Journal of Mathematics Research

http: www.ccsenet.org/jmr

Vol. 1, No. 2

September 2009

Accounting Information Systems Genetic Algorithms for All-Optical Shared Fiber-Delay-Line Packet Switches

Soung Yue Liew

Universiti Tunku Abdul Rahman

Petaling Jaya 46200, Malaysia

E-mail: syliew@utar.edu.my

Edward Sek Khin Wong (Corresponding author)

Faculty of Business and Accountancy University of Malaya

Kuala Lumpur 50603, Malaysia

E-mail: edwardwong@um.edu.my

Abstract

All-optical shared fiber-delay-line (FDL) packet switches have been studied intensively in the literature and with the literature, many scheduling genetic algorithms have been proposed. However, these genetic algorithms suffer from not being able to provide a delay bound, or require complex timing methods to compute scheduling assignments. In this paper, we propose two fast scheduling algorithms for all-optical shared-FDL packet switches. In the first algorithm, packet scheduling is formulated as a tree-searching problem. This is accomplished by breaking down the search tree into multiple smaller subsets and assigning each subset to a parallel processor. By using this method, scheduling solutions can be obtained in a shorter time. Although this approach is superior to other algorithms, its overall complexity and processing overheads are still too high to warrant its day to day use. In the second algorithm, the search tree is carefully trimmed down in order to reduce complexity and overheads. The conclusion will consider a 32×32 switch with 32 FDLs, and assume a processor clock rate of 200MHz for schedulers. With this new and second algorithm, a scheduling assignment can be calculated for a given packet in 30ns if 8 parallel processors are employed and we show by simulation that both algorithms can achieve a loss rate of $\sim 10^{-7}$ even at load 0.9, where the average delay is 11.5 timeslots.

Keywords: Information Systems, Genetic algorithms, Optical communications, Shared memory system, Optical fiber delay lines, Scheduling

1. Introduction

The critical issue in all-optical packet switching is the communication contention problem experienced among the packets of the same wavelength when they arrive at a switch and are requesting the same output facility. To resolve contention, either fiber delay lines (FDLs) are employed as a form of optical memory to buffer the contended packets (J. Ramamirtham, 2003; M. J. Karol, 1993; S. Y. Liew, 2005), or the wavelengths of the contended packets are required to be converted to other and unused wavelengths in order to avoid communication conflict (J. Li, 2007). In this paper, we focus using the buffering approach, though both contention resolution schemes, buffering and wavelength conversion, can be applied at the same time to obtain a hybrid solution (Z. Zhang, 2006), (E. Wong, 2006).

Here, using the buffering contention-resolution approach, contended packets can be routed to FDL coils of different lengths for buffering. However, since FDLs are bulky, only a limited amount of buffer facility can be included in the building of an all-optical buffered switch and from this limitation, it becomes a challenging issue to efficiently manage these precious FDL resources.

In (J. Ramamirtham, 2003), an all-optical input-buffered (or, input-FDL) switch has been studied and an efficient scheduling algorithm has been proposed using time-sliced burst switching. Although this proposed scheme can achieve high throughput, the buffer requirement is undesirably high. To reduce this buffer requirement, FDLs are required to be shared among inputs. An all-optical shared-buffer (or, shared-FDL) switch was proposed in (M. J. Karol, 1993). Fig. 1 depicts an example of a 4×4 switch with 4 shared FDLs, where the size of a timeslot is normalized to be 1 and the delay values

of the FDLs are 1, 1, 2, and 4, respectively. These FDLs, subject to availability, can be used by any of the inputs to buffer packets. Moreover, packets are also allowed to re-circulate within these FDLs when necessary so that flexible delay values can be achieved.

<Insert: Fig. 1 A 4×4 shared-buffer switch with 4 FDLs>

In (M. J. Karol, 1993), Karol further proposed a simple FDL assignment algorithm for buffer management. The proposed algorithm assigns FDLs for packets based on a set of simple criteria without output-port stipulation. As a result, when the buffered packets exit from the FDLs, there is no guarantee that any output ports will be available, and they may need to face another round of contention. This may lead packets to experience unbounded delay.

In contrast, the use of the sequential FDL assignment (SEFA) algorithm as proposed in (S. Y. Liew, 2005) can schedule the whole journey for a delayed packet, from input, through the assigned FDL route, to output. The resources along the path will be reserved for the packet once the schedule is established, and those packets that cannot be assigned an FDL route are discarded to save the available resources for further input packets. Although SEFA can achieve better performance, it suffers from a time and overhead complexity as it sequentially searches for a valid FDL assignment.

