen heuristic algorithm can at least freeze *U* connection at each iterations, there are at most $|I|=|J|$ iterations. Therefore, the computational complexity is $O(U^2|I|)$ and is much lower than the optimization algorithm based on the equivalent Weighted Bipartite Matching Problem, which runs $O(U^3|I|^3)$ time.

3.3. *WSD non-frozen algorithm*

Freezing the ports early may prevent moderate valued connections from consideration. The problem is even more apparent in certain 'hot spot' conditions. Consider a simple hot spot example In Table 2, where each port will support only two connections (*U=2*) for a schedule, i.e., a schedule cycle only consists of two wave slots. The value of the initial two wave slots (Offer 1 and Offer 2) from the egress port view is: 75 units to EP 1, 370 units to EP 2 (the hot spot), and zero units to EP 3 in Figure 8. For this example, the frozen algorithm will produce a schedule with a total value of 295, while the non-frozen solution will have a higher value of 370.

Ingress Port	Offer 1		Offer 2		Offer 3		Offer 4	
	Val	EP	Val	EP	Val	EP	Val	EP
1	90	2	65	1	20	3	10	2
2	100	2	85	2	80	1	5	3
3	95	2	10	1	9	3	8	1

Table 2. Hot Spot at Node 2

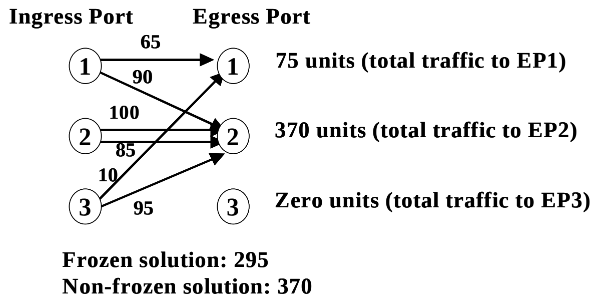
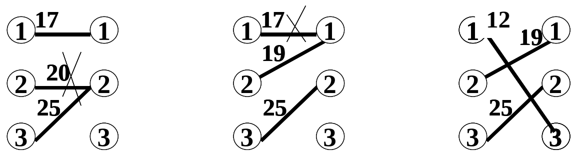

Ingress Port
Egress Port
65
90
100
85
10
95
1
2
3
75 units (total traffic to EP1)
370 units (total traffic to EP2)
Zero units (total traffic to EP3)
Frozen solution: 295
Non-frozen solution: 370

Figure 8. Example of Hot Spot Traffic

To address the hot spot problem, the non-frozen algorithm does not freeze the ports with *U* assigned payloads so that they can be improved at later iterations. However, any payloads that have been eliminated because of 'over subscription' on an *egress* port will not be reconsidered as a candidate for the future iterations.

At the end of each iteration, unless all ingress and egress ports have *U* connections, the algorithm proceeds to the next iteration by including the highest QoS value payloads not yet considered to bring up the total connections to *U* to each ingress whose number of assigned connection is less than *U*. These new offered connections may be targeted to egress ports that currently have *U* connections from the prior iteration. This operation could replace some of the earlier accepted connections with higher valued payloads. The WSD phase of the scheduling algorithm is considered to be complete if all ports have *U* connections or if no candidate connections are available at any of the undersubscribed ingress ports. The algorithm then proceeds to the WSA algorithm. Since there are a finite number of candidate connections at each ingress port, and each iteration at least freezes one port, the algorithm is assured to converge. The non-frozen algorithm will return either the same algorithm as the frozen algorithm, or a better one. This does not imply, however, that the non-frozen schedule will necessarily result in the optimal solution. Though the non-frozen algorithm in general takes more time than the frozen algorithm, its worst case computational complexity is still $O(U^2|I|)$.

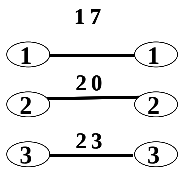
To further explain how the non-frozen heuristic algorithm works, we use the same example in Table 1 to compare the non-frozen algorithm with the frozen one. Though the non-frozen algorithm will begin the same assignment as the frozen algorithm, it will not freeze the ports that have *U* assigned connection (see Figure 9.a), but it will remove the connection from ingress port 2 to egress port 2 from the demand report because of oversubscription at egress port 2. In the second iteration as shown in Figure 9.b, the connection from port 2 to port 1 is allowed to replace the prior ingress port 1 to egress port 1 connection. The replacement is not an option under the frozen algorithm. This action now frees ingress port 1 to define a new connection. In the third iteration, ingress port 1 offers a payload of value 12 to egress port 3, which results in a total value of 56, higher than the value of 52 from the frozen algorithm in Figure 7.



