Time Slot Allocation for Real-Time Messages with Negotiable Distance Constraints*

Libin Dong, Rami Melhem, Daniel Mossé
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
(dong, melhem, mosse)@cs.pitt.edu

Abstract

In this paper we present a time slot allocation scheme to support real-time communications with strict rate requirements and flexible distance constraints. We first devise an algorithm to schedule message streams on a single link. We then extend it to schedule real-time traffic on WDM optical star couplers, where the validation requirements of the device configuration are also taken into consideration in the scheduling algorithm. The results show that by decoupling the rate and distance constraint requirements of the real-time streams, high schedulability can be achieved with a relatively small jitter.

1. Introduction

There has been an increased need for real-time computing and communication services in current applications, such as remote video display in multimedia conference. Predictable and guaranteed service has become one of the critical components of the *quality-of-service* (QoS) requirements. Real-time messages are to be sent and received within specified timing constraints. For example, in periodic message models, an isochronous message stream M_i generates c_i message frames in a certain period P_i. In the k^{th} period, the ready time for the k^{th} instance of M_i is $(k-1) \times P_i$ and the deadline of that instance is $k \times P_i$. In order to ensure that each frame meets its deadline, the most common solution is to schedule the transmission of messages. In this paper, we consider dividing time into slots of equal length, each large enough to transmit a message frame, and we schedule the real-time traffic within a *template*, which contains an integer number of time slots. The allocation pattern of the template can be applied repeatedly to control the on-line transmission of messages.

For the periodic scheduling problem, several optimal single resource scheduling algorithms, such as *rate-monotonic-scheduling* (RMS) and the *earliest-deadline-first* (EDF) algorithm [8], can be applied to schedule real-time traffic. In some real-time systems, periodic tasks or messages must satisfy a timing constraint requirement relative to the finishing time of the previous instance, which is defined in [5] as the *distance constraint* system model. For example, along a network link of bandwidth B, an isochronous video stream may require to transmit data at a rate of $R \times B$, where R is specified as a proportion of the total link bandwidth. So, the average distance between the transmission of consecutive frames must be $\lfloor 1/R \rfloor$ time slots. But in order to maintain the human perception condition of the video, the time interval between any two consecutive video frames must not exceed some maximum value, say, D time slots, which is taken as the distance constraint requirement of the message stream. As pointed out in [5], the scheduling algorithms for the periodic model, EDF or RMS, are not applicable to the distance constraint model.

The distance constraint scheduling problem is closely related to the *pinwheel problem* which is formally defined as follows [2, 6]: Given a multi-set of n positive integers $A = \{a_1, ..., a_n\}$ ($a_1 \leq ... \leq a_n$), the problem is to find an infinite sequence (schedule) of symbols from {1,..,n} such that there is at least one symbol i within any interval of a_i slots. In [2, 6], the *density* of A is defined as $\rho(A) = \sum_{i=0}^{n} \frac{1}{a_i}$ and several schedules are devised for the pinwheel problem. To guarantee a feasible schedule for a pinwheel problem, the schedulability condition derived in [6, 2] is $\rho(A) \leq \rho_{max}$, where ρ_{max} is equal to 1/2, 13/20, 2/3, 0.696, or 0.7 depending on a specific specialization operation whose objective is to transform A into another set $\bar{A} = \{\bar{a_1}, ..., \bar{a_n}\}$ such that $\bar{a_i} \leq a_i < 2\bar{a_i}$ and for all $i < j$, $\bar{a_i}$ divides $\bar{a_j}$.

*This work was supported in part by DARPA under Contract DABT63-96-C-0044, a part of the FORTS project.

In the pinwheel problem, for a given stream M_i, the average distance between consecutive message frames, $\lfloor 1/R_i \rfloor$, and the maximum distance constraint D_i are considered to be the same. This characterization fails to represent some real-time applications. The following example shows that the scheduler for the pinwheel problem fails to find a schedule even though there exists a feasible schedule with a more relaxed distance constraint requirement decoupled from the rate requirements.

