The Performance Guarantee of Rate Monotonic for Periodic Hard-Real-Time Packet Scheduling of Distributed Systems

Maurizio A. Bonuccelli, M. Claudia Clò

Dipartimento di Informatica, Università di Pisa, Corso Italia 40
56125 Pisa, Italy. Phone: +39-050-887255; Fax: +39-050-887226
e-mail: bonucce@di.unipi.it, clo@di.unipi.it

Abstract

In this paper, we investigate the worst case performance of a well known algorithm (a de facto standard in processor scheduling), Rate Monotonic (RM) algorithm, when applied to packet scheduling in distributed systems. We assume that the processing elements communicate via a multistage interconnection network, and that the system is synchronous. The simultaneous transmission over the same input or output port of two or more packets results in a collision. Collided packets are damaged, and need to be retransmitted later. This causes a performance degradation in terms of both throughput and delay. So, collisions must be avoided. The special type of traffic to be scheduled by RM is a periodic hard-real time one, and the objective is to schedule all the packets within their deadlines. We give the condition guaranteeing that the algorithm is always able to produce a schedule meeting the above objective. Such a condition can be effectively used as a fast feasibility test before actually running the algorithm.

KEY WORDS: distributed systems, multistage interconnection networks, packet scheduling, hard-real-time communication, performance guarantee, Rate Monotonic algorithm, feasibility test.

1 Introduction

Distributed memory processing systems centered around multistage interconnection networks are becoming increasingly widespread for their favourable cost-performance ratio. Notable examples of such systems are Network of Workstations and Network of PC's. Such systems are formed by a set of processing elements, each equipped with its own memory modules, and communicating by means of an interconnection network.

Usually, in these systems the communication is packetized and synchronous, and the interconnection network is a nonblocking permutation one, like crossbar switch or Benes network. The processing elements are linked to the interconnection network by means of input and output ports. When input or output ports are involved in more than one transmission at a time, a *collision* occurs and all the transmissions with common ports are damaged. Collided transmissions are useless and must be discarded and repeated later. This is a source of performance degradation, both from a system point of view (the network is used for transmitting messages that must be discarded, thus wasting bandwidth) and from a user perspective (the messages undergo a longer delay before being successfully received). Performance degradation can be very serious, and collisions should be avoided. However, we can have concurrent transmissions, provided that they do not involve the same port more than once. Collision avoidance can be done by a proper scheduling of the messages. For a given set of messages, several different schedules exist, each with a specific impact on the overall system performance.

The above scheduling problem is not specific of interconnection networks, but occurs in a wide variety of systems, called *multichannel single-hop* communication systems. For instance, WDM purely optical networks, when operated under a Time Division Multiplexing (TDM) scheme, fall into this class. Usually, communication is organized into *frames*. Each frame is divided into equal length *time slots*, representing the transmission of unitary blocks of information, called *packets*, originating from longer messages. Packets cannot be split and their transmission cannot be preempted. Packet scheduling is often called *Time Slot Assignment* (TSA for short), and has been extensively studied mainly in a communication setting [BaB94, BBD96, BCW81, BGW91, GBW83, PS94, RoS96, BM96, CCA96, THW$^+$96, JMI95, Jain97] with the objective of minimizing the schedule length.

Real time communication requires a timely delivery of periodic messages to the destination. Such a timely delivery is usually represented by a deadline associated to each message. Deadline meeting is essential for ensuring a safe system working, or reaching a minimum quality level, acceptable by the end user. The goal of the real time packet scheduling problem studied in this paper is to find a *feasible schedule*, whenever it exists, namely a schedule meeting the deadlines of all the messages. This is the most important objective in real time settings, where high delays are not tolerated.

Processor scheduling of single- and multi-processor computing systems for real time applications has been widely studied since the early 70's (a survey of the known results can be found in [KLR94, Leh94]), mainly under the objective function considered here. On the contrary, real time packet scheduling has received attention only recently. An earlier attempt to study the TSA problem of multimedia traffic can be found in [BGW91]. Real time transmissions with explicit deadlines have been investigated in [THW$^+$96]. In that paper, however, a model different from the one assumed here is considered. In

[PhLi], four heuristic algorithms are presented and the success rate of each of them is evaluated through simulation. In [GH97] other two heuristic algorithms have been proposed with the restriction that each message is formed by only one packet. So, such messages are a little different from those considered in this paper (we assume multiple packet messages). The performance guarantees of the two proposed heuristics are established to 1/4 and 1/14 in [GH97].

In a companion paper [BC99], these authors establish the performance guarantee of Earliest Due Date heuristic for the problem studied here. Earliest Due Date is a de facto standard among *dynamic* priority hard-real-time scheduling algorithms. In [BC99], it is assumed that each input/output pair can exchange one (multipacket) message at most. In [BC98a], the non periodic version of the problem considered in this paper, has been investigated. There, it is shown that the problem is computationally intractable (NP-complete) even in very restricted cases. Heuristics are proposed, and their performance (in terms of utilization factors) is presented.

In this paper, we continue the investigation started in [BC99], by considering another heuristic, Rate Monotonic (RM for short), and assessing its worst case performance. This heuristic is widely used in different (e.g. shared memory multiprocessors) settings [SSDB95], where it is considered a "de facto" standard among *static* scheduling algorithms. Static scheduling algorithms are often preferred to dynamic ones because usually they are easier to implement and run faster (however, dynamic scheduling algorithms in general produce schedules with a better performance). Thus, it is important to assess the performance guarantee of RM in this new setting. A more general model than that of [BC99] is assumed here. In fact, we allow multimple messages (each with its own deadlines, period and number of packets) to be sent between the same input/output pair. This model is the one assumed in [PhLi]. In Section 2, we briefly state the problem, and give some conditions, necessary for producing a feasible schedule. Then, we introduce the heuristic (Section 3), and present a worst case performance (in terms of utilization factors) for it. Such a worst case performance is also called performance guarantee and is very important in real time applications, since fast tests assessing whether the algorithm will produce a schedule meeting all the deadlines or not, without actually running the algorithm (a time consuming process), can be derived from it. This gives insights in the system design stage, also. Finally, conclusions and open problems terminate the paper.