In this paper, we propose two fast scheduling algorithms for all-optical shared-FDL packet switches. In the first genetic algorithm, packet scheduling is first formulated as a tree-searching problem. By breaking down the search tree into multiple smaller subsets and assigning each subset to a secondary processor, solutions can be obtained faster since the secondary processors are working in parallel. As a case study we considered a 32×32 switch with 32 FDLs, and assumed a processor clock rate of 200MHz. With our first algorithm, a scheduling assignment is done for a packet in 25ns if 64 parallel processors are used, or 55ns if 8 parallel processors are used. Through simulation, we also show that our approach can achieve a loss rate smaller than 10^{-7} even at load 0.9 (i.e. 90%), where the average delay is around 12 timeslots. (A timeslot actually is time taken for a processor clock).

Using the second genetic algorithm, the search tree is a smaller size by trimming it in such a way that the total time/space complexity of the algorithm can be reduced. The process is to skip the searching process for some valid but not favorable solutions in order to compensate for the inherent complexity of reaching a conclusion. Simulation shows that comparable performance to the first algorithm can still be achieved if the search tree is carefully reduced. Consider the same switch parameters as mentioned above, but now the searching is done on a trimmed search tree. A scheduling assignment can be done for a packet in 25ns if 24 parallel processors are used, or 30ns if 8 parallel processors are used, yet the loss rate is kept at the order of 10^{-7}, and average delay is around 12 timeslots, at loading of 0.9 (i.e. 90% rate).

The remainder of this paper is organized as follows: Section 2 studies the scheduling model in an all-optical shared-FDL packet switch and using the sequential search algorithm for scheduling assignment. In section 3, we propose using a fast scheduling algorithm, our first algorithm, which employs multiple parallel processors for computing the scheduling assignment for incoming packets. Section 4 further enhances the first scheduling algorithm by running the parallel search over a reduced search tree, the second algorithm. We show by simulation that comparable performance can be achieved with much lower time/space complexity with this second approach. Finally, section 5 gives the conclusion.

2. The Scheduling Model

We focus on the fixed-length all-optical packet switching in this paper, and the terms “packet” and “cell” are used interchangeably unless otherwise specified.

In general, a packet switch needs to maintain a configuration table which indicates the connectivity between its inlets (inputs and FDL exit feedback ports) and outlets (outputs and FDL entrance ports) in every timeslot. As connectivity information is given, the availability information can also be known from this configuration table. For example, using fig 2, assume that the scheduled connectivity of a shared-FDL switch in timeslot t is a given, where the outlets of FDLs 2, 3, 4, and outputs 2, 3 have been scheduled to be connected to some inlets 1, 2, and 3, and thus they are no longer available at time t. With reference to Fig. 3, the above connectivity information can be represented by the shaded entries in column t of the configuration table, where the other blank entries indicate that all other outlets are still available. It should be noted that such a table can be further expanded horizontally to other columns, each representing a future timeslot after t.

<Insert: Fig. 2 A 4×4 shared-buffer switch with existing connections in t.>

<Insert: Fig. 3 The availability information in the configuration table.>

Based on such availability information from the configuration table, scheduling assignments can then be evaluated, if needed, for the subsequent packets. For instance, with the above existing configuration, suppose there is another packet arriving from input 4 at timeslot t, and it is destined for output 3. As output 3 in timeslot t is not available, buffering this packet is necessary using an FDL.

To perform the FDL assignment for this packet, a tree can be derived from the configuration table, as shown in Fig. 4. In such a tree, each node is used to denote a column (timeslot) in the configuration table. A node is the parent of another if there is a FDL able to transfer a packet from the parent to the child. As different FDLs may have the same delay value,

there can be multiple arcs from a parent node to a child node, each representing a specific FDL.

<Insert: Fig. 4 The top-down search tree for the scheduling assignment.>

The number of children that a parent can have is equal to the number of different FDL delay values, as seen from the given example; each parent has exactly three children because there are three different FDL delay values, these are 1, 2, and 4. To show the availability of FDL information for those timeslots in which the output in question (output 3) is unavailable, the nodes are marked as shaded nodes and those in which the output is available, these nodes are unmarked, and the available FDL solid vectors and unavailable FDL dashed vectors are also used within this model. In a sequential scheduling algorithm, the FDL assignment is conducted through a breadth-first search, traversing the nodes via only the solid vectors in a top-down approach, searching for a blank node to allow the packet to exit. If there are two or more valid solutions at the same depth, the one with the least t delay will be chosen. In the event, where there is no valid solution identified even after the scheduler has traversed the maximum allowable depth, the packet is discarded before entering the switch. In the example given in Fig. 3, the valid solution with the least t delay value is Node 6 (the node $t + 3$ at depth 2), which makes use of FDL1 and FDL3 as the transit path from the root node. The path is then assigned to the packet, and the availability table is then updated.