Example 1: Assume three message streams, M_1, M_2 and M_3 requiring transmission rates of $1/2$, $1/3$ and $1/6$, respectively. If this example is translated to a pinwheel problem of three streams with requirement $\{2, 3, 6\}$, all the pinwheel scheduling algorithms fail to find a schedule and will reject the set of messages. However, Figure 1 shows that, by relaxing the distance constraint of M_2 to 4, instead of 3, there exists a feasible schedule with a template of size 6 which satisfies the rate requirement of all three message streams. In Figure 1, $M_{i,j}$ means the j^{th} instance of the message stream M_i. Specifically, the maximum distance between any two consecutive instances is 2 for M_1, 6 for M_3 and 4 for M_2. This can be seen by repeating the template and observing that the distance between $M_{2,2}$ and $M_{2,3}$ is 4. □

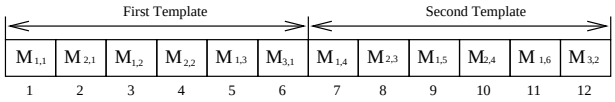

Figure 1. A Schedule for Example 1

In this paper, we propose a pre-allocation based scheme for scheduling n message streams M_1,...,M_n, where each stream M_i has a rate requirement R_i and a maximum distance constraint requirement D_i, such that $D_i \geq 1/R_i$. We assume that the rate requirement is the critical QoS requirement that cannot be violated, while the distance constraint requirement is negotiable. The problem is to find a schedule that satisfies both the rate and the distance constraint requirements of all message streams. First, a minimum template size is chosen according to the rate requirements of all message streams that are to be scheduled. Then a time slot allocation pattern for the template is sought such that for every message stream, M_i, the obtained rate is greater than or equal to R_i and the maximum distance between any two consecutive frames of M_i is not larger than D_i. If no such schedule can be obtained, then the scheduler may negotiate with the application for a more relaxed distance constraint condition and may attempt to schedule the stream again. The system rejects the message stream if negotiation fails.

2 Allocation Scheme for a Single Link

In this section, we consider the problem of assigning time slots to a set of message streams on a single link in a point-to-point network. This scheme can also be applied to scheduling the backbone bus on a broadcast network such as an Ethernet or a DQDB network [4].

Assume that there exists a message set $\mathcal{M}$ of n isochronous message streams , M_1, ..., M_n, to be sent along a link. Time is divided into time slots, where each slot is long enough for the transmission of one message frame. Each time slot should be allocated to a unique message stream. Every isochronous message stream is represented as $M_i = \{A_i, D_i\}$, where both A_i and D_i are integers representing numbers of time slots and $A_i \leq D_i$. A_i specifies the average distance between consecutive frames of M_i. D_i is the maximum distance constraint between any two consecutive frames.

Define the *density* of the stream M_i as $\rho(M_i) = 1/A_i$ and the *total density factor* of the message set $\mathcal{M}$ as $\rho(\mathcal{M}) = \sum_{i=0}^{n} \frac{1}{A_i}$. In order to get a feasible schedule for the link, the total density factor of the single link cannot exceed 100%. That is, $\rho(\mathcal{M}) = \sum_{i=0}^{n} \frac{1}{A_i} \leq 1$.

2.1 Calculating the Size of the Schedule Template

The algorithm presented in the next section is a pre-allocation based scheme which schedules all the time slots in a template to message streams. Our objective is to find the minimum template size which can satisfy the rate requirement of all message streams. A small template size decreases the time complexity of the scheduling algorithm and simplifies the on-line transmission control.

Assume the template size is N. In order to fulfill the rate requirement A_i of message stream M_i, $\lceil N/A_i \rceil$ time slots have to be allocated for M_i within a template of N slots. Given that the total utilization of the link should not exceed 1, a feasible schedule for all the messages in the template can be obtained only if