2 PHRT-TSA Problem

We now present the mathematical formulation of the periodic hard real time scheduling problem for distributed systems studied here, called PHRT-TSA. The problem consists in scheduling multi-packet messages in multichannel single-hop communication systems. Messages are periodic and have an associated deadline. Each message needs to be transmitted in each period within its deadline, in order to guarantee the system correct working. Besides, when two or more messages are transmitted simultaneously over the same channel, or from the same origin or to the same destination, a message collision occurs. To avoid message collisions, a proper schedule of the messages must be devised.

An $n \times n \times m$ nonnegative integer *traffic matrix* $T = (t_{ijh})$ models the system traffic, $1 \leq i, j \leq n$, $1 \leq h \leq m$. Rows and columns of matrix T represent origins and

destinations, respectively (or input and output ports), while the third dimension, called height, is used to represent different messages that the same origin/destination pair must exchange. We assume that such messages are at most m. The value of t_{ijh} is the number of packets to be sent from origin i to destination j for the h-th message. If such a pair (i,j) has $p < m$ messages, then $t_{ijh} > 0$ for $1 \leq h \leq p$, and $t_{ijh} = 0$ for $p < h \leq m$.
An $n \times n \times m$ nonnegative integer *deadline matrix* $D = (d_{ijh})$ contains the deadlines corresponding to matrix T entries, $1 \leq i,j \leq n$, $1 \leq h \leq m$, where d_{ijh} corresponds to t_{ijh}. Let r_{ijh} be the release time of entry t_{ijh}. The traffic is *periodic*, and a period p_{ijh} is associated to each message t_{ijh}. Each message must be transmitted once in each period. Hence, the first and k-th occurrences of a periodic message t_{ijh} take place in time intervals $[r_{ijh}, d_{ijh}]$ and $[r_{ijh}+(k-1)p_{ijh}, d_{ijh}+(k-1)p_{ijh}]$, $k > 1$, respectively. Obviously, the deadline, release time and period of message t_{ijh} are related in the following way:

$$\begin{array}{ccccc} r_{ijh} & < & d_{ijh} & \leq & r_{ijh} + p_{ijh} \\ t_{ijh} & \leq & d_{ijh} & \leq & p_{ijh} \end{array}$$

In this paper, we shall assume that $r_{ijh} = 0$ and $p_{ijh} = d_{ijh}$. Hence, a message t_{ijh} must be scheduled for transmission in each interval $[(k-1)d_{ijh}, kd_{ijh}]$, $k \geq 1$. This simplifying assumption is commonly made in real time scheduling [KLR94, Leh94, LeW82, LiL73].
A feasible schedule must meet traffic timing requirements, as well as message collisions constraints: a message transmission has neither to start before its release time, nor to be completed after its deadline, during each period. Besides, in each time slot, at most one message can be scheduled for transmission over one channel, or from an origin or to a destination.

Note that a deadline kd_{ijh} is both the deadline of the k-th message occurrence and the release time of $(k+1)$-th one.

Appendix A shows a feasible schedule for a PHRT-TSA problem instance with $m = 1$.

Let us now give some definitions we shall use in the following for establishing some properties of the PHRT-TSA problem. Let us assume that there are M different deadline values, i.e. $d_{ijh} \in \{\delta_1, \delta_2, \ldots, \delta_M\}$. Without loss of generality, assume that $\delta_1 < \delta_2 < \ldots < \delta_M$.

Definition 1 A *switching matrix* S is a traffic matrix with at most one nonzero entry in each line, namely in each row or column.

Definition 2 r_i and c_j will denote the *row* and *column sums* of $\frac{T}{D}$, respectively ($1 \leq i,j \leq n$):

$$r_i = \sum_{j=1}^{n}\sum_{h=1}^{m}\frac{t_{ijh}}{d_{ijh}} \ , \ c_j = \sum_{i=1}^{n}\sum_{h=1}^{m}\frac{t_{ijh}}{d_{ijh}}$$

Definition 3 A matrix line is called *exposed* if all entries in that line are zero and *covered* if there is at least one non-zero entry in it.

Definition 4 Let $X_k(i)$ and $Y_k(j)$ be the sum of the traffic entries in *row* i and in *column* j with deadline δ_k $(1 \leq k \leq M)$*:*

$$X_k(i) = \sum_{1 \leq j \leq n, 1 \leq h \leq m : d_{ijh} = \delta_k} t_{ijh} \ , \ Y_k(j) = \sum_{1 \leq i \leq n, 1 \leq h \leq m : d_{ijh} = \delta_k} t_{ijh}$$

Clearly, the following equalities hold:

$$r_i = \sum_{k=1}^{M} \frac{X_k(i)}{\delta_k} \ , \ c_j = \sum_{k=1}^{M} \frac{Y_k(j)}{\delta_k}$$

Definition 5 Let L_k be the *largest sum of entries* with deadline δ_k in each line of matrix T $(1 \leq k \leq M)$:

$$L_k = \max_{i,j}\{X_k(i), Y_k(j)\}$$

Definition 6 An entry t_{ijh} is said to be of *type* k if $d_{ijh} = \delta_k$ $(1 \leq k \leq M)$.

Let us consider an instance of the PHRT-TSA problem. An obvious necessary condition for the schedulability is the following $(1 \leq i, j \leq n, 1 \leq h \leq m)$:

$$\frac{t_{ijh}}{d_{ijh}} \leq 1 \tag{1}$$

Indeed, if this condition does not hold for an entry (i_1, j_1, h_1), i.e. if $\frac{t_{i_1,j_1,h_1}}{d_{i_1,j_1,h_1}} > 1$, then $t_{i_1,j_1,h_1} > d_{i_1,j_1,h_1}$ and it is not possible to completely transmit t_{i_1,j_1,h_1} within its deadline d_{i_1,j_1,h_1} in any period. Therefore, d_{i_1,j_1,h_1} cannot be met and a feasible schedule cannot exist.

We now give a less obvious necessary condition for the schedulability of $\frac{T}{D}$.

If a feasible schedule for the PHRT-TSA problem exists, then the following conditions must hold:

$$\begin{cases} \sum_{j=1}^{n} \sum_{h=1}^{m} \frac{t_{ijh}}{d_{ijh}} \leq 1 \quad (1 \leq i \leq n) \\ \sum_{i=1}^{n} \sum_{h=1}^{m} \frac{t_{ijh}}{d_{ijh}} \leq 1 \quad (1 \leq j \leq n) \end{cases} \tag{2}$$

These inequalities can be rewritten in terms of row and column sums, r_i and c_j:

$$\begin{cases} r_i \ \leq \ 1 \ \ (1 \leq i \leq n) \\ c_j \ \leq \ 1 \ \ (1 \leq j \leq n) \end{cases} \tag{3}$$

3 Performance Guarantee of Algorithm RM

Rate Monotonic (*RM*) is a static (fixed-priority) scheduling algorithm, widely used in real time systems to schedule periodic processes [SSDB95, BF97]. It gives higher priority to the entries with smaller period. Since we assume that the message periods coincide with the deadlines, it schedules the entries t_{ijh} $(t_{ijh} > 0)$ according to increasing deadlines and meeting the switching matrix constraints.