a. Iteration 1 Value 42 **b. Iteration 2 Value 44** **c. Iteration 2 Value 56**

Figure 9. Non-Frozen Algorithm Results

With complete information, the optimal solution to this problem could be determined using the traditional algorithms for the equivalent Bipartite Marching Problem. The optimal solution to the example of Table 2 is illustrated in Figure 10 and has a higher objective function value than either solution above. In other words, neither the frozen nor the non-frozen algorithm provides the optimal schedule. They, however, produce a good approximation with a smaller amount of computational time.



a. Optimal Value = 60

Figure 10. Optimal assignments of wave slots

3.4. *WSA algorithm*

The heuristic and optimal algorithms create the port-to-port connections in a scheduling cycle. They do not, however, spread the connections into wave slots so that the following edge-to-edge restrictions are satisfied:

(i) During one wave timeslot, no more than one ingress edge (port) can be connected to an egress edge (port).
(ii) During one wave slot, an ingress edge (port) cannot be connected to more than one egress destination edge (port).

Thus the existing connections need to be further distributed into 64 wave slots that satisfy the above constraints. The WSD process will enforce these port and edge constraints. Notice that edge-to-edge restriction satisfaction implies port-to-port restriction satisfaction, not vice versa. Thus the port-to-port connectivity is first consolidated to its edge-to-edge equivalence. The port-to-port connectivity is expressed as a 16-by-16 matrix *C*, whose element *C(i,j)* represents the number of connections from port *i* to port *j* during a given scheduler cycle. Each row (column) of *C* adds up to 16. By combining port-to-port connections into edge-to-edge connections, we form a 4-by-4 edge connectivity matrix *A*, whose element *A(i,j)* represents the number of connections from edge *i* to edge *j* during a given scheduler cycle. Each row or column of *A* adds up to 64.

The WSA process begins by splitting the matrix *A* into two matrices *A1* and *A2*, each having the same dimension as *A* and with rows and columns adding up to 32. The same WSA process is then applied to the resulting 2 matrices, then to 4, 8, 16, 32 matrices. There are 6 separate and independent WSA iterations to produce the final 64 permutation connectivity matrices. For any WSA action *n* (*i=1,2…6*), the number of total resulting matrices is 2^n and the rows (columns) of each of the 2^n matrices add up to $64/2^n$. Thus the summation for each row (column) is 1 for each of the 64 matrices at the final step, which imposes the restriction that one ingress (egress) edge can only be connected to one egress (ingress) edge within that wave slot. The 64 permutation matrices represent 64 wave slots and indicate in the time domain how the connections are established. The flow chart for the detailed WSA algorithm at iteration *n* is shown in Figure 11, which elaborates on how edge matrix *A'* is successfully divided into two matrices *A1'* and *A2'*.

(i) If there is an even number of connections from edge i to edge j (e.g. 2a,) in A' matrix, then there will be a connections in each half of the scheduling cycle A1' and A2'. In another word, if A'(i, j) = 2a, then A1'(i, j) =A2'(i, j)= a.

(ii) If A'(i, j) = 2a+1, then A1'(i, j)= A2'(i, j) = a. There will be (2a+1)/2 (integer division) connections in each half of the scheduling cycle.

(iii) The remainders of the divisions from step 2 form a remainder matrix B of 0s and 1s, i.e., B=A'-A1'-A2'. Each row/column of B adds up to an even number. The 1s of matrix B are then distributed back to the two halves of the scheduling cycle A1’ and A2’ by alternately assigning 1s to either half of the scheduling cycle over Eulerian Circuits [13] found over the remainder matrix B. The procedure will guarantee that each row or column of A1’ and A2’ adds up to $64/2^n$.

