Scheduling Algorithms for Dynamic Message Streams with Distance Constraints in TDMA protocol *

Libin Dong, Rami Melhem, Daniel Mossé
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
(dong, melhem, mosse)@cs.pitt.edu
Fax: 1-412-624-5249

Abstract

In many real-time communication applications, predictable and guaranteed timeliness is one of the critical components of the quality of service (QoS) requirements. In this paper, we propose a new real-time message model with both rate requirements and distance constraints. Two algorithms are presented to schedule dynamic real-time message streams in a TDMA frame based on different scheduling policies by making greedy choices or optimization choices. The performance of the two algorithms is evaluated and compared in terms of time complexity, acceptance ratio and scheduling jitter via simulation.

1 Introduction

There has been an increased need for real-time communication services in current applications, such as remote audio or video display in multimedia conferencing, automated control network system, avionics and military communication systems. In these applications, predictable and guaranteed timeliness has become one of the critical components of the *quality-of-service* (QoS) requirements.

Time Division Multiple Access (TDMA) is a widely used protocol, where time is divided into equal length *slots*, each of which is equal to the transmission time of a message packet. A TDMA *frame/template* which contains an integer number of time slots is derived, and the allocation pattern of the template is applied repeatedly to control the transmission of message streams. An isochronous message stream, M_i, continuously generates packets to be transmitted. We use $M_{i,j}$ to denote the transmission of the j^{th} packet of M_i, which is called the j^{th} *instance of* M_i. Each time slot in the template is allocated to transmit an instance of a unique message stream.

Real-time message streams require timeliness guarantees. The average transmission rate is an important requirement. Suppose the size of a TDMA template is N time slots, the average transmission rate decides the number of time slots that should be allocated to the message stream. For example, Link-16 is a radio communication system designed for tactical information distribution systems, and it applies a TDMA protocol [5]. Each TDMA template in Link-16 consists of 512 time slots. A message stream that requires an average transmission rate of $\frac{1}{16}$ of the total bandwidth must occupy 32 time slots in the template.

Another timing constraint that will be addressed in this paper is *distance constraint*, which is defined as the maximum timing interval between two adjacent instances. That is, each instance of an isochronous message stream must complete within a maximum timing constraint relative to the finishing time of the previous instance [7]. The distance constraint problem was originally motivated by the performance requirements of a ground station that processes data transmitted from a number of satellites or mobile sensors [8]. For example, the message stream in an Airborne Warning and Control System (AWACS) recording the position of an aircraft with a velocity of 900 km/hour may be subject to a distance constraint of 400 msecs, in order to ensure that the ground station obtains a positional accuracy of 100 meters [1]. In this paper, distance constraint is represented in terms of time slots. The *distance* between two time slots S_i and S_j is calculated as $distance(S_i, S_j) = |S_i - S_j|$, where $|x|$ denotes the absolute value of x.

We propose a new message model that includes both rate requirement and distance constraint. In this paper, each message stream is represented as $M_i = (A_i, maxD_i)$, where $A_i \leq maxD_i$, and where $maxD_i$ is the distance constraint, and A_i indicates the average rate requirement of

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

M_i in terms of the average distance between two consecutive instances of M_i. For example, if M_i requires 20% of the network bandwidth, then $A_i = \frac{1}{0.2} = 5$, which means that on average one of every five time slots should be assigned to M_i. The average rate requirement can be enforced in a TDMA template of length N by ensuring that each template contains at least $\lceil \frac{N}{A_i} \rceil$ instances of stream M_i.

Many communication applications require a QoS of both characteristics. In future interactive TV networks and video-on-demand applications, along a cable link of bandwidth B, an isochronous video stream M_i may require to transmit video frames at a rate of at least $R_i \times B$, where R_i is specified as a proportion of the total link bandwidth. So, the average distance between the transmission of consecutive frames must be smaller than or equal to $\frac{1}{R_i}$ time slots to satisfy the rate requirement. But the distance between consecutive video frames can tolerate being larger than $\frac{1}{R_i}$, as long as it does not exceed some maximum value, $maxD_i$, in order to maintain the human perception of the video and to avoid jagged display.

In this paper, we address the scheduling problem in a system that allows the message streams to dynamically arrive at and depart from the system. When a new message stream arrives at the system, some time slots in the template are already allocated to existing message streams. It is desirable that while the new message stream is scheduled in the same template, the allocation patterns of the already scheduled streams are not influenced, especially when the scheduling decisions need to be distributed to remote terminals, such as mobile objects. Thus, we assume that once a message stream is accepted and allocated in the template, its schedule will not be changed until the stream terminates. Let $location(i, j)$ denote the location of the time slot that is allocated to $M_{i,j}$. The problem is stated as follows.