In Appendix B, a feasible schedule for a matrix with $m = 3$, generated by RM, is given. A more detailed algorithm description can be found in Appendix C.

In order to derive algorithm RM performance guarantee, let us introduce algorithm SIMPLE: it gives a static priority to the entries and, like RM, schedules them according to increasing deadline values. So, type 1 entries have the highest priority, while type M entries have the lowest one. The main difference with RM is that SIMPLE never schedules in the same switching matrix entries with different priorities.
Thus SIMPLE schedules first type 1 entries only, for L_1 time slots; then, it schedules type 2 entries only, for L_2 slots or until the next δ_1 deadline. After completely scheduling type 1 and 2 entries, SIMPLE schedules type 3 entries and so on.
Note that algorithm RM never performs worse than SIMPLE, and only in the worst case, it behaves like SIMPLE. Such a worst case occurs when no entries with lower priority can be scheduled during the scheduling of entries with higher priority. So, the performance guarantee of SIMPLE holds for RM, too. Notice that algorithm SIMPLE always meets all the deadlines with highest priority (i.e. type 1 deadlines). Indeed, entries with lower priority cannot prevent the scheduling of entries with the highest one. Contrary to RM, algorithm SIMPLE does not produce a feasible schedule for the matrix presented in Appendix B.

In order to establish the performance guarantee of SIMPLE with an arbitrary number of deadlines, we first prove a bound for the special case with two deadline types, δ_1, δ_2, with $\delta_1 < \delta_2$. In this case, consider an arbitrary interval between two consecutive δ_2 deadlines, say $[k\delta_2, (k+1)\delta_2]$ and divide it in subintervals $I_1, I_2, \ldots, I_q$ such that $I_1 = [k\delta_2, l\delta_1]$, $I_2 = [l\delta_1,\ (l+1)\delta_1]$, $\ldots, I_q = [f\delta_1, (k+1)\delta_2]$, where $l\delta_1$ is the earliest deadline after $k\delta_2$. Let $|I_i|$ denote the length of interval I_i. The following lemma holds, where $u = \lfloor \frac{\delta_2}{\delta_1} \rfloor$ and $r = \delta_2 \bmod \delta_1$.

Lemma 1 *The number N of subintervals of length δ_1, i.e. subintervals $I_2, \cdots, I_{q-1}$, in $[(k-1)\delta_2, k\delta_2]$, is*

$$N = \begin{cases} u & \text{if } |I_1| + |I_q| < \delta_1 \\ u - 1 & \text{if } |I_1| + |I_q| \geq \delta_1. \end{cases}$$

Besides, if $|I_1| + |I_q| < \delta_1$, then $|I_1| + |I_q| = r$, otherwise $|I_1| + |I_q| = \delta_1 + r$.

Proof: If $|I_1| + |I_q| < \delta_1$, then $|I_1| + |I_q| = r$. All the intervals $I_2, \cdots, I_{q-1}$ have length equal to δ_1, and so: $\delta_2 \bmod \delta_1 = r$. Then, $N = u$.

If $|I_1| + |I_q| > \delta_1$, then it results $|I_1| + |I_q| = \delta_1 + r$. Since the intervals $I_2, \cdots, I_{q-1}$ all have length equal to δ_1, we can decompose δ_2 in the following way:

$$\delta_2 = u\delta_1 + r = (u-1)\delta_1 + \delta_1 + r\,,$$

besides $\delta_2 = |I_1| + \sum_{i=2}^{q-1} |I_i| + |I_q|$, and so we have:

$$\sum_{i=2}^{q-1} |I_i| = (u-1)\delta_1\ ,\ |I_1| + |I_q| = \delta_1 + r.$$

This is the only possible decomposition of δ_2 in $|I_i|$, $1 \leq i \leq q$, since

$$|I_1| < \delta_1\ ,\ |I_q| < \delta_1\ ,\ |I_1| + |I_q| > \delta_1.$$

Then, $N = u - 1$.

We can now assess the performance guarantee of SIMPLE in the two deadline values case.

Theorem 1 *Let a matrix* $\frac{T}{D}$ *be given, with* $d_{ijh} \in \{\delta_1, \delta_2\}$, $\delta_1 < \delta_2$, $\delta_2 = u\delta_1 + r$. *SIMPLE always generates a feasible schedule* $\Longleftrightarrow$

$$L_2 \leq \delta_2 - uL_1 - \min(r, L_1) \tag{4}$$

Proof: Let us prove the theorem with $r \leq L_1$ first, and with $r > L_1$ later.
i) Assume first that $r \leq L_1$.
$\Leftarrow$
We prove the sufficient condition by induction. First, we show that deadline δ_2 is met by assuming that equation (4) holds. Afterwards, we assume that $k\delta_2$ is met and show that $(k+1)\delta_2$ is met.
Let us assume that $L_2 \leq \delta_2 - uL_1 - r$. Consider time interval $[0, \delta_2]$. Let $I_1, I_2, \cdots, I_q$ be the subintervals in $[0, \delta_2]$ between two consecutive deadlines, i.e. $I_1 = [0, \delta_1], I_2 = [\delta_1, 2\delta_1], \cdots, I_q = [u\delta_1, \delta_2]$.
Since $|I_q| = r$, then by Lemma 1 there are exactly u intervals of length δ_1 in $[0, \delta_2]$. So, type 1 entries are scheduled for L_1 slots at least u times, and there are $u(\delta_1 - L_1)$ slots available for transmission of type 2 entries. Besides, since $r \leq L_1$, in interval I_q only type 1 entries are scheduled and there are no time slots available for type 2 entries. Therefore, since

$$\delta_2 - uL_1 - r = u\delta_1 - uL_1 \tag{5}$$

if $L_2 \leq \delta_2 - uL_1 - r$, all type 2 entries can be scheduled within deadline δ_2, which is met.
Let us assume now that deadline $k\delta_2$ is met $(k > 1)$. We want to show that deadline $(k+1)\delta_2$ is met. Hence, we consider the interval $[k\delta_2, (k+1)\delta_2]$. Let $I_1, I_2, \cdots, I_q$ be its subintervals between two consecutive δ_1 deadlines, i.e. $I_1 = [k\delta_2, l\delta_1], I_2 = [l\delta_1, (l+1)\delta_1], \cdots, I_q = [f\delta_1, (k+1)\delta_2]$.
On the basis of Lemma 1, we can consider two cases:

1) $|I_1| + |I_q| = r$
2) $|I_1| + |I_q| = \delta_1 + r$

i.1) Assume that $|I_1| + |I_q| = r$. From Lemma 1, the number of subintervals I_i of length δ_1 in $[k\delta_2, (k+1)\delta_2]$ is u, where $u = \lfloor \frac{\delta_2}{\delta_1} \rfloor$. So, there are at least $u(\delta_1 - L_1)$ slots available for the scheduling of type 2 entries. If equation (4) holds, all type 2 entries are scheduled for transmission before time slot $(k+1)\delta_2$ because of (5), so this deadline is met.

i.2) Let us assume now that $|I_1| + |I_q| = \delta_1 + r$. ¿From Lemma 1, the number of subintervals I_i of length δ_1 in $[k\delta_2, (k+1)\delta_2]$ is $u - 1$. Then, there are at least $(u-1)(\delta_1 - L_1)$ slots available for the transmission of type 2 entries. Since $I_1 = [k\delta_2, l\delta_1]$ and $I_q = [f\delta_1, (k+1)\delta_2]$, in this case we have:

$$f\delta_1 = l\delta_1 + (u-1)\delta_1 \tag{6}$$

There are two more subcases:

a) $|I_q| > L_1$
b) $|I_q| \leq L_1$

i.2.a) If $|I_q| > L_1$, then

$$f\delta_1 + L_1 < (k+1)\delta_2. \tag{7}$$

Let $\alpha = (k+1)\delta_2 - (f\delta_1 + L_1)$. Type 2 entries can be scheduled in I_q for α slots. We now compute the number of time slots in I_1 available for type 2 entries. In interval $[(l-1)\delta_1, l\delta_1]$, type 1 entries are scheduled for L_1 slots. Hence:

$$\begin{aligned}(l-1)\delta_1 + L_1 = \quad & f\delta_1 - u\delta_1 + L_1 \quad < (k+1)\delta_2 - u\delta_1 \;= \\ = \quad & k\delta_2 + \delta_2 - u\delta_1 \quad = k\delta_2 + r\end{aligned}$$

which follows from (6), (7), and $\delta_2 = u\delta_1 + r$. So, $(l-1)\delta_1 + L_1 < k\delta_2 + r$. Let $\beta = l\delta_1 - (k\delta_2 + r)$. Type 2 entries can be scheduled in I_1 for at least β slots (these are the slots from $k\delta_2$ to $l\delta_1$, during which only type 2 entries must be scheduled). ¿From (6) and since $\delta_2 = u\delta_1 + r = (u-1)\delta_1 + \delta_1 + r$, we have:

$$\begin{aligned}\beta + \alpha &= l\delta_1 - (k\delta_2 + r) + (k+1)\delta_2 - (f\delta_1 + L_1) = \\ &= l\delta_1 - k\delta_2 - r + k\delta_2 + \delta_2 - f\delta_1 - L_1 = \\ &= l\delta_1 - r + \delta_2 - l\delta_1 - (u-1)\delta_1 - L_1 = \\ &= -r + \delta_2 - (u-1)\delta_1 - L_1 = \delta_1 - L_1\end{aligned}$$

Thus, type 2 entries can be scheduled in I_1 and I_q for at least $\delta_1 - L_1$ slots. So, in total they are scheduled in $[k\delta_2, (k+1)\delta_2]$ for $u(\delta_1 - L_1)$ slots. If $L_2 \leq \delta_2 - uL_1 - r$, then type 2 entries are all scheduled before $(k+1)\delta_2$ and this deadline is met.

i.2.b) Now let $|I_q| \leq L_1$. Then

$$f\delta_1 + L_1 \geq (k+1)\delta_2 \tag{8}$$

This means that no type 2 entries can be scheduled in I_q.
During $[(l-1)\delta_1, l\delta_1]$, SIMPLE schedules type 1 entries for L_1 slots. Then,

$$\begin{aligned}(l-1)\delta_1 + L_1 = \quad & f\delta_1 - ud_1 + L_1 \quad > (k+1)\delta_2 - u\delta_1 \;= \\ = \quad & k\delta_2 + \delta_2 - u\delta_1 \quad = k\delta_2 + r\end{aligned}$$

because of (6), (8) and $\delta_2 = u\delta_1 + r$. Summarizing, $(l-1)\delta_1 + L_1 > k\delta_2 + r$. A total of

$$l\delta_1 - [(l-1)\delta_1 + L_1] = \delta_1 - L_1$$

time slots are available in I_1 for scheduling type 2 entries. So, there is a total of $u(\delta_1 - L_1)$ slots available for type 2 entries scheduling. If $L_2 \leq u(\delta_1 - L_1)$, deadline $(k+1)\delta_2$ is met.

$\Rightarrow$
Let us assume that $L_2 > \delta_2 - uL_1 - r$, and consider interval $[0, \delta_2]$. Let $I_1, I_2, \cdots, I_q$ be the subintervals in $[0, \delta_2]$ between two consecutive deadlines. From Lemma 1, the number of such subintervals with length δ_1 is u, where $u = \lfloor \frac{\delta_2}{\delta_1} \rfloor$. In each such subinterval,

SIMPLE schedules type 1 entries for L_1 slots, so leaving $(\delta_1 - L_1)$ slots available for type 2 entries. Since $|I_q| = r \leq L_1$ $(r = \delta_2 - u\delta_1)$, SIMPLE does not schedule type 2 entries in I_q. Therefore, there is a total of $u(\delta_1 - L_1)$ slots available for type 2 entries in $[0, \delta_2]$. Finally, if $L_2 > u(\delta_1 - L_1)$, then type 2 entries cannot be all scheduled in $[0, \delta_2]$ and deadline δ_2 is not met: a contradiction.