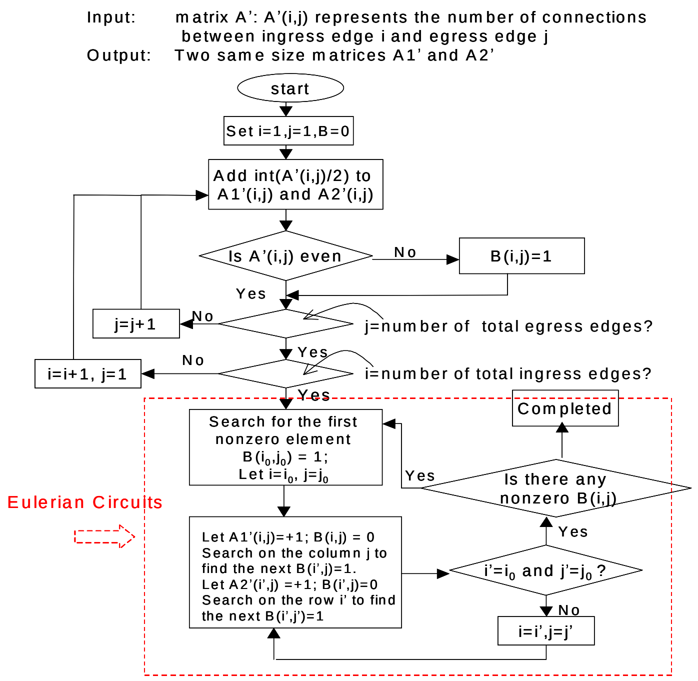

Input: matrix A’: A’(i,j) represents the number of connections between ingress edge i and egress edge j
Output: Two same size matrices A1’ and A2’
start
Set i=1,j=1,B=0
Add int(A’(i,j)/2) to A1’(i,j) and A2’(i,j)
Is A’(i,j) even
No
B(i,j)=1
Yes
j=j+1
No
j=number of total egress edges?
Yes
i=i+1, j=1
No
i=number of total ingress edges?
Yes
Completed
Search for the first nonzero element B(i0,j0) = 1; Let i=i0, j=j0
Yes
Is there any nonzero B(i,j)
Eulerian Circuits
Let A1’(i,j)=+1; B(i,j) = 0 Search on the column j to find the next B(i’,j)=1. Let A2’(i’,j) =+1; B(i’,j)=0 Search on the row i’ to find the next B(i’,j’)=1
Yes
i’=i0 and j’=j0 ?
No
i=i’,j=j’

Figure 11. WSA Algorithm Flow Chart

The following example further explains how WSA algorithm works. Table 3 gives an example of an *A* matrix that results from the WSD (either frozen or non-frozen) process. In this example we assume there are 4 ingress/egress edges. One scheduling cycle consists of 4 wave slots. As introduced in the WSA algorithm, *A(i,j)* represents the number of connections from ingress edge *i* to egress edge *j* during one scheduler cycle. For this example, during one scheduling cycle there are 2 connections for the edge pair (1,1), 0 for edge pair (1,2), 1 for edge pair (1,3), 1 for edge pair (1,4), 1 for edge pair (2,1), etc. However, matrix *A* does not tell when these connections can happen during a scheduling cycle in order to satisfy the edge-to-edge restrictions. WSA process splits matrix *A* into 4 permutation connectivity matrices, *A1*, *A2*, *A3*, *A4*, with each of them corresponding to one wave slot, as shown in Table 4. For example, *A1* tells that the following port-to-port connections can be supported simultaneously in one wave slot without violating edge-to-edge restrictions: (1, 1), (2, 2), (3, 3), (4, 4). Since all 4 matrices are permutation matrices, one ingress edge can only communicate one egress edge in one time slot during the scheduling cycle. Thus edge-to-edge restrictions are enforced by WSA process.

A matrix

ingress/egress #	**1**	**2**	**3**	**4**
1	2	0	1	1
2	1	2	1	0
3	0	1	2	1
4	1	1	0	2

Table 3. An *A* matrix from WSD process

A1 matrix

ingress/egress #	1	2	3	4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

A2 matrix

ingress/egress #	1	2	3	4
1	0	0	1	0
2	1	0	0	0
3	0	0	0	1
4	0	1	0	0

A3 matrix

ingress/egress #	1	2	3	4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

A3 matrix

ingress/egress #	1	2	3	4
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	1	0	0	0

Table 4. Results from WSA process: A1- A4 matrices

4. Simulation Modelling and Performance Evaluations

4.1. *Traffic Modeling*

The described optical core router is able to support different traffic types. We suppose that one or more line cards can be dedicated to TDM traffic for each edge, and the rest line cards to Packet over SONET (POS) traffic. A TDM (or POS) ingress port only communicates to egress TDM (or POS) ports. Composition of traffic over each ingress POS port is distributed among the types of MPLS, DFS1, DFS2, and BE, and the total traffic added together is 100%.

The various traffic types are treated as follows:

(i) The interarrival time for TDM flow requests on each TDM port is Poisson distribution with an average rate of λ_{TDM} ms. Holding time is exponential distribution with a mean of μ_{TDM} ms. A TDM flow that cannot fit without blocking is ever rejected. Each TDM flow is generated at a rate of OC-12. For the results presented in this section, λ_{TDM} =20 ms and μ_{TDM} =100 ms.

(ii) MPLS connection requests are Poisson distributed with an average rate of λ_{MPLS} ms, and the average session holding time is exponentially distributed with a mean of μ_{MPLS} ms. For the results reported in this section, λ_{MPLS} =10 ms and μ_{MPLS} =100 ms. The MPLS connections generate IP packets with specified rate in the appropriate self-similar patterns. Individual MPLS Label Switched Path (LSP) is given an average rate of 250 Mbps, a peak rate of 430 Mbps and is characterized by a Hurst parameter of 0.7, to match the router bandwidth configuration.

(iii) For Diffserv and Best Effort traffic, only packet level modelling is needed. DFS1, DFS2, and BE packets are generated in self-similar patterns with a Hurst parameter of 0.7.

The POS packet size distribution is given in Table 3 [14]. In the simulation, the QoS values per byte for different traffic types are differentiated as follows: MPLS: 8; DFS1: 4; DFS2: 2; BE: 1.