Problem statement : Given a template of size N and a set $\mathcal{S}$ of m vacant time slots in the template, $\mathcal{S} = \{S_1, S_2, ..., S_m\}$, the problem is to allocate the vacant time slots to the instances of the new message stream $M_i = (A_i, maxD_i)$ such that the following conditions are true.

1. There are k_i instances of M_i in the template where $k_i = \lceil \frac{N}{A_i} \rceil$.
2. $location(i, j) \in \mathcal{S}, \forall j, 1 \leq j \leq k_i$.
3. $location(i, j+1) - location(i, j) \leq maxD_i, \forall j \geq 1$.

The first condition specifies that in order to satisfy the rate requirement, there should be k_i instances in the template, where $k_i = \lceil \frac{N}{A_i} \rceil$. Since it is expected that some other new streams will arrive in the future, the current stream should not occupy more slots than it needs so as to increase the schedulability of the future streams. Without this constraints, a simple way to find a feasible schedule for the rate requirement and distance constraint of the new stream is to allocate every vacant time slot to it. The second condition indicates that only vacant time slots are available to be allocated to the new arriving stream. The third condition specifies the distance constraint of the new stream.

Note that, because the TDMA template is applied repeatedly to control the transmission of the streams, if there are k_i instances of M_i in the template, then it is obvious that $location(i, j + k_i) = location(i, j) + N$.

We present two different scheduling algorithms to allocate the time slots in a TDMA template to dynamic message streams by making greedy choices and optimization choices, respectively.

The performances of the algorithms are evaluated and compared. Since dynamic message streams usually need to be responded to in a short time, the time complexity of the scheduling algorithm is an important performance measurement. Another critical measurement is the acceptance ratio, which is defined as the percentage of the successfully scheduled streams over the total number of streams arriving in the system. The measurement of acceptance ratio indicates the utilization of the network resource. Moreover, jitter control is an important performance issue for the QoS control in multimedia systems, such as audio/video presentations. In these applications, scheduling jitter, defined as the variance of the distances between all neighboring instances [10], must be low for a smooth servicing quality. We will evaluated and analyze the performance of the two algorithms according to the time complexity, the system-wide acceptance ratio and the scheduling jitter.

The rest of this paper is organized as follows. Section 2 introduces the related research work. Section 3 presents the two algorithms. The performance of the algorithms is analyzed in Section 4. The paper is concluded in Section 5.

2 Related Work

A large body of literature has been developed to guarantee the timing constraints related to servicing rate. For example, in the periodic message model [6], an isochronous message stream generates one message packet at the beginning of each period, and the packet has to be transmitted before the end of the period. Since exactly one packet is transmitted in each period, the period in the periodic message model is equivalent to the average distance A_i in our proposed message model. For the periodic message model, two algorithms widely used for scheduling periodic real-time tasks can be applied if we treat scheduling isochronous message streams on a link in a way similar to scheduling periodic real-time tasks on a processor. The first algorithm is the *rate-monotonic-scheduling* (RMS) algorithm which is proved to be the optimal fixed priority scheduling algorithm [11]. The other algorithm, *earliest-deadline-first* (EDF), is the optimal dynamic priority scheduling scheme that achieves 100% utilization of the resource[11, 13]. However, EDF and RMS are not designed to solve the problems with distance constraints. For a given instance of message stream M_i, the earliest time slot allocated by EDF or RMS can be at the beginning of the current period, whereas the latest time slot allocated for the next instance can be at the end of the next period. So, the distance between two consecutive instances can be as large as twice the length of the period.

The distance constraint scheduling problem is closely related to the *pinwheel problem* which is formally defined as follows [8]. Given a multi-set of n positive integers $A = \{a_1, ..., a_n\}$, where $a_1 \leq a_2 \leq ... \leq a_n$, the problem is to find an infinite sequence(schedule) of symbols from $\{$1,2,...,$n\}$ such that there is at least one symbol i within any interval of a_i slots. For example, if $A = \{3, 4, 4\}$, one solution sequence is $\{$1,1,2,3,1,1,2,3...$\}$ where $\{$1,1,2,3$\}$ is the template that repeats forever. Several algorithms have been derived to solve the pinwheel problem [8, 2, 3, 9], which apply a common operation, *specialization*, to transform A into another set $\bar{A} = \{\bar{a_1}, ..., \bar{a_n}\}$ such that $\bar{a_i} \leq a_i$ and $\bar{A}$ is easier to schedule. Clearly, the new set $\bar{A}$ is more conservative than A in the sense that it provides tighter distance constraints, and thus results in higher density.