$$\sum_{i=0}^{n} \lceil \frac{N}{A_i} \rceil \leq N \tag{1}$$

If N is equal to the least common multiple of $A_1, A_2, ..., A_n$, which we will simply call LCM, then $\frac{\sum_{i=0}^{n} \lceil \frac{N}{A_i} \rceil}{N} = \sum_{i=0}^{n} \frac{1}{A_i}$. Therefore, $\rho(\mathcal{M}) \leq 1$ guarantees that there exists at least one value of N that satisfies condition (1). It is quite possible, however, that the value of LCM is very large and there is a smaller N which also satisfies (1). A *fixed point* scheme [7, 1] can be used to iteratively find the minimum N that satisfies (1). Specifically, starting from an initial estimate N^0, we can find successive

estimates for N from

$$N^{k+1} = \sum_{i=0}^{n} \lceil \frac{N^k}{A_i} \rceil \quad (2)$$

The initial estimate N^0 can be set to n. The right side of equation (2) is monotonically non-decreasing in N, in the sense that $N^{k+1} \geq N^k$. The iteration will stop at the first point where $N^{k+1} = N^k$, which makes the total utilization of the link equal to 1. The iteration is guaranteed to converge at the first point satisfying condition (1) if $\rho(\mathcal{M}) \leq 1$ [7, 1].

Example 2: A message stream set $\mathcal{M} = \{M_1, ..., M_5\}$ needs to be scheduled on a single link, where $M_1 = (4, 4), M_2 = (5, 6), M_3 = (6, 6), M_4 = (7, 7), M_5 = (10, 10)$. The total density of $\mathcal{M}$ is 0.86. Using the fixed point scheme described above, we begin at $N^0 = 5$ and $N^1 = \sum_{i=1}^{5} \lceil \frac{5}{A_i} \rceil = 6$. Repeating the computation, we obtain $N = N^5 = 10$. Note that the template size for this problem is 10 and the LCM of $A_i (1 \leq i \leq 5)$ is 420. □

2.2 Scheduling Message Streams On a Single Link

In this section, a time slot allocation algorithm is proposed. The scheme is applied repeatedly to allocate each time slot in the template to the next instance of a message stream according to the priority of the streams.

Our allocation algorithm is similar to EDF, in that the highest priority is assigned to the stream with the earliest deadline. The difference is that in the EDF scheme, all instances have fixed ready times and deadlines, while in our algorithm, the deadline and ready time of each instance are dynamically calculated based on the allocation of the previous instances and the distance constraint requirement. The deadline of the next instance is calculated in such a way that the distance between the current instance and the next instance does not exceed the distance constraint, which is called the distance constraint concern in the *forward* direction. Because the allocation pattern of the template repeats continuously, the distance between the first instance in the current template and the last instance in the previous template should also be constrained, which is called the distance constraint concern in the *backward* direction. The ready time of each instance is set such that the backward distance constraint can be satisfied.

We define the *active message set* at a time slot as the set of messages whose ready times are smaller than or equal to the current time. The pseudo-code of the template scheduling algorithm **Tpl_Sched** is presented in Figure 2. The objective of the algorithm is to allocate $\lceil N/A_i \rceil$ time slots to every stream M_i within the template such that the distance constraint requirement D_i is satisfied. Specifically, at each time slot, **Tpl_Sched** allocates the slot to the message stream with the earliest deadline in the active message

```
1.  calculate template size N using fixed point method;
2.  for i = 1 to n do {        /* initialization */
3.     distance_i = A_i;
4.     ready_i = 0;
5.     deadline_i = distance_i;
6.     num_instance_i = ⌈N/A_i⌉;
7.  }
8.  for s = 1 to N do {        /*allocate slots in template */
9.     if there are active messages {
10.       choose an active stream M_u with earliest deadline;
11.       if s > deadline_u  /* M_u misses the deadline */
12.          distance_u ++;
13.    }
14.    else {                  /* no stream is ready */
15.       choose a stream M_u to relax distance constraint;
16.       distance_u += ready_u − s;
17.    }
18.    if distance_u > D_u {/* relaxed too much */
19.       if negotiate and get a D'_u ≥ distance_u
20.          D_u = D'_u;
21.       else fails to find a schedule, exit.
22.    }
23.    allocate current slot s to stream M_u;
24.    record first_u if s is the first slot allocated to M_u;
25.    num_instance_u = num_instance_u − 1;
       /*compute next instance's ready time & deadline */
26.    ready_u = N + first_u −
              num_instance_u × distance_u;
27.    deadline_u = s + distance_u;
28. } /* for */
```

Figure 2. The Algorithm Tpl_Sched

set, and dynamically computes the deadline of the next instance in the stream.