Packet Size (Bytes)	Percentage
40	55
52	10
576	20
1500	15

Table 5. POS packet size distribution

4.2. *Performance evaluations*

Simulations and analysis have been conducted to evaluate the performance of the proposed scheduling algorithms in the core optical router architecture introduced in Section 2. There are 4 ports per edge and 4 edges in total. Each edge has one TDM card and 3 POS cards. Thus there are 4 TDM card and 12 POS cards in the router under study. The simulations have been performed in OPNET by assuming various traffic scenarios. In the results reported in this section, composition of traffic over each ingress POS port is: 40% of offered traffic for MPLS; 20% for DFS1; 20 % for DFS2; 20% for BE. The destination of a TDM flow is uniformly selected among 4 TDM egress ports. The probability distribution of the traffic from a particular ingress POS port is as follows: 75% of the traffic from a particular POS ingress port will be distributed uniformly to 2 H-POS egress ports, 20% will be distributed uniformly to the other 4 M-POS egress ports and 5% will be distributed uniformly to the remaining 6 L-POS egress ports. The above distribution comes from the assumption that traffic tends to go to few popular destination ports. Small background/best-effort traffic (5%) is distributed uniformly (e.g., emails) to all 12 POS egress cards. In order to maintain similar traffic loads on 12 POS egress ports, the connection distribution will be rotated over ingress ports, as shown in Figure 12. The rotation eliminates the effects of traffic congestion because of the traffic distribution, so the simulations can evaluate scheduling performances better. All patterns are rotated, i.e., ingress port *i* distribution becomes ingress port *i+1* distribution, once every 30 ms. The variations on distribution shown in Figure 12 is introduced for the purpose of studying the adaptability of the scheduling algorithm to changes in the traffic patterns.

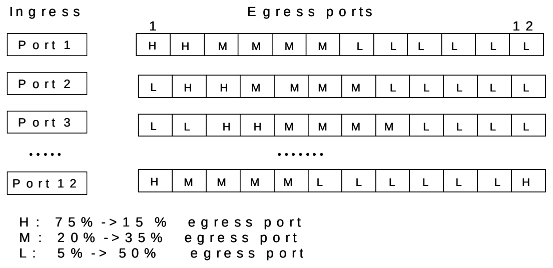


Figure 12. POS Traffic distribution

Figure 13 shows that, at the steady state, the offered load to the system and the throughput are both 23.5 Gbps by using non-frozen heuristic. Among the 23.5 Gbps, 8.5 Gbps (85% of 10 Gbps total TDM capacity) is TDM traffic and, 15 Gbps (50% of 30 Gbps total POS card capacity) is POS traffic. Theoretically TDM could achieve 100% throughput or 10 Gbps rate. The actual throughput is only 8.5 Gbps because a new request of each ingress TDM port has a random TDM destination. As more of the $4 \times 4 = 16$ available TDM connections are allocated, new requests are less likely to be accepted.

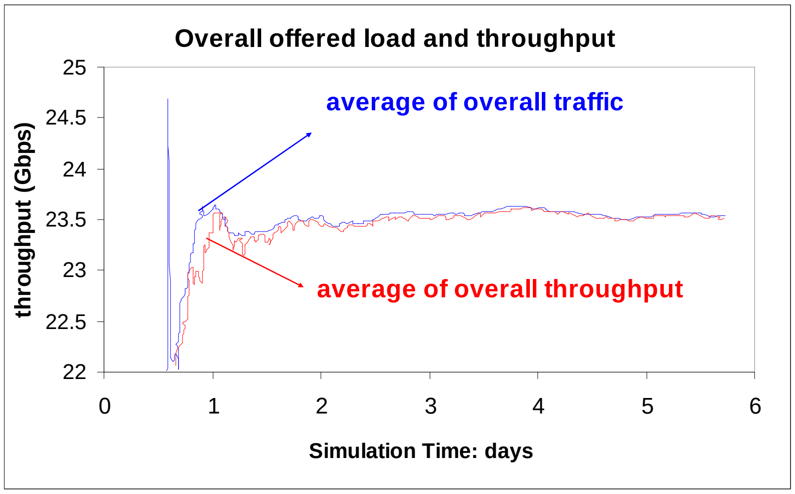


Figure 13. Overall Offered Load And Throughput

The non-frozen heuristic schedulers are evaluated in three slightly different versions: classical scheduler (CLS), scheduler with limited average rate allocation (LARS) and scheduler with limited peak rate allocation (LPRS).

(i) CLS does not reserve any bandwidth for any POS traffic. CLS is slow in reacting to new traffic since entire schedule computation process takes about 2 ms or 32 scheduler cycles.