Since the pinwheel problem does not address the servicing rate, it is inefficient to apply pinwheel algorithms to solve the problems with both the rate requirements and the distance constraints. For example, a message stream $M_i = (20, 25)$ requires that the average distance must be no smaller than 20 time slots, while a maximum distance of 25 time slots is tolerable. If we cast this problem into a pinwheel problem, a_i should be set to 20 in order to satisfy the rate requirement. The scheduler for the pinwheel problem will either succeed in finding a schedule with $maxD_i = 20$ or it will fail to find a schedule even though there may exist a feasible one with the more relaxed distance constraint specification of $maxD_i = 25$. Moreover, because the specialization procedure of the pinwheel algorithms require to know the parameters of all the message streams beforehand, pinwheel algorithms are not suitable to efficiently schedule dynamic message streams.

3 Dynamic allocation of message streams

In this section, we consider scheduling a dynamically arriving message stream in a TDMA template where some other message streams have already been scheduled.

3.1 A greedy algorithm Dynamic_greedy

We first presents a simple greedy algorithm to schedule a dynamic message stream. The algorithm is proven to be *locally optimal* in the sense that there exists a feasible scheme to schedule all the instances of the new message stream in the vacant slots of the template if and only if **Dynamic_greedy** can find it.

As defined in Section 1, given a list of m vacant slots $\mathcal{S} = \{S_1, S_2, ..., S_m\}$ in a template of size N, the problem is to allocate $k_i = \lceil \frac{N}{A_i} \rceil$ instances to a new message stream $M_i = (A_i, maxD_i)$. Assume that the list of vacant slots are sorted in increasing order, i.e., $S_i < S_j, \forall 1 \leq i < j < m$. In order to satisfy the distance constraint, there should be at least one instance of M_i in the first $maxD_i$ slots at the beginning of the template. Otherwise, the distance constraint will be violated when the template is applied repeatedly. We define a set $legal\_first$ which consists of the valid choices for $M_{i,1}$. That is, $legal\_first$ contains all the vacant slots that are smaller than or equal to $maxD_i$, $legal\_first = \{S_1, ..., S_v\}$, s.t. $S_v \leq maxD_i$. If $legal\_first$ is empty, then clearly it is impossible to find a feasible solution for the new stream. For the same reason, it is impossible to find a feasible solution if the distance between any two vacant time slots is larger than $maxD_i$.

As defined in the problem statement in Section 1, $k_i = \lceil \frac{N}{A_i} \rceil$ vacant time slots will be assigned to M_i to satisfy its rate requirement. Assume that the first instance of M_i is scheduled at time slot S_f ($S_f \in legal\_first$), then the $(k_i + 1)^{th}$ instance is fixed at slot $S_f + N$, due to the repetition of the template. The problem is reduced to scheduling the remaining $k_i - 1$ instances within the interval $(S_f, S_f + N)$ such that the distance constraint is satisfied. The function $min\_instance\_number$ presented in Figure 1 will apply a greedy policy to generate a set of the minimum number of instances that are needed in the interval to generate a feasible schedule.

```
num = 1;
selected = S_f;
A = {S_f};
while (N + S_f − selected > maxD_i), repeat
{
   limit = selected + maxD_i;
   get the maximum un-selected vacant slot S_x,
   which is smaller than or equal to limit,
   that is, selected < S_x ≤ limit and S_{x+1} > limit.
   if no S_x exists, return IMPOSSIBLE.
   selected = S_x;
   A = A ∪ {S_x};
   num = num + 1;
}
return num and the set of selected slots A.
```

Figure 1. The $min\_instance\_number$ function

In function $min\_instance\_number$, after deciding the position of the first instance, a vacant slot is selected that is as far as possible from the first instance, but within the distance constraint for the next instance. The scheme is repeated until it reaches the end point, $location(i, k_i + 1)$. In Figure 1, the variable num records the number of instances that have been scheduled. The variable $selected$ represents the selected time slot for the previously scheduled instance. During the loop that starts from line 5, the farthest legal position, $limit$, is calculated for the next instance based on the location of the previously selected slot and the distance constraint (line 6). Then the position of the vacant time slot closest to the limit is located, and assigned to the next instance (line 7 and 9). If no such a time slot exists, then there is a time interval of length $maxD_i$ that does not contain any vacant slot. So the function returns $IMPOSSIBLE$ (line 8), indicating that it is impossible to find a feasible schedule for the current message stream with $maxD_i$ constraint, The loop continues until the previously selected slot is close enough to the end point (line 4). Then the number of instances needed and the set of selected slots are returned (line 13).