In the algorithm shown in Figure 2, after calculating the minimum appropriate template size N (line 1), **Tpl_Sched** initializes several variables for each message stream M_i (line 2 to line 7). The variable $distance_i$ is the dynamic distance constraint condition whose value is in the range of $[A_i, D_i]$. Because $D_i \geq A_i$, if we directly set $distance_i = D_i$, it is quite possible that the distance between consecutive instances of M_i may vary greatly. In order to prevent this situation and keep the maximum distance as close to the average distance A_i as possible, we first initialize $distance_i = A_i$ (line 3), then increase $distance_i$ only when the algorithm cannot continue the scheduling process and $distance_i$ has not reached D_i (line 18, 19).

The algorithm initializes the ready time, $ready_i$, and the deadline for the first instance of message stream M_i, $deadline_i$. The variable $num\_instance_i$ denotes the num-

ber of instances that still need to be allocated to M_i from the current time slot to the end of the template. So it is initialized to the total number of instances of M_i in the template according to the rate requirement A_i (line 6).

From line 8 to line 28, **Tpl_Sched** repeats the allocation scheme for every time slot within the template. First, from the set of active streams it chooses an appropriate message, M_u, with the highest priority (line 10). If several streams have the same deadline, the tie is broken by assigning the highest priority to the stream M_u with the largest value of $\frac{distance_u}{D_u}$, which means that M_u has the least ability to further relax $distance_u$. The algorithm allocates the current time slot s to M_u (line 23) and decreases $num\_instance_u$ indicating that one more instance of M_u has been scheduled (line 25). Then the deadline for the next instance of M_u is calculated to be $distance_u$ slots away from the current completion time (line 27), such that the distance constraint in the forward direction is obeyed. The ready time is adjusted according to the distance constraint in the backward direction. Assume that the total number of instances of M_u is k in a template of size N and recall that $M_{u,j}$ is the j^{th} instance of M_u. If $M_{u,1}$ transmits in slot $first_u$, then $M_{u,k+1}$ occupies slot $N + first_u$. To guarantee the distance constraints, the slots $M_{u,k+1}$ and $M_{u,k}$ must be separated by $distance_u$. Thus, the ready time for $M_{u,k}$ must be no earlier than $N + first_u - distance_u$. For the same reason, $M_{u,k-1}$ must have a ready time no earlier than $N + first_u - 2 \cdot distance_u$, and so on. Line 26 computes the earliest ready time for the next instance when M_u still has $num\_instance_u$ instances left in the template.

The algorithm may fail to schedule an instance $M_{u,i}$ in two cases. In the first case, the message which is chosen to be scheduled at the current slot misses its deadline (line 11 - line 13). In this case, $distance_u$ is increased by one to extend the deadline (line 12). In the second case, the active message set may be empty because every stream has a ready time later than the current time slot (line 14 - line 17). To solve this problem, the stream M_u with the smallest value of $\frac{distance_u}{D_u}$ is chosen to relax $distance_u$ (line 15) such that M_u's ready time is exactly the current time (line 16). After increasing $distance_u$, we need to check whether it is relaxed too much that it exceeds D_u. If true, **Tpl_Sched** negotiates for larger D_u (line 18 - line 22). If the negotiation is successful, **Tpl_Sched** will continue to allocate successive slots using the new $distance_u$ value to calculate the ready time and deadline of M_u. Otherwise, the algorithm fails to find a schedule and exits (line 21).