(ii) LARS reserves average rate equivalent wave slots for each MPLS flow before acceptance. The average MPSL flow rate is 250 Mbps. Notice that two wave slots actually correspond to a bandwidth of 312.5 Mbps. Any other traffic on the same port-to-port combination may temporarily use the two wave slots in periods of low MPLS traffic. Extra MPLS traffic beyond 312.5 Gbps will be queued and compete for the bandwidth as best effort traffic.

(iii) LPRS is similar to LARS, with the difference that the guaranteed number of wave slots covers the peak rate of MPLS flows. We assume the peak rate is 430 Mbps, which requires 3 wave slots. Unused reserved bandwidth can be temporarily utilized by other POS traffic.

Figure 14 and Figure 15 display the steady state average delay and delay distributions for MPLS traffic under the three versions of the scheduler. As expected, among these schedulers, LPRS achieves the lowest average delay at 40.5 µs and smallest 90 percentile at 73 µs, since peak rate bandwidth is reserved ahead of time. LARS, on the other hand, results in highest average delay at 78 µs and highest 90-percentile delay at 180 µs. LARS guarantees MPLS bandwidth based on average rate, and the remaining traffic is considered as best effort. Thus, the delay is pushed higher by the portion of best-effort treated traffic. CLS results in an average delay of 43 µs and a 90 percentile of 75 µs. Although CLS always considers MPLS traffic the highest priority among POS traffic, it is slow in reacting to traffic changes. Thus CLS experiences higher average delay than LPRS does.