The algorithm **Dynamic_greedy** is presented in Figure 2. It first makes sure that the number of vacant slots, m, is sufficient for the number of instances to satisfy the rate requirement of M_i (line 1). It selects a vacant slot S_f from $legal\_first$ as $location(i, 1)$, and calls the function $min\_instance\_number$ with S_f as the parameter (line 2 and 4). The algorithm rejects the new stream if the function returns $IMPOSSIBLE$ (line 5 and 6). If the minimum number of instances to satisfy the distance constraint is smaller than or equal to the required instance number, k_i, then there exists a feasible schedule which is constituted by the time slots selected in function $min\_instance\_number$ and the randomly selected ones from the remaining vacant slots to make a total of k_i slots (line 7-11). Otherwise, if the minimum number of instance needed is already larger than k_i, then there is no feasible schedule that starts with S_f (line 12). The next candidate in the set $legal\_first$ is picked for the same procedure. If all the time slots in $legal\_first$ are tried without success, then there is no feasible schedule, and the algorithm rejects the new stream (line 14).

In order to prove that algorithm **Dynamic_greedy** is locally optimal for the new message stream, the correctness of function $min\_instance\_number$ is proven in **Lemma 1**.

Lemma 1: Function $min\_instance\_number(S_f)$ produces a set of minimum number of vacant time slots that satisfies the distance constraint $maxD_i$ in the time interval $(S_f, S_f + N)$.

Proof Suppose that $B = \{S_f, b_1, b_2, ...\}$ $(S_f < b_1 < b_2 < ...)$ is a *correct* set which contains the minimum number of time slots constrained by $maxD_i$. Suppose the set of time slots generated by function $min\_instance\_number$

```
1.  if m < k_i, no feasible schedule, reject M_i;
2.  for every S_f ∈ legal_first = {S_1, ..., S_v}, do
3.  {
4.     instance_needed = min_instance_number(S_f);
5.     if it returns IMPOSSIBLE
6.        no feasible schedule, reject M_i;
7.     if instance_needed ≤ k_i
8.     {
9.        randomly select (k_i − instance_needed)
          vacant slots for the remaining instances;
10.       exit and output the k_i selected locations.
11.    }
12.    else There is no feasible schedule starting with S_f.
13. }
14. no feasible schedule, reject M_i;
```

Figure 2. The Dynamic_greedy Algorithm

is $C = \{S_f, c_1, c_2, ...\}$ $(S_f < c_1 < c_2 < ...)$. We can show that there is a correct solution which also selects c_1 as the first slot after S_f. If B starts with c_1, that is, $b_1 = c_1$, then B is the schedule. Otherwise, if $b_1 \neq c_1$, let $B' = B - \{b_1\} \cup \{c_1\}$. We will prove that B' is also a correct set.

Note that if $b_1 > c_1$, then the distance between S_f and b_1 is larger than $maxD_i$ which is not valid, because c_1 is the largest time slot within in $maxD_i$ constraint. If $b_2 < c_1$, then $b_2 - S_f < maxD_i$, which contradicts with the fact that B is a correct set, because the set $\{S_f, b_2, ...\}$ is a set of time slots that satisfies $maxD_i$ constraint and it has one less time slot than B. Thus, $b_1 < c_1 < b_2$. Figure 3 illustrates the relationship between B and B'. Obviously, replacing b_1 with c_1 does not influence the validity of the schedule since $c_1 - S_f \leq maxD_i$ and $b_2 - c_1 < b_2 - b_1 < maxD_i$. So there is a correct solution, B', that selects c_1 as the first slot after S_f.

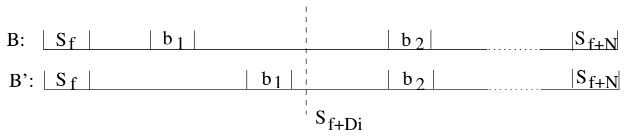


Figure 3. Relationship between B and B'.

Once the greedy choice of the first slot is made, the problem reduces to finding a correct solution in the time interval $(c_1, S_f + n)$. Therefore, after each greedy choice, we are left with a problem of the same form, but within a smaller interval. By induction on the number of choices made, we show that the greedy choice at every step produces a correct solution. □

Now that we have proved the correctness of the function $min\_instance\_number(S_f)$, the proof of the local optimality of algorithm **Dynamic_greedy** is trivial. Randomly selecting some vacant time slots in addition to the greedy choices will not violate the distance constraint, and the rate requirement is satisfied by enforcing the total number of instances in the template. So **Dynamic_greedy** will find a feasible schedule for M_i if there is one.