These relationships can be described as follows:

Let $\check{Z}$ denote the number of distinct delay values that FDLs can have (the number of children of the parent). The time complexity of the sequential-search algorithm is thus given by $\check{Z}^0 + \check{Z}^1 + \cdots + \check{Z}^d$, or $O(d\check{Z}^d)$, where d is the maximum depth allowed in the algorithm.

3. The fast FDL Assignment algorithm over the Full Search Tree

It can be seen that using the sequential search for all-optical packet scheduling is very time consuming. For instance, if we consider a 32×32 switch where a time slot size of 1μ s is adopted, a scheduling assignment for a packet needs to be done within 31.25ns. In order to speedup the searching process, we propose using a fast algorithm which employs multiple parallel processors for computing the scheduling assignment. In the fast algorithm, the same search tree will be used but the algorithm adopts a bottom-up approach which starts from all the nodes at the maximum allowable depth, as shown in Fig. 5.

<Insert: Fig. 5 The bottom-up approach for the fast algorithm>

To start this process, each node at the maximum depth will be assigned to a processor, which will simultaneously work with other processors at the same depth. The processors will first check whether they are blank nodes (i.e., the output process is achievable using the available timeslots). If a node turns out to be a blank node and the vector connected from the parent to the node is solid (i.e., the FDL is available), the node will pass such information to its parent. Each parent will then pick the best potential solution among itself and its children, and pass it to the grandparent for further comparison and selection. However, if a node itself is a valid solution, it will always pick itself as the best solution and abandon those proposed from its children. This process continues until the root node is reached, where the global best solution will be picked as the final scheduling assignment.

Since all the nodes of the same depth work in parallel, the number of steps required for comparing and choosing is $d + 1$, where the number of steps required for passing information is d. The time complexity of the algorithm is therefore given by $2d + 1$, or O(d).

The price of this approach is the quantity of parallel processors required, which is $\check{Z}^d$. For example, if $\check{Z}$=8 and d=2, a total of 64 processors are required. If we further assume that the processors have a clock rate of 200MHz, then each processor takes 5ns to execute an operational step, And this means the total time required for calculating a scheduling assignment is $(2 \times 2 + 1) \times 5ns = 25ns$.

In order to achieve a balance between the time complexity and the number of processors required, the algorithm can start at any depth.

Consider that the algorithm starts at depth i, where $0 \leq i \leq d$. In this case, a total of $\check{Z}^i$ processors are needed and each is assigned to a depth-i node.

With reference to Fig. 6, in the first part of the algorithm, each of these processors will do the top-down sequential search for a best local solution, using the corresponding depth-i node. It takes $\check{Z}^0 + \check{Z}^1 + \cdots + \check{Z}^{d-i}$ steps for a processor to explore all its descendants. Next, the depth-i nodes transmit the local best solutions to their parents for further processing leading to a solution selection. After this selection, the solutions are recursively passed up until the global best solution is selected at the root node. This takes 2i steps to complete. Thus, the entire algorithm has a time complexity of $O((d - i)\check{Z}^{d-i} + i)$. For example, if $\check{Z}$=8, d=2, and i=1, the total number of processors required is 8, and it takes 1+8 steps for a sequential search, that is, one step for data passing, and one step of final comparison at the root node, making it a total of 11 steps to accomplish the task. Assuming each step takes 5ns, then, even in the worst case, the total time needed to calculate an FDL assignment for a packet is 55ns. This is still much faster than SEFA proposed in (M. J. Karol, 1993), which takes

145ns in the worst case.

<Insert: Fig. 6 The searching process that starts from depth i.>

Regardless of at which depth (i value) the algorithm begins, and providing the selection criteria are the same, both FDL assignment algorithms yield the same results. This has been verified by simulation.

However, such verifications are not shown in this paper as the performance curves are all identical.

From the following charts, we show only the simulation results by taking i=1 for all cases. In our simulation model, we considered a 32×32 FDL switch with 32 FDLs. The delay values of FDLs are distributed as evenly as possible among 1, 2, 4, 8, to F, where F is the maximum FDL delay value. We further consider cases whereby F = 32, 64, 128, 256, and 512, respectively. It should also be noted that $\check{Z} = log_2 F + 1$. For example, when F=128, there are $\check{Z}$=8 distinct FDL delay values, and each value has 4 FDLs noting that this also results in a total FDL length (buffer size) of 1020.