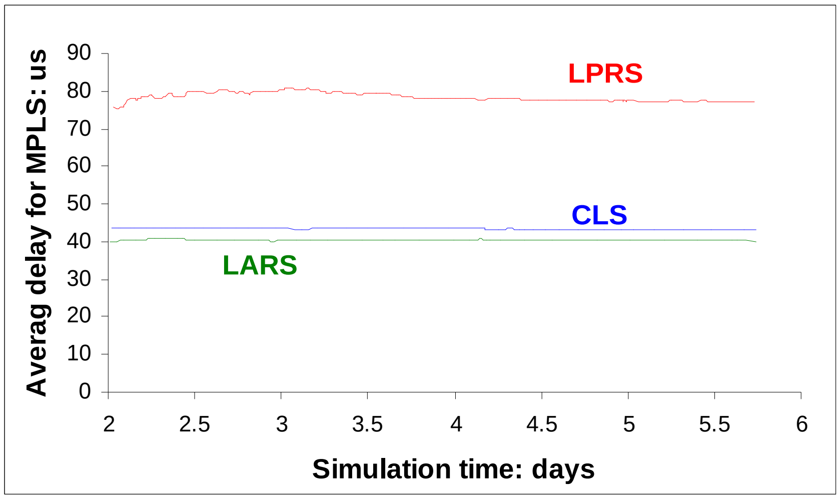


Figure 14. Steady state average delay for MPLS

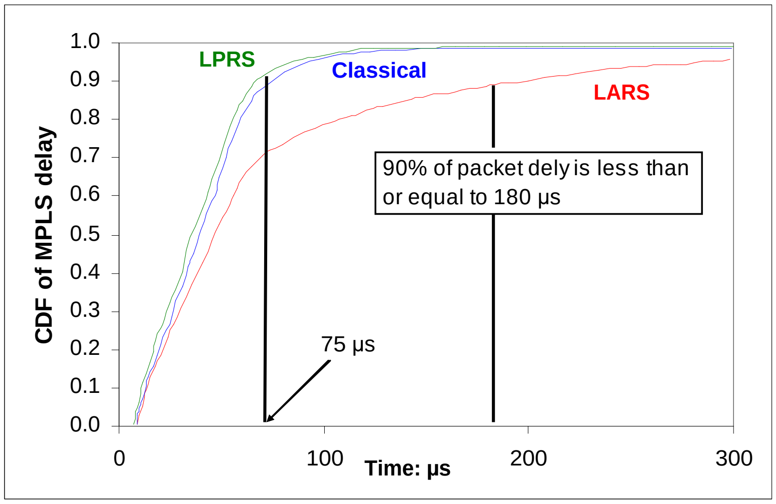


Figure 15. CDF for MPLS delay distribution

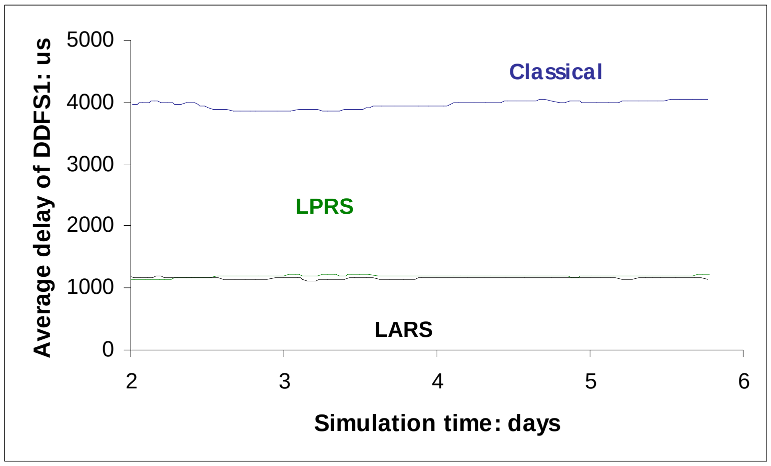


Figure 16. Steady state average delay for DFS1 traffic

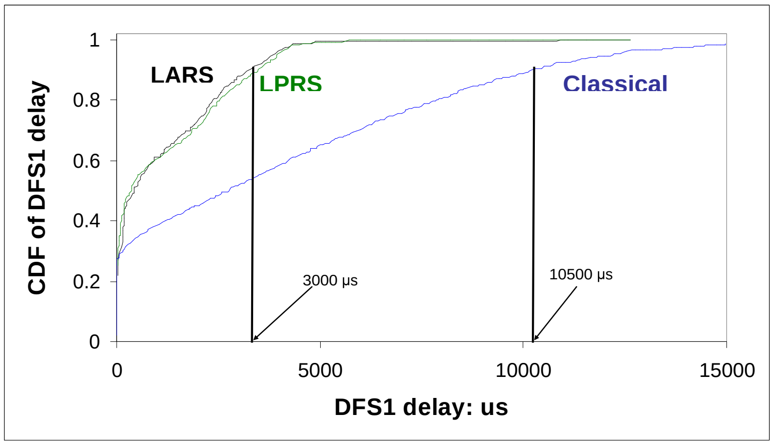


Figure 17. CDF for DFS1 traffic

In Figure 16 and Figure 17, DFS1 traffic experiences lowest average delay at approximately 1200 µs and lowest 90 percentile at 2950 µs by LARS. LARS limits the bandwidth for MPLS traffic and consequently leaves more room for lower priority traffic such as DFS1. Although LPRS reserves peak rate equivalent bandwidth for MPLS, unused reserved bandwidth can be temporarily utilized by other POS traffic of the same port-to-port combination. LPRS achieves very similar results to those of LARS with a slightly higher average delay and 90 percentile. However, CLS, which always favours TDM and MPLS traffic, degrades DFS1 traffic performance greatly with an average delay at 4050 µs and 90 percentile at 10,500 µs. This performance degradation is mainly caused by the long scheduler computation time and oscillations on bandwidth allocation. In fact, a schedule is in effect for approximately 2,000 µs or about 32 scheduling cycles. This is the amount of time we estimated for computing a new 64-wave-slot schedule. DFS2 and BE traffic experience very similar performance trends as compared to DFS1 traffic. Table 6 summarizes the simulated behaviour of the 4 types of POS traffic under the 3 versions of the scheduler.

Traffic type and delay: µs		Classical	LPRS	LARS
MPLS	Average	44	40.5	78
	90 percentile	75	73	180
DFS1	Average	4,050	1,210	1,200
	90 percentile	10,500	3,000	2,950
DFS2	Average	5,400	2,370	2,350
	90 percentile	12,000	4,500	4,400
BE	Average	7,550	3,800	3,700
	90 percentile	15,500	6,750	6,500

Table 6. POS delay performance under 3 schedulers

Figure 18 shows the average port connectivity for POS ports, i.e., the number of egress ports reached during 16 wave slots of a port schedule, averaged over all 12 POS ports. It is an indirect indication of how well connectionless traffic, i.e., DFS1, DFS2, BE, is served by the scheduler. Higher connectivity means more queues can be served during a scheduling cycle, thus causing smaller average delay and jitter. LARS achieves slightly higher connectivity than LPRS, and both clearly outperform CLS with values around 7.5 out of 12, as opposed to 4.5 out of 12. Small connectivity of CLS leads to oscillating behaviour and longer delay/jitter, as apparently shown in results presented earlier.

We assessed the accuracy of the heuristic through the ratio of the total values achieved by optimal and heuristic schedulers.
To evaluate the algorithms, we generated 250 random instances of traffic to fill the set of demand reports for each port. For each instance we used the same demand report data to compute:

- M, the value of traffic carried through the switch as determined by an optimal solution to the integer programming problem presented earlier; and
- *M'*, the value obtained through application of the heuristic algorithm (frozen or non-frozen).

The assessed accuracy is expressed as the ratio of *M'/M*. As shown in Figure 19, when compared with the frozen algorithm, the non-frozen solution typically demonstrated 5 to 10 percent improvement. In the extreme hot spot cases where the frozen algorithm produced a solution in the range of 10% of the optimal solution, the non-frozen algorithm improved the solution to about 85% of the optimal solution. The improvements of the frozen algorithm however came at a cost of 33% to 48% increase in execution time compared with the non-frozen algorithm. Both heuristic algorithms need much less computation time than the optimal algorithm. The calculation time for the optimal schedule is approximately 125 ms compared with 2.5 ms for the non-frozen heuristic algorithm in C for the 4 edge 16 port optical system. With the scale of the system increase, the computation time required for the optimal scheduler is expected to go up much faster than the heuristic algorithms do, which further limits the possibility of optimal algorithm implementation in practice.