The basic step of **Dynamic_greedy** is to make a greedy choice for each of the k_i instances of M_i, and each choice needs at most $O(maxD_i)$ time to find the vacant slot in the range specified at line 7 in Figure 1. In the worst case, the procedure will be repeated $maxD_i$ times when all the time slots in $legal\_first$ are examined. So the worst case time complexity of the algorithm is $O(k_i \times maxD_i^2)$.

Although we have presented a locally optimal algorithm with polynomial time complexity to generate a feasible schedule for a dynamic message stream, it is an interesting issue to study how the instance allocation pattern of the current message stream will influence the schedulability of the message streams arriving in the future. The following example illustrates the characteristics of the allocation pattern and scheduling jitter of this algorithm.

Example 1: Assume in a template of size 12, there are 9 vacant slots, $\mathcal{S} = \{2, 3, 5, 6, 7, 8, 10, 11, 12\}$, as shown in Figure 4. In Figure 4, the first 12 slots in the time line is the scheduling template and is repeated continuously. The crossed time slots indicates the already occupied ones. A message stream M_i arrives with $A_i = 3$ and $maxD_i = 6$. So 4 instances need to be scheduled in the template. In this case, the set $legal\_first$ is $\{2, 3, 5, 6\}$, and the algorithm begins with choosing slot 2 for the first instance of M_i. By applying the function $min\_instance\_number()$, a set of greedy choices are returned as $\{2, 8\}$. In the worst case, the remaining 2 instances can be scheduled at slot 3 and 5, such that some instances are clustered together, and 5 consecutive time slots are occupied, as shown in Figure 4.

Example 1 shows that **Dynamic_greedy** tends to generate irregular allocation pattern where some of the instances are scattered, while some other instances are clustered. Note that the irregularity is especially significant when $maxD_i$ is large relative to A_i. This can lead to large scheduling jitter, as shown in Figure 4 where the largest distance between two instances is 6 and the smallest distance is 1. Moreover, the schedulability of the future message streams can be affected. In the example of Figure 4, assume the next arriving message stream is $M_j = (4, 5)$. Its distance constraint of 5 cannot be satisfied and it will be rejected, because the cluster generated by the algorithm results in an interval of length 5 without a single vacant slot.

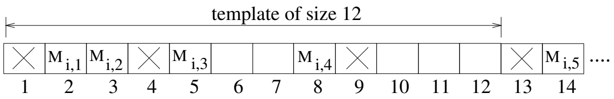


Figure 4. An example of the allocation pattern generated by Dynamic_greedy.

3.2 An optimization problem and its algorithm

Although the greedy algorithm is simple and locally optimal, we have observed that it tends to have large jitter and low acceptance ratio. In this section, we transform the original problem into an optimization problem. The solution to the optimization problem, **Dynamic_optimization**, is locally optimal for the new message stream as the greedy algorithm. It also improves the scheduling jitter and the allocation pattern.

In a template of size N with m vacant slots, assume the first instance of M_i is scheduled at vacant slot S_f ($S_f \in legal\_first$), we know that the time slot $S_f + N$ is dedicated to the $(k_i + 1)^{th}$ instance. Then the objective is to select $k_i - 1$ vacant slots from the remaining $m - 1$ vacant slots in the interval $(S_f, S_f + N)$, such that the distance constraint is satisfied. This problem can be transformed to a graph problem $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges. Let the vertices represent the $m + 1$ vacant slots in the interval $[S_f, S_f + N]$, that is, $\mathcal{V} = \{V_0, V_1, ..., V_m\}$ where vertex V_0 represents S_f, the first vacant slot in the interval, vertex V_1 represents the second one, and so on, and vertex V_m represents $S_f + N$, the last one in the interval. We define a directed edge (V_i, V_j) from vertex V_i to V_j if and only if $i < j$, and the distance between the two corresponding time slots is smaller than or equal to $maxD_i$. (Note that the graph is a directed acyclic graph.) Thus, the original scheduling problem is equivalent to the problem of finding a directed path from vertex V_0 to V_m consisting of k_i edges. Such a path exists if and only if there exists a feasible schedule in the template for the k_i instances of M_i with $maxD_i$ constraint. The vertices on the path correspond to the selected vacant slots in the feasible schedule. Note that the greedy algorithm can be applied to solve this equivalent problem.

The problem transformation is illustrated in Figure 5. In the figure, a template of size 12 has $m = 5$ vacant time slots, $\{1, 4, 6, 8, 10\}$. Assume that the next incoming stream in that example is M_j with $A_j = 4$ and $maxD_j = 5$. We need to schedule 3 instances in the template. If $M_{j,1}$ is allocated at slot 1, then the first instance in the next template, $M_{j,4}$ is fixed at slot 13. So vertex V_0 represents slot 1 and V_5 for slot 13. The transformation result is shown in Figure 5 below the time line. The three thick edges in the figure constitute a path from V_0 to V_5, indicating a feasible schedule for M_i.