We assume a Bernoulli packet arrival process at each input, and each packet is equally likely to be destined for any of the output. We also assume d = 2 for the simulation given in Fig. 7 and Fig. 8. As can be observed in Fig. 7, the larger the F, the better the performance of the switch. This is because a larger F means more memory, thus the overall FDLs are capable of storing more packets to avoid loss. However, it can also be observed from Fig. 8 that the larger F also implies larger average packet delay. In particular, when F = 128 and at load 0.9, a packet loss rate $< 10^{-7}$ can be achieved, where the average packet delay is 11.5ns. It should also be noted that the delay bound can be given by $F \times d$.

<Insert: Fig. 7 Packet loss rate for d=2, where F=32, 64, 128, 256, and 512>

<Insert: Fig. 8 Average packet delay for d=2, where F=32, 64, 128, and 256>

Fig. 9 shows an interesting simulation result, where packet loss rates are almost identical for the cases of d = 2, 3, and 4, respectively. The result is that even if we limit the maximum re-circulation times of packets in the FDLs to be twice only, the optimum performance of the switch can be achieved. Comparing the performance of our algorithms with SEFA given in (M. J. Karol, 1993), they are again almost the same.

<Insert: Fig. 9: Simulation results for F=128, where d=1, 2, 3, and 4.>

4. The Fast FDL assignment algorithm over the reduced search tree

Although the algorithm discussed in section 3 can achieve a better overall performance than all known existing scheduling algorithms, it may still not be enticing enough as the time/space complexity may still be too high for all-optical switches. In this section, we introduce an enhanced version of the fast algorithm which can achieve an even lower time/space complexity. The process is to trim the full search tree given in the above algorithm by removing some valid but unfavorable solutions, this, to reduce the overall complexity of searching.

Using the example given in Fig. 4, we can restrict FDL4 (which is with a delay value of 4) in such a way that it can only be used by the root node, but not all other nodes at higher depth. This restriction yields a reduced search tree shown in Fig. 10, where T is defined as the set of FDL values that can be used by the non-root nodes.

Let $\check{T}$ be the number of elements in T, that is, $\check{T} = |T|$. The search tree is modified in such a way that only the root node has $\check{Z}$ children, and all the nodes at higher depth can only have $\check{T}$ children or none at all.

With the reduced search tree, we can apply the same top-down, bottom-up, or a hybrid approach to search for a valid solution to the scheduling problem, resulting in a lower overall complexity. Consider a maximum depth d = 2, using only a bottom-up approach, a total of $\check{Z} \times \check{T}$ parallel processors are required and it takes the same 5 steps to find a scheduling assignment for a packet. If the algorithm starts from depth 1, then a total of $\check{Z}$ parallel processors are needed and it takes $\check{T} + 1 + 1 + 1 = \check{T} + 3$ steps to accomplish the assignment task.

<Insert: Fig. 10 Reduced search tree where T = {1, 2} >

It should also be noted that the elements in T may be of any FDL delay values in the switch preferring those with low delay values, this preference is because these impose lower delay values on packets when re-circulations are involved. Furthermore, $\check{T}$ is flexible enough to accept a value range of $1 \leq \check{T} \leq \check{Z}$. However, setting $\check{T}$ too high increases the algorithm's complexity, while setting $\check{T}$ too low increases the algorithm's packet loss rate, and neither are considered good design in an all-optical switch.

For the sake of simplicity, within the following discussion, T will contain the first $\check{T}$ delay values of the FDLs in the switch. For example, $\check{T} = 2$ implies that $T = \{1, 2\}$ is adopted, while $\check{T} = 3$ implies that $T = \{1, 2, 4\}$ is adopted, and so on, and also that $\check{T} = \check{Z}$ and $T = \{1, 2, 4, \cdots, F\}$ denotes that case discussed in section 3, where the full set of FDLs can be used by any of the parent nodes.

Simulations were conducted under the same environment as the one in Section 3 that is, using a 32×32 switch with 32 FDLs. The simulation results are shown in Fig. 11 to Fig. 14. From the results in Fig. 11, where we take F = 128 and $\check{Z}$ = 8 as a benchmark, concluding that a larger $\check{T}$ in general has a better loss performance than those with smaller values. At

higher offered load, their performances, however, are quite the same as the switch has reached a saturated state. Fig. 12 shows its corresponding average delays. Here, it can be observed that the average delays are almost the same for all loads across all values of $\breve{T}$.