ii) Let us assume that $r > L_1$.
$\Leftarrow$
We prove the sufficient condition by induction. Let us first show that deadline δ_2 is met when equation (4) holds. Consider time interval $[0, \delta_2]$. Let $I_1, I_2, \cdots, I_q$ be the subintervals in $[0, \delta_2]$ between two consecutive deadlines, i.e. $I_1 = [0, \delta_1]$, $I_2 = [\delta_1, 2\delta_1], \cdots$, $I_q = [u\delta_1, \delta_2]$. Since $|I_q| = r$, there are exactly u subintervals of length δ_1 in $[0, \delta_2]$ (Lemma 1). Then, SIMPLE schedules type 1 entries for L_1 slots in each such subinterval. So, there are at least $u(\delta_1 - L_1)$ slots available for scheduling of type 2 entries during $[0, \delta_2]$. Besides, since $r > L_1$, SIMPLE is able to schedule type 1 entries in I_q for L_1 slots and type 2 entries for the remaining $(r - L_1)$ slots. Hence, there are $u(\delta_1 - L_1) + (r - L_1)$ slots available for type 2 entries in $[0, \delta_2]$. Since

$$\delta_2 - uL_1 - L_1 = u\delta_1 + r - uL_1 - L_1, \tag{9}$$

if $L_2 \leq \delta_2 - uL_1 - L1$, then all type 2 entries can be scheduled within deadline δ_2, which is met.

Let us assume now that deadline $k\delta_2$ is met $(k > 1)$. We shall show that $(k+1)\delta_2$ is met. Consider interval $[k\delta_2, (k+1)\delta_2]$ and its subdivision in intervals, $I_1, I_2, \cdots, I_q$, between two consecutive deadlines. There are two cases:

1) $|I_1| + |I_q| = r$
2) $|I_1| + |I_q| = \delta_1 + r$

ii.1) Assume that $|I_1| + |I_q| = r$. From Lemma 1, type 1 entries are scheduled in uL_1 slots during $[l\delta_1, f\delta_1]$. So, $u(\delta_1 - L_1)$ slots are available for scheduling type 2 entries in that interval. Since $I_1 = [k\delta_2, l\delta_1]$ and $I_q = [f\delta_1, (k+1)\delta_2]$ we have:

$$f\delta_1 = l\delta_1 + u\delta_1 \tag{10}$$

We consider two subcases:

a) $|I_q| > L_1$
b) $|I_q| \leq L_1$

ii.1.*a)* Let us assume that $|I_q| > L_1$. Then

$$f\delta_1 + L_1 < (k+1)\delta_2 \tag{11}$$

Let $\alpha = (k+1)\delta_2 - (f\delta_1 + L_1)$. Then type 2 entries can be scheduled in I_q for α slots. SIMPLE schedules type 1 entries for L_1 slots in $[(l-1)\delta_1, l\delta_1]$. It follows:

$$\begin{aligned}(l-1)\delta_1 + L_1 = \;& f\delta_1 - \delta_1 - u\delta_1 + L_1 < \;(k+1)\delta_2 - \delta_1 - u\delta_1 = \\ = \;& k\delta_2 + \delta_2 - \delta_1 - u\delta_1 = \;k\delta_2 - \delta_1 + r \;< k\delta_2\end{aligned}$$

from (10), (11) and $\delta_2 = u\delta_1 + r$. Then, all slots in I_1 are available for the scheduling of type 2 entries. Let $\beta = l\delta_1 - k\delta_2$. So, $\alpha + \beta$ represents the time slots available for scheduling of type 2 entries in I_1 and I_q:

$$\begin{aligned}
\beta + \alpha &= l\delta_1 - k\delta_2 + (k+1)\delta_2 - (f\delta_1 + L_1) = \\
&= l\delta_1 - k\delta_2 + k\delta_2 + \delta_2 - f\delta_1 - L_1 = \\
&= l\delta_1 + \delta_2 - l\delta_1 - u\delta_1 - L_1 = r - L_1
\end{aligned}$$

Hence, the time slots available for type 2 entries in $[k\delta_2, (k+1)\delta_2]$ are $u(\delta_1 - L_1) + (r - L_1)$. If $L_2 \leq \delta_2 - uL_1 - L_1$, deadline $(k+1)\delta_2$ is met.

*ii.*1.*b)* Let us assume that $|I_q| \leq L_1$. Then

$$f\delta_1 + L_1 \geq (k+1)\delta_2 \tag{12}$$

This means that no type 2 entries can be scheduled in interval I_q. Now, we establish the following two inequalities:

$$|I_1| = l\delta_1 - k\delta_2 \geq r - L_1.$$

Namely, there are more than $(r - L_1)$ slots in interval I_1, and $l\delta_1 - (r - L_1) > (l-1)\delta_1 + L_1$, meaning that type 1 entries are not scheduled in the last $(r - L_1)$ slots of interval I_1. In fact, from (10), (12) and $\delta_2 = u\delta_1 + r$, we have:

$$\begin{aligned}
l\delta_1 - k\delta_2 &= f\delta_1 - u\delta_1 - k\delta_2 \geq (k+1)\delta_2 - L_1 - u\delta_1 - k\delta_2 = \\
&= k\delta_2 + \delta_2 - L_1 - u\delta_1 - k\delta_2 = \delta_2 - u\delta_1 - L_1 = \\
&= r - L_1
\end{aligned}$$

Besides,

$$l\delta_1 - (r - L_1) > l\delta_1 - (\delta_1 - L_1) = (l-1)\delta_1 + L_1$$

So, in I_1 there are $(r - L_1)$ slots available for scheduling type 2 entries. Then, there is a total of $u(\delta_1 - L_1) + (r - L_1)$ slots available for type 2 entries in $[k\delta_2, (k+1)\delta_2]$. If $L_2 \leq u(\delta_1 - L_1) + r - L_1$, then deadline $(k+1)\delta_2$ is met.