The schedule result is the set of time slots in the template that are represented by the vertices on the path, which is $\{1, 6, 10\}$.

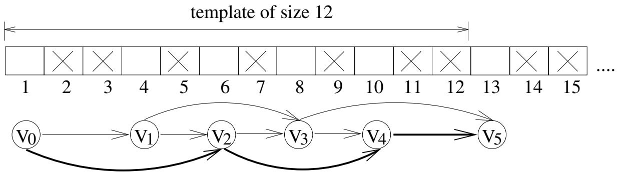


Figure 5. An example of the problem transformation.

For simplicity of notation, we use $distance(V_i, V_j)$ to denote the distance between slots S_a and slot S_b, where S_a and S_b are the vacant time slots represented by V_i and V_j, respectively. We further define the *weight of edge* (V_i, V_j) as $\frac{|distance(V_i,V_j)-A_i|}{A_i}$ if (V_i, V_j) exists in the graph. If the distance between slot S_a and slot S_b is close to the average distance A_i, then the value of $weight(V_i, V_j)$ is small. The *weight of a path* is the summation of the weights of all the edges on the path. A small path weight implies that the allocation pattern of the instances is close to be uniform. In order to decrease the scheduling jitter and to uniformly schedule the instances, the original problem can be solved by solving the following optimization problem. Clearly, the algorithm that solves this optimization problem is a locally optimal algorithm for the dynamically new stream M_i.

Optimization problem: Given a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{V_0, V_1, ..., V_m\}$, the problem is to find the shortest path from V_0 to V_m that consists of k_i edges.

The optimization problem can be solved by a dynamic programming technique [4]. Let us define a two-dimensional table $shortest(x, y)$ to be the weight of the shortest path from V_0 to V_x that has y paths. x is in the range of $[0, m]$ and y is in the range of $[0, k_i]$. Each table entry is calculated based on the following recursive relation.

- $shortest(0, 0) = 0$.
- $shortest(0, y) = \infty$, if $y \neq 0$. ∞ indicates that there is no path in this situation.
- $shortest(x, 0) = \infty$, if $x \neq 0$.
- $shortest(x, y) = MIN_{w<x}(shortest(w, y-1) + weight(V_w, V_x))$, if $x \neq 0$ and $y \neq 0$.

The formula in the last item is based on the observation that the shortest path from V_0 to V_x with y edges must be formed by two components. The first one is the shortest path from V_0 to a certain proceeding vertex V_w $(w < x)$ with $y - 1$ edges. The second component is the edge (V_w, V_x). The optimal value is calculated by searching the minimum calculation result among all the possible V_w's. Note that the maximum number of possible V_w's cannot be larger than $maxD_i$, because there is no edge connecting V_w and V_x if $distance(V_w, V_x)$ exceeds $maxD_i$.

If $shortest(m, k_i)$ is ∞, then no path is found to satisfy the requirements. Otherwise, $shortest(m, k_i)$ is the weight of the shortest path, and the path itself can be traced backward via an assistant table. The worst case execution time to find this solution is $O(m \times k_i \times maxD_i)$, because each table entry takes at most $O(maxD_i)$ time to compute.

Algorithm **Dynamic_optimization** is described in Figure 6. Since the recursive calculation needs to be executed at most $maxD_i$ times for all the time slots in the set $legal\_first$, the time complexity of this algorithm is $O(m \times k_i \times maxD_i^2)$. It is larger than the time complexity of the greedy algorithm by a factor of m, the number of vacant slots. When the network bandwidth is highly utilized, the extra time overhead can be insignificant.

1. *If $m < k_i$, the new stream is rejected because the rate requirement cannot be satisfied.*
2. *If the set $legal\_first$ is empty, then the message stream is rejected because its distance constraint cannot be satisfied.*
3. *For every S_f in $legal\_first = \{S_1, ..., S_v\}$, solve the transformed optimization problem, until a feasible schedule is found for the new message stream.*

Figure 6. The Dynamic_optimization Algorithm

If **Dynamic_optimization** is applied to Example 1 in Section 3.1, we obtain a uniform schedule of the 4 instances at slot $2, 5, 8, 11$ which is shown in the time line in Figure 7. So the scheduling jitter is 0 in this case. Moreover, because clustering is prevented, the next incoming message stream $M_j = (4, 5)$ that is rejected by **Dynamic_greedy** in Example 1 can now be accepted by **Dynamic_optimization**.