<Insert: Fig. 11 Packet loss rates for F = 128 and d=2 with different sets of $\breve{>}$

<Insert: Fig. 12 Average delays for F = 128 and d=2 with different sets of $\breve{T}$ >

In Fig. 13, we take $T = \{1, 2, 4\}$(*i.e.*, $\breve{T} = 3$) as an example to study the performance of the switch with different settings of F. Compared to the result of Fig. 12 with that of Fig. 7, the performance is slightly worse, but still acceptable. In particular, when F = 128, the packet loss rate is 2.78×10^{-7} at a load 0.9. It should be noted that, though the loss performance is slightly degraded, the complexity of the algorithm is much improved. For instance, consider $F = 128(\breve{Z} = 8)$ and $\breve{T} = 3$, and a processor clock rate of 200MHz. The time needed for computing a scheduling assignment for a packet is $5 \times 5ns = 25ns$ if $8 \times 3 = 24$ parallel processors are used and $(3 + 3) \times 5ns = 30ns$ if 8 parallel processors are used. Fig. 14 shows the average delays for its corresponding packet loss rates in Fig. 13.

Similar to the algorithm in the previous section, the average delay packet for F = 128 at load 0.9 is 11.5 and the packet loss rate is still below 10^{-6}.

<Insert: Fig. 13 Packet loss rates with different values of F where $T = \{1, 2, 4\}$ and $d = 2$ >

<Insert: Fig. 14 Average delays with different values of F. where $T = \{1, 2, 4\}$ and $d = 2$ >

Table 1 summarizes the performance of the switch in terms of packet loss rate at offered load 0.9, considering F = 128. It can be seen that simulations involving various values of $\breve{T}$ and d can still be able to yield packet loss rates as low as $\sim 10^{-7}$.

<Insert: TABLE 1: Packet loss rates at load 0.9 with various values of $\breve{T}$ and d, for $F = 128$. >

5. Conclusion

In this paper, we have proposed two fast scheduling algorithms for all-optical shared-FDL packet switches. Our algorithms aim to solve the formulated search tree in a parallel manner where multiple processors working in parallel are employed to reduce the time-complexity problem. Compared with other existing algorithms, our two approaches possess many apparent advantages, such as high throughput, low loss rate, low delay, and low time complexity. We found that our algorithms may calculate a scheduling assignment for a packet in as low as 30ns with 8 parallel processors, assuming a processor clock rate of 200MHz. Through simulation, we have also showed that our algorithm can achieve a loss rate of $\breve{1}0^{-7}$ even at load 0.9 for a 32×32 switch with 32 FDLs.

References

E. Wong and M. Zukerman. (2006). *Bandwidth and Buffer Tradeoffs in Optical Packet Switching in Proc*. J. Lightwave Technol., vol. 24, no. 12, pp. 4790-4798.

J. Li. (2007). *Maximizing Throughput for Optical Burst Switching Networks* in IEEE/ACM Transactions on Networking, vol. 15, no. 5, pp. 1163-1176.

J. Ramamirtham and J. Turner. (2003). *Time sliced optical burst switching*, in Proc. IEEE INFOCOM 2003, San Francisco, United States.

M. J. Karol. (1993). *Shared-memory optical cell (ATM) switch in Proc*. SPIE, Multigigabit Fiber Communications Systems, vol. 2024, Jul. 1993.

S. Y. Liew, G. Hu, and H. J. Chao. (2005). *Scheduling Algorithms for Shared Fiber-Delay-Line Optical Cell Switches*-Part I: The Single-Stage Case in Proc. J. Lightw. Technol., vol. 23, no. 4, pp. 1586-1600

Z. Zhang, Y. Yang. (2006). *WDM Optical Interconnects with Recirculating Buffering and Limited Rage Wavelength Conversion* in IEEE Transactions on Parallel and Distributed Systems, vol. 17, no. 5, pp. 466-480.

Table 1. Packet loss rates at load 0.9 with various values of $\breve{T}$ and d, for F=128

$\breve{T}$	d	Packet Loss Rate at load 0.9
2	2	9.728E-07
2	3	2.08E-07
3	2	2.78E-07
3	3	< 1.0E-07
4	2	1.39E-07
4	3	< 1.0E-07
8	2	< 1.0E-07
8	3	< 1.0E-07