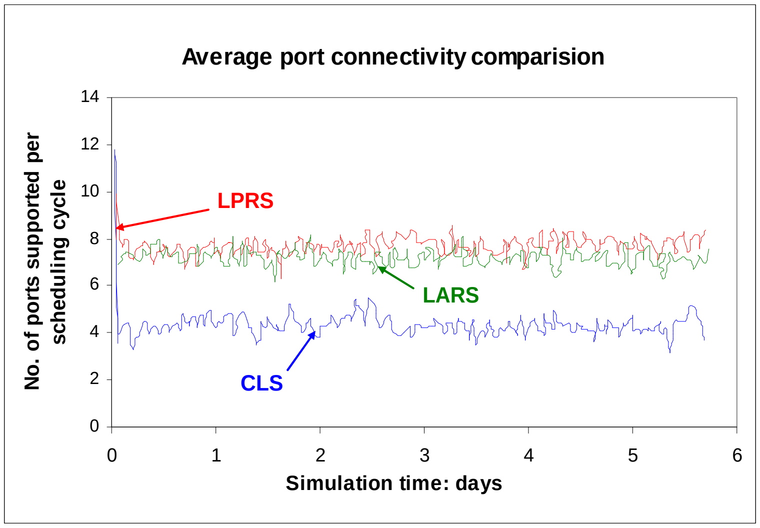


Figure 18. Average port connectivity comparison

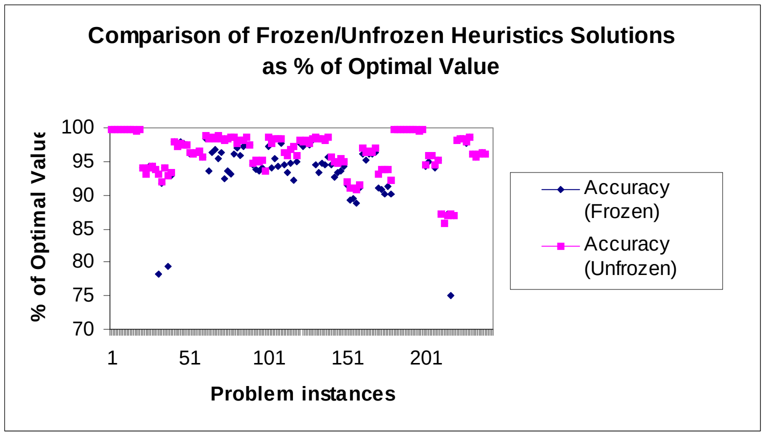


Figure 19. Comparison of Heuristic and optimal solutions

5. Conclusions

This paper addresses a new scheduling problem in the high capacity core optical switching systems for heterogeneous traffic. The scheduling problem is formulated as a Linear Integer Programming model. Two heuristic algorithms, frozen and non-frozen, are developed to solve the problem with much less time than optimality algorithm. Due to the large capacity of the system and the long computation time of the scheduler, bandwidth efficiency and smooth transitions between the consecutive schedules are very critical to the traffic performance such as delay and jitter. The heuristic scheduling algorithms are evaluated in three different versions: classical scheduler (CLS), limited with average rate scheduler (LARS) and limited with peak rate scheduler (LPRS). For the investigated core optical system, bandwidth reservation ahead of traffic arrival for high priority traffic is very beneficial to QoS support for all traffic types in the system.

Appendices

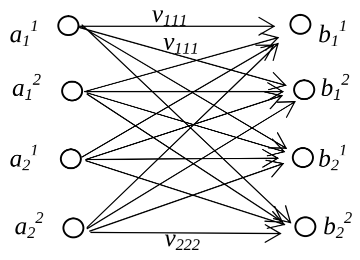
1. The poof of integrality property of the optimization model