*ii.*2) Let us now assume that $|I_1| + |I_q| = \delta_1 + r$. From Lemma 1, the number of subintervals I_i of length δ_1 in $[k\delta_2, (k+1)\delta_2]$ is $(u-1)$. Then, type 1 entries are scheduled in $(u-1)$ L_1 slots during $[l\delta_1, f\delta_1]$. So, there are at least $(u-1)(\delta_1 - L_1)$ slots available for type 2 entries.
Since $|I_1| + |I_q| = \delta_1 + r$, SIMPLE schedules type 1 entries for at most L_1 slots in I_1 and L_1 slots in I_q. There remain at least $(\delta_1 - L_1) + (r - L_1)$ slots for scheduling of type 2 entries. Thus, a total of

$$(u-1)(\delta_1 - L_1) + (\delta_1 - L_1) + (r - L_1) = u(\delta_1 - L_1) + (r - L_1)$$

slots are available for type 2 entries in $[k\delta_2, (k+1)\delta_2]$. If $L_2 \leq u(\delta_1 - L_1) + (r - L_1)$ and deadline $(k+1)\delta_2$ is met.

$\Rightarrow$
Assume that $L_2 > \delta_2 - uL_1 - L_1$. Let us consider interval $[0, \delta_2]$ and let $I_1, I_2, \cdots, I_q$ be the subintervals in $[0, \delta_2]$ between two consecutive deadlines. From Lemma 1, the

number of subintervals with length δ_1 is u. In each such subinterval, SIMPLE schedules type 1 entries for L_1 slots, so $(\delta_1 - L_1)$ slots are left available for type 2 entries. Since $|I_q| = r > L_1$, SIMPLE schedules type 1 entries for L_1 slots in interval I_q and type 2 entries for the remaining $r - L_1$ slots. Then, a total of $u(\delta_1 - L_1) + (r - L_1)$ slots are available for scheduling type 2 entries in $[0, \delta_2]$. Finally, if $L_2 > \delta_2 - uL_1 - L_1$ (see (9)), type 2 entries cannot be all scheduled in $[0, \delta_2]$ and deadline δ_2 cannot be met, a contradiction.

By exploiting this theorem, we are able to prove a feasibility condition for the general case with an arbitrary number of deadlines.

Theorem 2 *Let a matrix* $\frac{T}{D} = \left(\frac{t_{ijh}}{d_{ijh}}\right)$ *be given, with* $d_{ijh} \in \{\delta_1, \delta_2, \ldots, \delta_M\}$. *Without loss of generality, assume that:* $\delta_1 < \delta_2 < \ldots < \delta_M$. *If*

$$L_q \leq \delta_q - \sum_{p=1}^{q-1} [u_{pq} L_p + \min(r_{pq}, L_p)] \tag{13}$$

for $2 \leq q \leq M$, *where* $\delta_q = u_{pq}\delta_p + r_{pq}$, *then SIMPLE always generates a feasible schedule.*

Proof: We prove the theorem by induction. The case $M = 2$ has been established in Theorem 1. Let us assume that the theorem holds for a generic value $M = k$: we prove that it holds for $M = k+1$. Thus, if inequality (13) holds for $2 \leq q \leq k$, then SIMPLE always generates a feasible schedule. We show that, if inequality (13) holds for $2 \leq q \leq k+1$, then SIMPLE schedules matrix $\frac{T}{D}$, meeting all the deadlines.
Let us assume that SIMPLE does not meet a deadline. By the induction assumption, and by the scheduling strategy of SIMPLE, all deadlines $\delta_1, \delta_2, \ldots, \delta_k$ are met. Then, a deadline δ_{k+1} is not met. Note that type 1 entries must be scheduled for L_1 slots in each of the $m_{1,k+1}$ intervals of length δ_1 in δ_{k+1} intervals, plus at most $\min(r_{1,k+1}, L_1)$ other slots. Indeed, if $r_{1,k+1} > 0$, type 1 entries are scheduled one more time for $\min(r_{1,k+1}, L_1)$ slots. We can repeat the above argument for each type q entries, which must be scheduled at most for $m_{q,k+1}L_q + \min(r_{q,k+1}, L_q)$ slots, $2 \leq q \leq k$. Type $k+1$ entries must be scheduled for L_{k+1} slots. So, an upper bound to the sum of the entries to be scheduled during a δ_{k+1} interval is

$$\sum_{q=1}^{k} [m_{q,k+1}L_q + \min(r_{q,k+1}, L_q)] + L_{k+1}$$

Since we assumed that a deadline δ_{k+1} is not met, it follows:

$$\delta_{k+1} \leq \sum_{q=1}^{k} [m_{q,k+1}L_q + \min(r_{q,k+1}, L_q)] + L_{k+1}$$

i.e. inequality (13) does not hold for $q = k+1$, but this is a contradiction.

Note that in the worst case, RM behaves like SIMPLE, i.e. it does not schedule together entries with different priorities. Hence, the performance behavior of RM is not worst than that of SIMPLE. So, we can exploit the feasibility condition of SIMPLE to state a feasible condition for RM.

Theorem 3 *Let a matrix* $\frac{T}{D} = \left(\frac{t_{ijh}}{d_{ijh}}\right)$ *be given, with* $d_{ijh} \in \{\delta_1, \delta_2, \ldots, \delta_M\}$*. Without loss of generality, we assume that:* $\delta_1 < \delta_2 < \ldots < \delta_M$*.If*

$$L_q \leq \delta_q - \sum_{p=1}^{q-1}[m_{pq}L_p + \min(r_{pq}, L_p)] \tag{14}$$

for $2 \leq q \leq M$*, then RM always generates a feasible schedule.*

Proof: If condition (14) holds, SIMPLE always generates a feasible schedule (see Theorem 2). Since the performance behavior of RM is not worst than that of SIMPLE, RM is able to generate a feasible schedule, too.

4 Conclusions

In this paper, we derived the performance guarantee of Rate Monotonic algorithm, when applied to single-hop, multichannel communication systems, like crossbar switches or all optical, passive-star networks. The performance guarantee is obtained by using a simplified version of Rate Monotonic, called SIMPLE.

Many problems remain open in periodic real-time packet scheduling. Among them, the adoption of other objective functions, like jitter, and the soft deadlines case, namely when the missing of a limited number of deadlines is tolerated (like in many multimedia applications) are prominent.

References

[AMO92] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms and Applications.* Prentice Hall, New York, 1992.

[BaB94] P. Barcaccia and M. A. Bonuccelli, "A polynomial time optimal algorithm for time slot assignment in variable bandwidth systems." *ACM-IEEE Transactions on Networking* , Vol. 2, n.3 pp. 247–251, 1994

[BBD96] P. Barcaccia, M. A. Bonuccelli and M. Di Ianni,"Minimum length scheduling of precedence constrained messages in distributed systems." *EuroPar '96 Conference*, Lione, France, (August 1996); *Lecture Notes in Computer Science*, n.1024, pp. 594–601, Springer Verlag, Berlin.