Example 3: Apply **Tpl_Sched** to Example 2 in Section 2.1. Here $\rho(\mathcal{M})$ is 0.86, the template size is 10 and every message stream M_i except for M_2 has a D_i value equal to A_i. If we cast this example into a pinwheel problem, the input should be $A = \{4, 5, 6, 7, 10\}$. No pinwheel algorithm is able to schedule this problem. Figure 3 illustrates

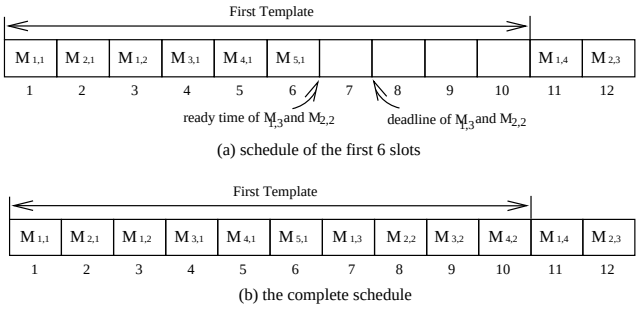

Figure 3. A Scheduling Example

how **Tpl_Sched** obtains a feasible schedule making good use of the specification $D_2 > A_2$. Following the scheduling scheme in **Tpl_Sched**, Figure 3a shows the scheduling of the template up to the sixth time slot. At this point, $distance_1 = A_1 = 4$, $distance_2 = A_2 = 5$, and both M_1 and M_2 need to schedule one more instance. The computation shows that both the ready time and the deadline of the next instance of M_1 are 7. The same results are obtained for the next instance of stream M_2. So, at the beginning of the 7^{th} slot, M_1 and M_2 are ready and both should be scheduled by the end of the 7^{th} slot, which causes a conflict. Note that, at this point, $distance_1 = D_1$ while $distance_2 = D_2 - 1$, which means that M_1 has a bigger value of $\frac{distance}{D}$ than M_2. According to the algorithm, slot 7 is allocated to M_1, which makes M_2 miss its deadline at slot 8. Then $distance_2$ is increased by one to reach D_2 and the problem is solved. Figure 3b displays the final template. □

Since we need to scan the list of streams for each time slot, the time complexity of the algorithm is $\Theta(nN)$ where n is the number of message streams and N is the template size.

3 Performance Results

In this section, the results of applying **Tpl_Sched** to an artificial workload of message streams are presented and contrasted with the results of the best pinwheel scheduling algorithm applied to the same inputs.

We generate sets of message streams $\mathcal{M} = \{M_i = (A_i, D_i) | 1 \leq i \leq n\}$. The rate requirements are generated such that the total density factor, $\rho(\mathcal{M}) = \sum_{i=0}^{n} \frac{1}{A_i}$, is in some range $[\rho_l, \rho_h]$ where ρ_l and ρ_h are the lowest and highest limit. We define the *relative scheduling jitter (RSJ)* of stream M_i as $RSJ_i = (D_i - A_i)/A_i$. RSJ_i is a measurement of the flexibility allowed for scheduling M_i.

In Figure 4, the Y-axis represents the *success rate*, which is the percentage of successfully scheduled stream sets from an input population of 10,000 random stream sets, and the X-axis represents maximum limitation of RSJ for all the