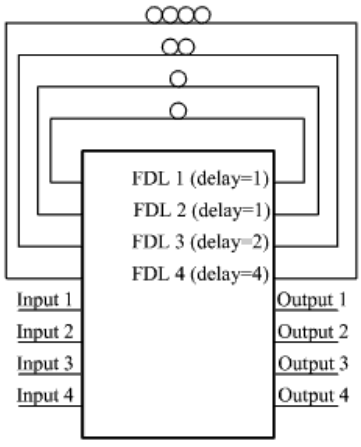

Figure 1. A 4×4 shared-buffer switch with 4 FDLs

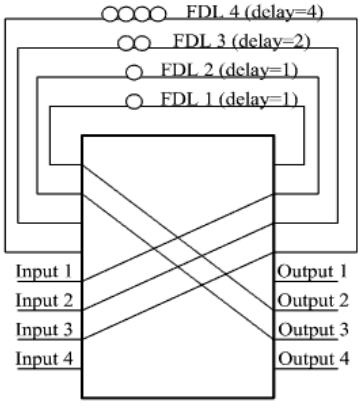

Figure 2. A 4×4 shared-buffer switch with existing connections in t.

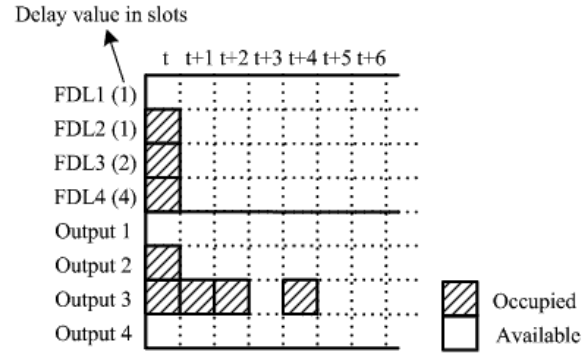

Figure 3. The availability information in the configuration table

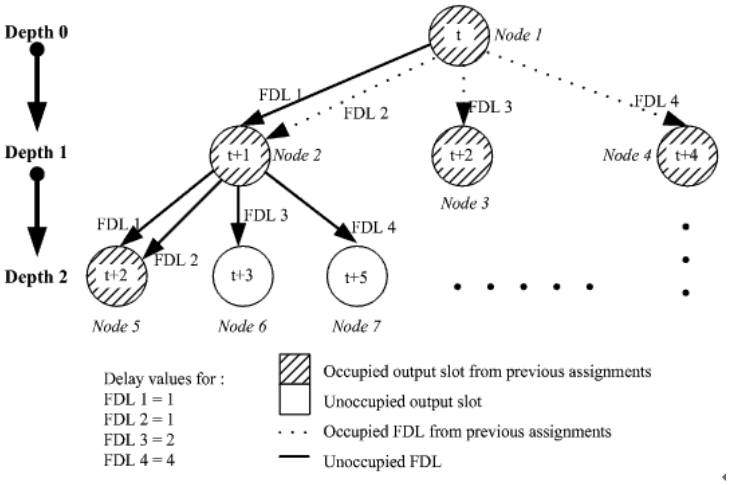

Figure 4. The top-down search tree for the scheduling assignment

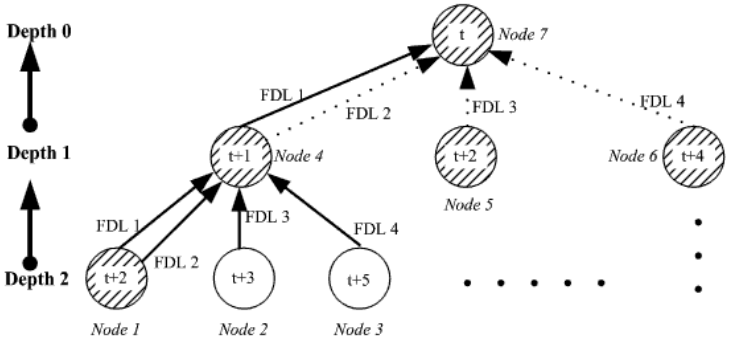

Figure 5. The bottom-up approach for the fast algorithm

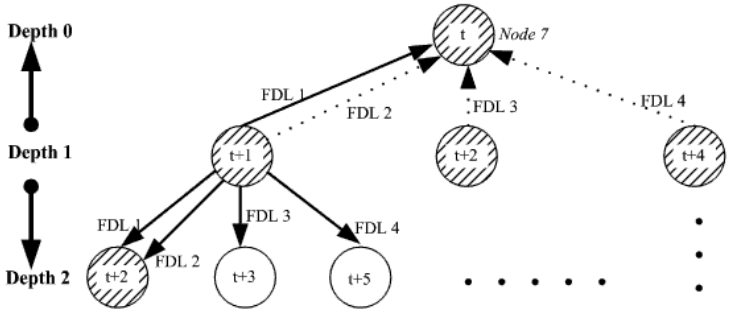

Figure 6. The searching process that starts from depth i.