[BCW81] G. Bongiovanni, D. Coppersmith, and C. K. Wong, "An optimal time slot assignment for a SS/TDMA system with variable number of transponders." *IEEE Transaction on Communication*, Vol.29, pp. 721–726, 1981.

[BC98a] M. A. Bonuccelli and M. C. Clò, "Scheduling of real time messages in optical broadcast-and-select networks" *manuscript, submitted for publication*, available from: http://www.di.unipi.it/ bonucce/HRTsporadico.ps.gz

[BC99] M. A. Bonuccelli and M. C. Clò, "EDD Algorithm Performance Guarantee for Periodic Hard-Real-Time Scheduling in Distributed Systems", IPPS/SPDP 1999, San Juan, Puerto Rico, 12-16th April 1999.

[BF97] A. A. Bertossi and A. Fusiello, "Rate Monotonic Scheduling for Hard-Real-Time Systems", *Europ. J. Operational Research*, 96, pp. 429-443, 1997.

[BGW91] M. A. Bonuccelli, I. S. Gopal, and C. K. Wong, "Incremental time slots assignment in SS/TDMA satellite systems." *IEEE Transaction on Communication*, Vol.37, pp. 870–874, 1989.

[BM96] M. S. Borella and B. Mukherjee, "Efficient scheduling of nonuniform packet traffic in a WDM/TDM local ligthwave network with arbitrary transceiver tuning latencies." *IEEE J. on Selected Areas on Communication*, Vol.14, pp. 923–934, June 1996.

[CCA96] H. Choi, H. A. Choi and M. Azizoglu, "Efficient scheduling of transmissions in optical broadcast networks." *IEEE/ACM Transaction on Networking*, Vol.4, pp. 913–920, Dec. 1996.

[GH97] J. Giles and B. Hajek, "Scheduling Multirate Periodic Traffic in a Packet Switch." *1997 Conference on Information Sciences and Systems at John Hopkins University*, available in postscript version at http://tesla.csl.uiuc.edu/ hajek/

[GBW83] I. S. Gopal, M. A. Bonuccelli, and C. K. Wong, "Scheduling in multibeam-satellites with interfering zones." *IEEE Transactions on Communication*, Vol.31, pp. 941–951, 1983.

[GGJ78] M. R. Garey, R. L. Graham, and D. S. Johnson, "Performance Guarantees for Scheduling Algorithms.", *Operations Research*. Vol. 26, n. 1, Gen-Feb. 1978, pp. 3–21.

[Hwa93] K. Hwang,*Advanced Computer Architecture*, McGraw-Hill, Inc., USA, 1993.

[Jain97] R. Jain, K. Somalwar, J. Werth, J. C. Browne, "Heuristics for the Scheduling I/O Operations " *IEEE Transactions on Parallel and Distributed Systems*, Vol. 8 , n. 3, pp. 310–320, Mar. 1997.

[JMI95] F. Jia, B. Mukherjee and J. Iness, "Scheduling variable-length messages in single-hop multichannel local ligthwave networks." *IEEE/ACM Transaction on Networking*, Vol.3, pp. 477–489, Aug. 1995

[KLR94] M. H. Klein, J. P. Lehoczky, and R. Rajkumar, "Rate-Monotonic Analysis for Real-Time Industrial Computing" *IEEE COMPUTER*, pp. 24–33, January 1994.

[Leh94] J. P. Lehoczky, "Real-Time Resource Management Techniques.", J. J. Marciniak, ed., *Encyclopedia of Software Eng.*, John Wiley and Sons, New York, 1994, pp. 1011–1020.

[LeW82] J. Leung and J. Whitehead, "On the Complexity of Fixed-Priority Scheduling of periodic Real-Time tasks.", *Performance Evaluation*, Vol. 2, n. 4, 1982, pp. 237–250.

[LiL73] C. L. Liu and W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment.", *J.ACM*.Vol. 20,n.1, Jan.1973, pp. 46–61.

[PhLi] I. R. Philp and J. W. Liu, "A Switch Scheduling Problem for Real Time Periodic Messages." Submitted to Telecommunications Systems Journal, available in postscript version at http://pertsserver.cs.uiuc.edu/papers

[PS94] G. R. Pieris and G. H. Sasaki, "Scheduling transmissions in WDM broadcast-and-select networks." *IEEE/ACM Transaction on Networking*, Vol.2, pp. 105–110, Apr. 1994.

[RoS96] G. N. Rouskas and V. Sivaraman, "On the design of optimal TDM schedules for broadcast WDM networks with arbitrary transceiver tuning latencies." *Proc. IEEE INFOCOM'96*, pp. 1217–1224, 1996.

[SSDB95] J. A. Stankovic, M. Spuri, M. Di Natale and G. C. Buttazzo, "Implications of Classical Scheduling Results for Real-Time Systems." *IEEE Computer*, Vol. 28, n. 6, pp. 16-25, 1995.

[THW+96] 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. IEEE Real Time Systems Symposium*, 1996, pp. 175–184.

Appendix

A) Consider the following traffic and deadline matrices:

$$T = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 2 \\ 3 & 0 & 1 \end{bmatrix} \qquad D = \begin{bmatrix} 2 & 2 & 0 \\ 0 & 3 & 3 \\ 6 & 0 & 6 \end{bmatrix}$$

According to the definition, we can consider the composition $\frac{T}{D}$:

$$\frac{T}{D} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 0 & 1/3 & 2/3 \\ 3/6 & 0 & 1/6 \end{bmatrix}$$

In this case, $\Delta = 6$ and we have to find a feasible schedule over time interval $[0, 6]$. If we schedule the boxed entries in time slot 1:

$$\begin{bmatrix} 1/2 & \boxed{1/2} & 0 \\ 0 & 1/3 & \boxed{2/3} \\ \boxed{3/6} & 0 & 1/6 \end{bmatrix}$$

then matrix $\frac{T}{D}$ becomes:

$$\frac{T'}{D'} = \begin{bmatrix} 1/1 & 0 & 0 \\ 0 & 1/2 & 1/2 \\ 2/5 & 0 & 1/5 \end{bmatrix}$$