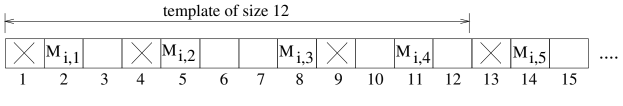
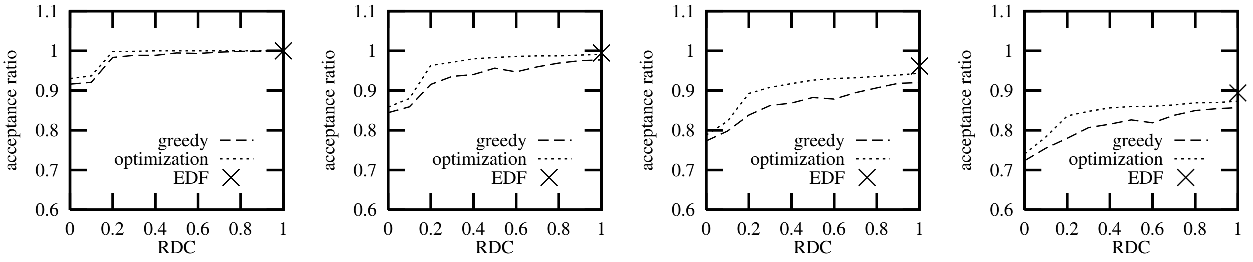


Figure 7. The allocation pattern generated by Dynamic_optimization on the example.

4 Performance evaluation and comparison

We have simulated the above two algorithms in an environment where message streams dynamically arrive and



(a) average load:30%

(b) average load:50%

(c) average load:70%

(d) average load:90%

Figure 8. The acceptance ratio measurement of the two algorithms

depart. For each message stream $M_i = (A_i, maxD_i)$, A_i is randomly generated with a uniform distribution in the range $[5, 100]$, and $maxD_i$ is generated according to *relative distance constraint (RDC)* which is defined as $(maxD_i - A_i)/A_i$. RDC is one of the input parameters of the simulator. It measures the flexibility of the distance constraint relative to the average distance.

In the simulation, the connection duration of M_i is uniformly distributed between 2000 time slots and 5000 time slots. The average system workload is another input parameters to the simulator. Given a certain average system workload, the inter-arrival time of the message streams can be determined according to the rates of the arriving streams and the average connection duration.

The system-wide acceptance ratio of the two algorithms when the average system work load is equal to $0.3, 0.5, 0.7$ and 0.9 is plot in Figure 8(a), Figure 8(b), Figure 8(c) and Figure 8(d), respectively. The X-axis of the figures represents the input parameter RDC which is in the range of $[0, 1]$. The Y-axis represents the system wide message stream acceptance ratio, which is the percentage of successfully scheduled message streams over 10,000 incoming message streams. The two curves in each figure plot the performance of the two algorithms. When RDC is close to 0, the choices of the possible scheduling pattern is limited, because of the restrictive distance constraints. When RDC is increased, the acceptance ratios of the two algorithms are increased, due to the larger flexibility generated by the relaxed distance constraints. However, in this case, the performance of algorithm **Dynamic_optimization** is increased faster than algorithm **Dynamic_greedy**. Because when the distance constraints are not tight, the allocation pattern generated by the greedy algorithm becomes irregular, which may hurt the future incoming streams, as explained earlier. Whereas the optimization algorithm prevents the irregular allocation pattern by solving the optimization problem. This performance advantage of **Dynamic_optimization** is achieved at the cost of time complexity.

In order to evaluate the acceptance ratio of the two algorithms, we also apply the general EDF algorithm to schedule the same set of dynamic message streams, taking A_i as the period of each stream M_i. EDF will accept a new stream if the current resource utilization plus the average rate of the new stream is smaller than or equal to 100%. Though EDF can fully utilize the resource and achieve the highest possible acceptance ratio, it can only guarantee the distance constraints when $RDC = 1$, since the distance between two consecutive instances in a EDF schedule can be as large as twice the length of the period, as pointed out in Section 2. So the point on the vertical line of $RDC = 1$ is the acceptance ratio of EDF. It is shown that when $RDC = 1$, the acceptance ratio of **Dynamic_optimization** is close to the highest possible value, even when the average system work load is high.

For each accepted task, the simulator also measures the scheduling jitter which is the variance of the distances between all neighboring instances. Figure 9 shows the average scheduling jitter of all the accepted message streams versus the relative distance constraint when the system workload is 0.9. The simulation results under other kinds of workload follow a similar trend and is not shown in this paper. In the figure, the curve of the optimization algorithm is almost flat. It shows that the average jitter is almost unchanged even when a large distance constraint is allowed by the relaxed requirements. On the other hand, the jitter of the greedy algorithm rapidly increases following the increasing of RDC. Again, this illustrates the importance of the allocation pattern for the QoS results.