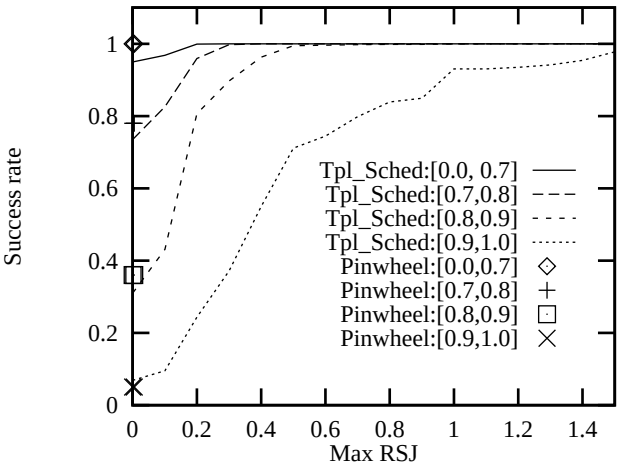

Figure 4. Success Rate vs. RSJ

streams. When $RSJ = 0$, D_i is equal to A_i, which means that this scheduling problem is equivalent to a pinwheel problem. The four points on the vertical line of $RSJ = 0$ represent the success rate of the pinwheel algorithm (the pinwheel only finds solutions for $RSJ = 0$). The figure shows the improvement obtained by **Tpl_Sched** when RSJ is increased. When $\rho \leq 0.7$, the algorithm for the pinwheel problem is guaranteed to generate a feasible schedule [2]. However, when the total density factor increases, the pinwheel algorithm fails to have a high success rate, while the **Tpl_Sched** algorithm can achieve a high success rate by slightly relaxing the distance constraint requirement. For example, when the total density factor is in the range $[0.8, 0.9]$ and $RSJ_i = 0.2$, that is, D_i is 20% more than A_i, the success rate is as high as 80%. In summary, Figure 4 illustrates that, by decoupling the rate requirement and the distance constraint requirement, the **Tpl_Sched** algorithm makes use of the flexibility in D_i to increase schedulability.

Table 1 measures the average RSJ of all the streams when the total density of the stream set is in the following ranges: [0.0, 0.1], [0.1, 0.2], ..., [0.9, 1.0]. For each range, we generate a group of message streams according to the specified total density range and apply **Tpl_Sched** without limiting the maximum D_i value. So, all the streams can relax D_i as much as needed to get a successful schedule. The measured average RSJ indicates the level of flexibility needed for a group of streams to obtain a schedule. From the table, we can see that even when the total density range of the stream set is [0.9, 1.0] the average RSJ is less than 12%, which means that the algorithm keeps the maximum distance between two consecutive frames close to the average distance for most of the streams in the set.

density	[0, 0.7]	[0.7, 0.8]	[0.8, 0.9]	[0.9, 1]
avg RSJ	0.048%	0.624%	2.16%	11.6%

Table 1. Average RSJ v.s. Total Density

4 Algorithm for WDMA Passive Star coupler

In order to further evaluate the performance of our algorithm, we adapt the algorithm presented in the previous section to scheduling passive star couplers in WDMA optical networks and compare its results with those of an algorithm given in [9] for solving the same problem when $D_i = A_i$.

In a wavelength division multiple access (WDMA) network, an optical wavelength represents a transmission channel, and multiple channels can be multiplexed onto a single fiber. Stations may transmit/receive packets on different channels using a tunable laser transmitter/detector. A widely used optical network topology is the passive star topology which uses a broadcast star coupler to transmit messages. In this configuration, every source station uses a tunable transmitter to send messages on a specific wavelength. The star coupler combines the messages from various source stations and broadcasts the mixed optical information to all the destination stations. In order to receive a message from a certain source, a destination station needs a tunable receiver to pick up its messages on the expected wavelength from the wavelength division multiplexed broadcast stream. In the passive star coupler, A valid configuration should satisfy the following two conditions:

1. **No input conflict**: Multiple inputs cannot be switched to the same output simultaneously.
2. **No output conflict**: An input cannot be connected to multiple outputs simultaneously.

In an $a \times b$ star coupler network with a source stations and b destination stations, the number of channels is equal to the number of tunable wavelengths, W, which is usually smaller than b.

We modified the fixed point method to calculate the minimum template size for multiple channels [3]. Then we adapted the algorithm for the single link to the star coupler situation by following the same philosophy of earliest deadline first and the same policies of relaxing and negotiating the distance constraint. Since we need to schedule the messages on W wavelength channels, the algorithm is changed to choose the first W streams without any input or output conflict to fit into a slot [3].

An algorithm named *Binary Splitting* was presented in [9] for scheduling passive star couplers. This is a time slot allocation scheme for time-constraint communications based on the specialization result of the pinwheel problem. The algorithm, however, only applies to sets of streams

whose specialization is successful and whose utilization satisfies certain validation conditions [9]. In other words, a valid set of streams may still be rejected by *Binary Splitting* after passing the validation conditions in [9].