A complete and feasible schedule of the given matrices $\frac{T}{D}$ is the following:

$$\underset{t=0}{\begin{bmatrix} 1/2 & \boxed{1/2} & 0 \\ 0 & 1/3 & \boxed{2/3} \\ \boxed{3/6} & 0 & 1/6 \end{bmatrix}} \qquad \underset{t=1}{\begin{bmatrix} \boxed{1/1} & 0 & 0 \\ 0 & \boxed{1/2} & 1/2 \\ 2/5 & 0 & \boxed{1/5} \end{bmatrix}}$$

$$\underset{t=2}{\begin{bmatrix} \boxed{1/2} & 1/2 & 0 \\ 0 & 0 & \boxed{1/1} \\ 2/4 & 0 & 0 \end{bmatrix}} \qquad \underset{t=3}{\begin{bmatrix} 0 & \boxed{1/1} & 0 \\ 0 & 1/3 & \boxed{2/3} \\ \boxed{2/3} & 0 & 0 \end{bmatrix}}$$

$$\underset{t=4}{\begin{bmatrix} 1/2 & \boxed{1/2} & 0 \\ 0 & 1/2 & \boxed{1/2} \\ \boxed{1/2} & 0 & 0 \end{bmatrix}} \qquad \underset{t=5}{\begin{bmatrix} \boxed{1/1} & 0 & 0 \\ 0 & \boxed{1/1} & 0 \\ 0 & 0 & 0 \end{bmatrix}}$$

B)

$$\begin{bmatrix} \boxed{1/4}, 1/6 & 0 & 1/6 & 1/4, 1/12 \\ 1/12 & 1/6, 1/12 & \boxed{2/4} & 0 \\ 0 & 0 & 1/12 & 0 \\ 0 & 3/6 & 0 & \boxed{1/4}, 1/6, 1/12 \end{bmatrix}$$

$$\begin{bmatrix} 0, 1/6 & 0 & 1/6 & \boxed{1/4}, 1/12 \\ 1/12 & 1/6, 1/12 & \boxed{1/4} & 0 \\ 0 & 0 & 1/12 & 0 \\ 0 & \boxed{3/6} & 0 & 0, 1/6, 1/12 \end{bmatrix}$$

$$\begin{bmatrix} 0, \boxed{1/6} & 0 & 1/6 & 0, 1/12 \\ 1/12 & \boxed{1/6}, 1/12 & 0 & 0 \\ 0 & 0 & \boxed{1/12} & 0 \\ 0 & 2/6 & 0 & 0, \boxed{1/6}, 1/12 \end{bmatrix}$$

$$\begin{bmatrix} 0, 0 & 0 & \boxed{1/6} & 0, 1/12 \\ \boxed{1/12} & 0, 1/12 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \boxed{2/6} & 0 & 0, 0, 1/12 \end{bmatrix}$$

$$\begin{bmatrix} \boxed{1/4}, 0 & 0 & 0 & 1/4, 1/12 \\ 0 & 0, 1/12 & \boxed{2/4} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1/6 & 0 & \boxed{1/4}, 0, 1/12 \end{bmatrix}$$

$$\begin{bmatrix} 0, 0 & 0 & 0 & \boxed{1/4}, 1/12 \\ 0 & 0, 1/12 & \boxed{1/4} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \boxed{1/6} & 0 & 0, 0, 1/12 \end{bmatrix}$$

$$\begin{bmatrix} 0, \boxed{1/6} & 0 & 1/6 & 0, 1/12 \\ 0 & \boxed{1/6}, 1/12 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 3/6 & 0 & 0, \boxed{1/6}, 1/12 \end{bmatrix}$$

$$\begin{bmatrix} 0, 0 & 0 & \boxed{1/6} & 0, 1/12 \\ 0 & 0, 1/12 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \boxed{3/6} & 0 & 0, 0, 1/12 \end{bmatrix}$$

$$\begin{bmatrix} \boxed{1/4}, 0 & 0 & 0 & 1/4, 1/12 \\ 0 & 0, 1/12 & \boxed{2/4} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 2/6 & 0 & \boxed{1/4}, 0, 1/12 \end{bmatrix}$$

$$\begin{bmatrix} 0, 0 & 0 & 0 & \boxed{1/4}, 1/12 \\ 0 & 0, 1/12 & \boxed{1/4} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \boxed{2/6} & 0 & 0, 0, 1/12 \end{bmatrix}$$

$$\begin{bmatrix} 0, 0 & 0 & 0 & 0, \boxed{1/12} \\ 0 & 0, 1/12 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \boxed{1/6} & 0 & 0, 0, 1/12 \end{bmatrix}$$

$$\begin{bmatrix} 0, 0 & 0 & 0 & 0, 0 \\ 0 & 0, \boxed{1/12} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0, 0, \boxed{1/12} \end{bmatrix}$$

C) **Algorithm RM:**

$k \leftarrow 1$; $succ \leftarrow true$; $\tau \leftarrow 0$;

$\Delta \leftarrow$ l.c.m.(d_{ijh}); $T_2 \leftarrow T$; $D_2 \leftarrow D$; $list \leftarrow Sort[D]$;

sort all the deadline occurrences, until Δ, and put them in *occur*;

while $\tau < \Delta$ **and** *succ* **do**

begin

  $T_1 \leftarrow T$; $S_k \leftarrow 0$;

  while $T_1 \neq 0$ **do**

  begin

    scan *list* from the first entry,

    until you find one with $t^1_{ijh} > 0$;

    set s^k_{ijh} to t^1_{ijh};

    set to zero row i and column j of T_1;

  end;

  find the minimum positive value sk_{min} in S_k;

  set all non zero entries of S_k equal to sk_{min};

  $T_2 \leftarrow T_2 - S_k$; $D_2 \leftarrow D_2 - sk_{min}$;

  $\tau \leftarrow \tau + sk_{min}$;

  if $\tau = \Delta - 1$

    then Stop with $succ = true$

      and $\{S_1, S_2, \ldots, S_v\}$ is the schedule for T and D

    else if $d_{ijh} - t_{ijh} \geq 0$ for each $t_{ijh} > 0$

      then $k \leftarrow k + 1$

      else $succ = false$;

  if *occur*$\neq \emptyset$ **and** $\tau =$*occur*[1] **then**

  begin

    find d_{ijh} such that τ mod $d_{ijh} = 0$;

    set $t^2_{ijh} = t_{ijh}$, $d^2_{ijh} = d_{ijh}$;

    occur $\leftarrow$ *occur* - *occur*[1];

  end

end;