5 Conclusion

In this paper, we propose a new real-time message model with both rate requirements and distance constraints. Two algorithms, **Dynamic_greedy** and **Dynamic_optimization** are derived to schedule dynamic real-time message streams

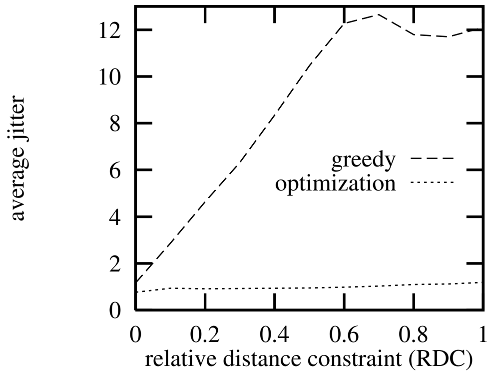


Figure 9. The relative jitter measurement, average system workload: 0.9

in a TDMA frame, satisfying the specified QoS requirements. The performances of the two algorithms are evaluated and compared in terms of time complexity, acceptance ratio and scheduling jitter via simulation. The characteristics of the two algorithms are summarized in Table 1.

algorithm	**greedy**	**heuristic**
time	$O(k_i \cdot maxD_i^2)$	$O(m \cdot k_i \cdot maxD_i^2)$
jitter	large	small
accept ratio	lower	higher

Table 1. Comparison of the two algorithms

The comparison of the two algorithms shows that The greedy algorithm is applicable in systems that require low run time overhead. The optimization algorithm is suitable for systems that need to schedule dynamic real-time traffic with high acceptance ratio and small scheduling jitter, whereas a certain scheduling overhead is tolerable, or the time slots in the TDMA template is highly utilized such that the time overhead is not significant.

Finally, we note that the algorithms described in this paper can be applied to schedule real-time tasks with invocation rate specifications and distance constraints in the systems which use the cyclic executive [12] scheduling paradigm. Similar to the TDMA protocol, cyclic executive systems generate an effective schedule off-line, then dispatch and execute the tasks cyclically according to the schedule.

References

[1] Sanjoy Baruah and Azer Bestavros. Pinwheel Scheduling for Fault-tolerant Broadcast Disks in Real-Time Database Systems. *Proceedings of the IEEE International Conference on Data Engineering*, 4 1997.

[2] M. Y. Chan and F. Y.L. Chin. General Schedulers for the Pinwheel Problem Based on Double-Integer Reduction. *IEEE Trans. on Computers*, 41(6):755–768, 1992.

[3] M. Y. Chan and F. Y.L. Chin. Schedulers for Larger Classes of Pinwheel Instances. *Algorithmica*, 9:425–462, 1993.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction To Algorithms*. The MIT Press, Cambridge, Massachusetts London, England, 1994.

[5] B.W. Davis, C. Graham, D. Stamm, and C. Parker. Tactical Digital Information Link (TADIL) J Range Extension (JRE). *IEEE MILCOM'97 Proceedings*, 1:408–412, 11 1997.

[6] L. Dong, D. Mosse, and R. Melhem. Time Slot Allocation for Real-Time Messages with Negotiable Distance Constrains. *Proceedings of IEEE Real-time Technology and Applications Symposium*, pages 131–136, 6 1998.

[7] 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.

[8] R. Holte, A. Mok, L. Rosier, I. Tulchinsky, and D. Varvel. The Pinwheel: A Real-Time Scheduling Problem. *Proc. of the 22nd Hawaii International Conference on System Science*, pages 693–702, 4 1989.

[9] C.W. Hsueh and K.J. Lin. An Optimal Pinwheel Scheduler Using the Single-Number Reduction Technique. *Proceedings of IEEE Real-time Systems Symposium*, pages 196–205, 1996.

[10] C.W. Hsueh and K.J. Lin. Schedulability Comparisons Among Periodic and Distance-Constrained Real-Time Schedulers. *Proceedings of the 4th International Workshop on Real-Time Computing Systems and Applications*, pages 60–66, 10 1997.

[11] 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.

[12] C.D. Locke. Software Architecture for Hard Real-Time Applications:Cyclic Executives vs. Fixed Priority Executives. *rtsj*, 4:37–53, 1992.

[13] M. Spuri, G.C. Buttazzo, and F. Sensini. Robust aperiodic scheduling under dynamic priority systems. *Proceedings of IEEE Real-time Systems Symposium*, 12 1995.