Proof: Without loss of generality, we assume a fractional optimal solution with x_{111} as a fraction. Since x_{111} is a fraction, there must be at least another fraction of x_{1jk} ($j\neq 1$ and/or $k\neq 1$), say x_{122}, to satisfy the constraints that $\sum_{j\in I}\sum_{k\in K_{1j}} x_{ijk}$ must be an integer. Similarly if x_{122} is a fraction, there should be another fractional connection x_{i2k} ($i\neq 1$ and/or $k\neq 2$). Therefore, a factional cycle can be found. If we assume $U=2$, a fractional cycle example can be $x_{111}=0.2$, $x_{122}=0.8$, $x_{223}=0.2$, $x_{211}=0.8$, in which all numbers are fractions. If $v_{111}+v_{223} \geq v_{122}+v_{211}$, the new feasible solution of $x_{111}=1$, $x_{122}=0$, $x_{223}=1$, $x_{211}=0$ and the same values for all other x_{ijk} has at least the same objective function value as the original fractional solution. If $v_{111}+v_{223} \leq v_{122}+v_{211}$, another feasible solution of $x_{111}=0$, $x_{122}=1$, $x_{223}=0$, $x_{211}=1$ and the same values for all other x_{ijk} is at least as good as the original fractional solution. With more ports, the new constructed feasible solutions may still have fractional numbers. However, for any fractional solution, we can always find a solution with more integer numbers that is better or the same. Thus, an optimal integer solution can be found to the linear relaxed model. Obviously it is also the optimal solution to the ILP. □

2. Proof and example for Theorem 1.

Proof: WSD optimization model can be trasformed into a Weighted Bipartite Matching Problem (Assignment Problem) by duplicating each ingress and exgress port U times and having one arc from each orgin node to each destination node. The benefit on the arc from a_i^m to b_j^n is v_{ijm}.

For example, assume a WSD optimtiztion problem with $|I|=|J|=U=2$ and $K_{ij}=2$ (if $K_{ij}=1$, we can have $K_{ij}=2$ by letting $v_{ij2}=0$. The WSD optimization model can be represented by a graph of the Weigted Bipartite Matching Problem as follows.



The benefit on the arc from a_i^m to b_j^n is v_{ijm}. In the graph, we duplicate each ingress a_i and egress port b_j twice. In the Weighted Bipartite Matching problem, only one arc can be chosen for each node, so there are exactly $U=2$ arces (loads) for each original port. Since both $U=2$ v_{ijm} are connected to the new node a_i^m, at most one v_{ijm} is included in the solution. Therefore, the developed Weighted Bipartite Matching problem is equivalent to the WSD optimizaiton model. □

References

[1] http://klamath.stanford.edu/or/

[2] Yun, K. Y., , "A Terabit Multiservice Switch", IEEE Micro, Vol. 21, No. 1, pp.58-70, January/February 2001.

[3] Verma, S., H. Chaskar, and R. Ravikanth, "Optical Burst Switching: A Viable Solution for Terabit IP Backbone", IEEE Network, Vol. 14, No. 6, pp.48-53, November/December 2000.

[4] Chang C. S. , W. J. Chen and H.Y. Huang, “Birkhoffvon Neumann input buffered crossbar switches”, Proceedings of IEEE INFOCOM 2000, Vol.3, pp.1614-1623, March 2000.

[5] Chang C. S. , W. J. Chen and H.Y. Huang, “On service guarantees for input buffered crossbar switches: a capacity decomposition approach by Birkhoff and von Neumann”, Proceedings of IEEE IWQoS’99, pp.79-86, 1999.

[6] Baw, C. S., R. D. Chamberlain, and M. A. Franklin, “Fair Scheduling in an Optical Interconnection Network”, Proceedings of 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp.56-65, October 1999.

[7] Bo Li, and Yang Qin, “Traffic Scheduling in a Photonic Packet Switching System with QoS Guarantee”, Journal of Lightwave Technology, Vol.16, No.12, pp.2281-2295, December 1998.

[8] McKeown, N., “The *i*SLIP Scheduling Algorithm for Input-Queued Switches”, IEEE/ACM Transactions on Networking, Vol.7, No.2, pp.188-201, April 1999.

[9] McKeown, N., A. Mekkittikul, V. Anantharam, and J. Walrand, “Achieving 100% Throughput in an Input-Queued Switch”, IEEE Transactions on Communications, Vol.47, No.8, pp.1260-1267, August 1999.

[10] Yang, M. and S. Q. Zheng, “An efficient scheduling algorithm for CIOQ switches with space-division multiplexing expansion”, Proceedings of IEEE INFOCOM 2003.

[11] Marsan, M. A., P. Giaccone, E. Leonardi, and F. Neri, “Local Scheduling Policies in Networks of Packet Switches with Input Queues”, Proceedings of IEEE INFOCOM 2003.

[12] Marsan, M. A., P. Giaccone, E. Leonardi, and F. Neri, “On the Stability of Local Scheduling Policies in Networks of Packet Switches With Input Queues”, IEEE Journal on Selected Areas in Communications, Vol.21, No.4, pp.642-655, May 2003.

[13] Skiena, S. “Eulerian Cycles”, Addison-Wesley, pp. 192-196, 1990.

[14] Thomposn, K., G. J. Miller, and R. Wilder, “Wide-Area Internet Traffic Patterns and Characteristics”, IEEE Network, Vol. 11, No. 6, pp.10-23, November/December 1997.

[15] Ahuja, R. K., T. L. Magnanti, J. B. Orlin, “Network Flows”, Prentice Hall, New Jersey, 1993.