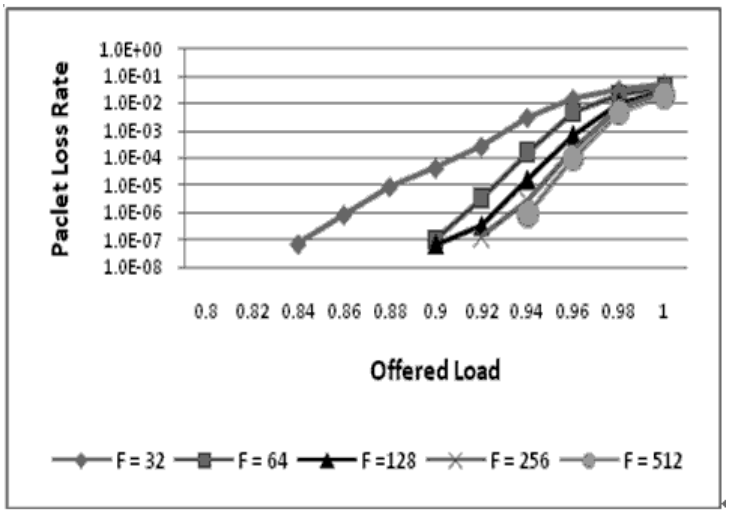

Figure 7. Packet loss rate for d=2, where F=32, 64, 128, 256, and 512

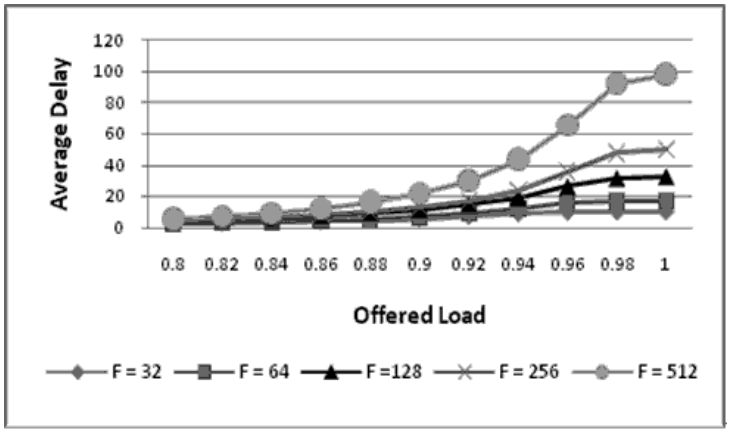

Figure 8. Average packet delay for d=2, where F=32, 64, 128, and 256

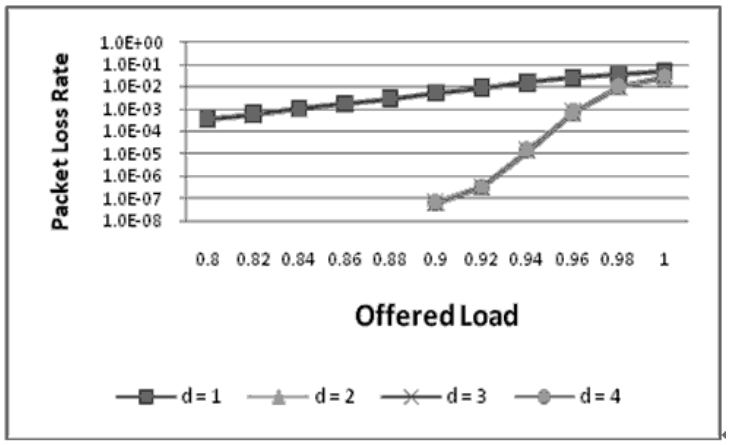

Figure 9. Simulation results for F=128, where d=1, 2, 3, and 4

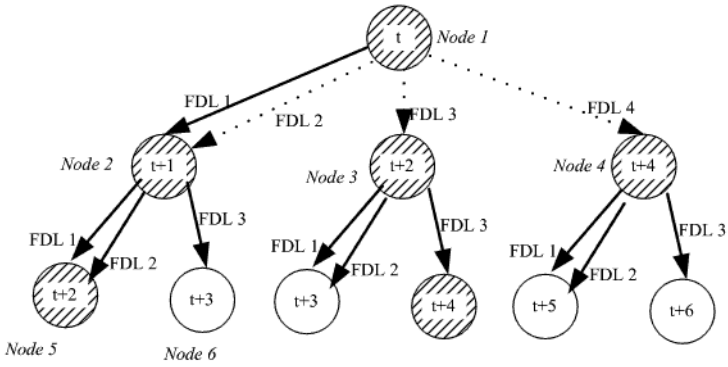

Figure 10. Reduced search tree where T = {1, 2}

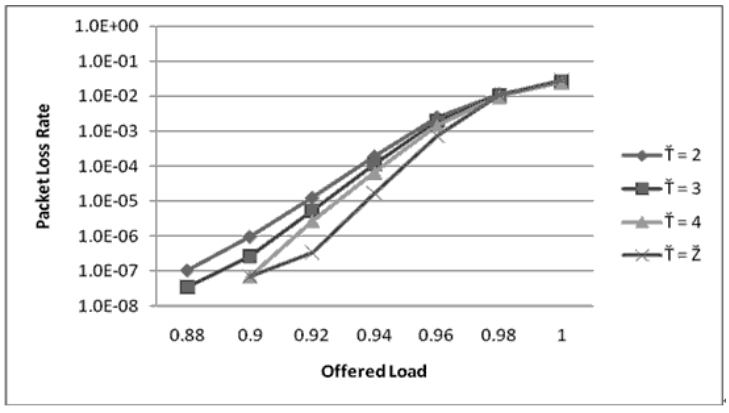