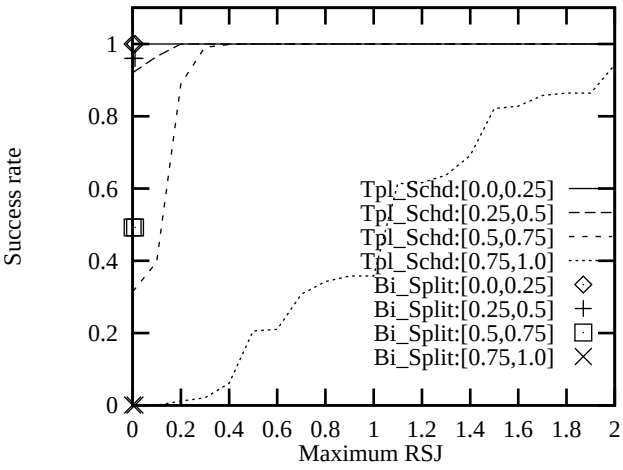

Figure 5. Performance of 8×8 WDMA couplers

Figure 5 presents the performance of our algorithm on an 8×8 star coupler configuration. The four curves correspond to the success rate of four groups of random inputs, each generated based on the specified utilization range for every destination station. Since the number of channels cannot be smaller than the total utilization, we choose the value of W according to the possible highest utilization of each station. The four points on the vertical line of $RSJ = 0$ represent the success rate of the validation test of the *Binary Splitting* algorithm on the same input sets. When the utilization is high, most input sets are rejected before applying *Binary Splitting* because they violate the validation conditions needed to apply the algorithm. The success rate of *Binary Splitting* cannot be higher than the rate shown in Figure 5. Again, Figure 5 illustrates that the policy of relaxing distance constraint improves the success rate, especially when the network utilization is high.

5 Conclusion

In this paper, we consider a real-time message stream model with strict rate requirements and flexible distance constraint requirements. In the model, each stream requires that the average distance between any two consecutive frames of the stream does not exceed a strict predefined constraint. Each stream also has a maximum distance constraint requirement between any two consecutive frames, but this constraint is relaxable and negotiable. The problem is to find a schedule that satisfies the rate and distance constraint requirements of all the streams.

We presented algorithms to schedule message streams on single links, and on WDMA optical star couplers. The algorithm gives higher priority to streams with early deadlines, but differ from "earliest deadline first" algorithms in that the deadlines as well as the ready times of stream instances are modified dynamically according to the distance constraints. Fixed point schemes are used to calculate the minimum size of a scheduling template, according to the rate requirements of the streams. In scheduling star couplers, input and output conflicts also have to be taken into consideration in the scheduling algorithms.

We compared our scheduling algorithms with the pinwheel scheduling which ties the maximum distance constraints to the average transmission rate of the message streams. The results show that higher schedulability is achieved when the rate and distance constraint requirements of the real-time streams are decoupled, especially when the load is high.

References

[1] N. C. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings. Hard real-time scheduling: The deadline-monotonic approach. *Proc. of the 8th IEEE Workshop on Real-Time Operating Systems and Software, Atlanta*, 4 1991.

[2] M. Y. Chan and F. Y. Chin. Schedulers for larger classes of pinwheel instances. *Algorithmica*, 9:425–462, 1993.

[3] L. Dong, R. Melhem, and D. Mossé. Time slot allocation for real-time messages with negotiable distance constraints. *Technical report TR98-07, Department of Computer Science, University of Pittsburgh*, 12 1997.

[4] C. C. Han, C. J. Hou, and K. G. Shin. On slot allocation for time-constrained messages in dual-bus networks. *Proc. of INFOCOM*, pages 1164–1171, 1995.

[5] C. C. Han, K. J. Lin, and C. J. Hou. Distance-constrained scheduling and its applications to real-time systems. *IEEE Trans. on Computers*, 45(7):814–826, 1996.

[6] R. Holte, A. Mok, L. Rosier, I. Tulchinsky, and D. Varvel. A real-time scheduling problem. *Proc. of the 22nd Hawaii International Conference on System Science*, pages 693–702, 4 1989.

[7] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, Oct. 1986.

[8] C. L. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):47–61, 1973.

[9] H. Y. Tyan, C. J. Hou, B. Wang, and C. C. Han. On supporting time-constrained communications in wdma-based star-coupled optical networks. *Proc. of the IEEE Real-time Systems Symposium*, pages 175–184, 12 1996.