Figure 11. Packet loss rates for F = 128 and d=2 with different sets of $\check{T}$

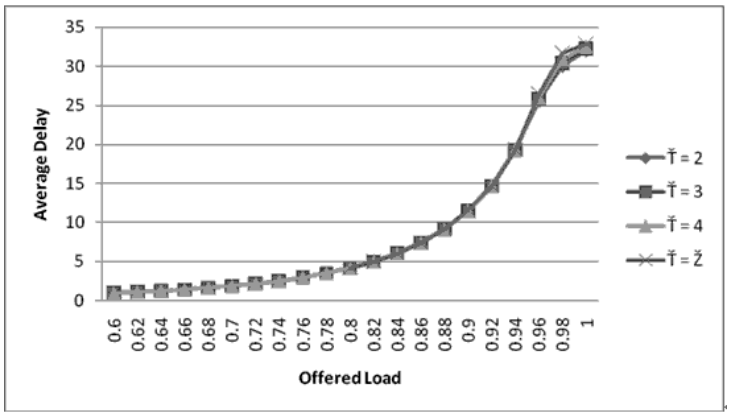

Figure 12. Average delays for F = 128 and d=2 with different sets of $\check{T}$

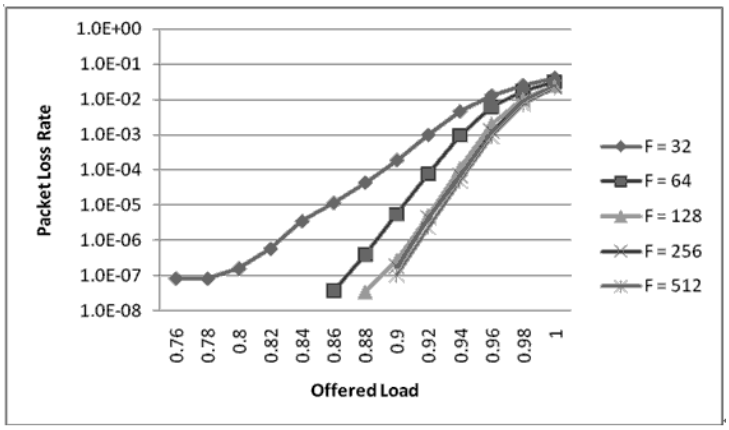

Figure 13. Packet loss rates with different values of F where T = {1, 2, 4} and d=2

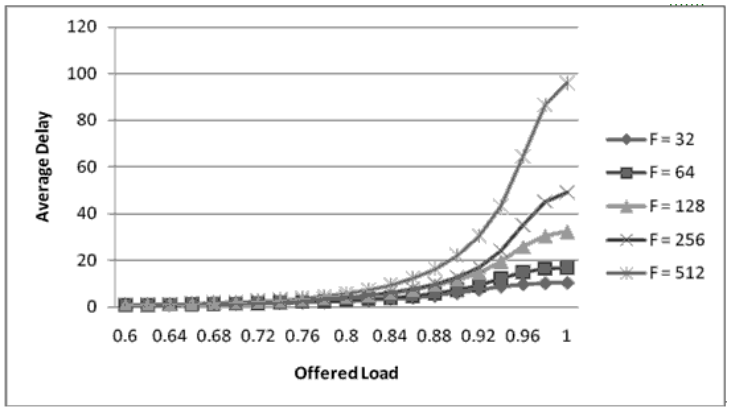

Figure 14. Average delays with different values of F. where T = {1, 2, 4} and d=2