Two Processor Scheduling with Large Communication Delays *

(EXTENDED ABSTRACT)

F. Afrati ‡ E. Bampis § L. Finta ¶ I. Milis ‡†

‡NTUA, Department of Computer Science, 9, Heroon Polytechniou str., 15773, Athens, Greece
§LaMI, Université d'Evry, Boulevard des Coquibus, 91025 Evry Cedex, France
¶INRIA-Sophia Antipolis, B.P. 93, 06902 Sophia-Antipolis Cedex, France

Abstract: The problem of scheduling partially ordered tasks on two identical processors is considered. It is assumed that tasks have unit execution time, and the arcs of the task graph are associated with integral identical communication delays greater or equal to two time units. We prove that the problem of minimizing the schedule length is NP-hard even for binary trees and it becomes polynomial for complete binary trees.

Keywords: Scheduling, Parallelism, Makespan, Precedence Constraint, Trees, Large Communications, Complexity, Optimal Algorithm.

1 Introduction

Scheduling partially ordered tasks with unit execution times on two identical processors to minimize the latest finishing time (makespan) has been extensively studied during the last three decades. The main motivation behind this area has been the programming of parallel systems, one of the most important challenges facing computer science today.

It is well known that when communication delays are considered negligible then the problem is polynomially solvable [**?**,] However, in parallel systems there is a communication delay, between the time some information is produced at a processor and the time it can be used by another. This delay plays a fundamental role in the performance of a parallel system [11], and was indentified by Bell [2] as the single most important parameter of parallel architectures.

Due to this fact, there is recently a strong interest on the scheduling problem in the presence of communication delays. A large of the work carried out during last years is concentrated on the unit communication delays case, i.e. when communication delays are considered indentical of unit

*This work has been partially supported by the French-Greek bilateral exchange program PLATON and the GDR-PRS "Ordonnancement pour le parallélisme" program of the French government.

†Fellow of the European Commission (Program T.M.R.)

length. For a thorough survey of the results on this case the reader is referred to [3] [17]. Still the major question concerning the complexity of the two-processor scheduling problem with unit execution-time tasks and unit communications delays remains open. On the other hand polynomial algorithms yielding optimal solutions for special ordered tasks are known, especially trees [15] [10] [13] [16], interval orders [13] [1] and series-parallel graphs (SP1) [6].

However, from the practical point of view large communication delays are more close to real parallel architectures. Indeed, for todays parallel systems the communication delay for the transmission of an elementary data is usually larger than unit elementary processor-steps. As a function of the number m of processors c can range, depending on the architecture, from m (on a ring) to $m^{\frac{1}{2}}$ (on a grid) to $\log m$ (on a hypercube). For actual computers values of c range from small constants to millions (for slow clusters).

Despite this fact no many results are known for the large communication case. No results are known for the two-processor scheduling problem and all known results assume that an infinite number of processors is available:
Papadimitriou and Yannakakis in [12] have shown that sheduling general dags remains NP-hard in the strong sense even if the tasks can be duplicated. For the same problem they also shown an approximation algorithm with worst case ratio two, while Jung, Kirousis and Spyrakis in [4] introduced an $O(n^{c+1})$ dynamic programming optimal algorithm and shown some lower bounds for the schedule length of complete binary trees.
Jakoby and Reischuk [5] have proven that scheduling binary trees is NP-hard and it becomes polynomially solvable for complete binary (k-ary) trees. This last polynomial result is also independently proven by Gao, Rosenberg and Sitaraman [7].

In this paper we deal with the two-processor scheduling problem in the presence of large identical communication delays (≥ 2). We prove that in contrast with unit communication case the problem becomes NP-hard even for the extremely special class of binary trees. Thus, we give a border between polynomiality and NP-completeness of the sheduling problems as the communication delay increases. Furthermore, we present a polynomial algorithm for complete binary trees.
Our results prove that the complexity behavior of the two-processor case is analogous to the case where an infinite number of processors are available. The paper is organised as following: We give below the scheduling problem model. In the next section we present the NP-completeness result and in Section 3 we the polynomial algorithm.

1.1 Problem model and notation

In this paper we assume the following model incorporating communication delays first introduced by Papadimitriou and Yannakakis [12]: There is a set $P = \{p_1, p_2, ..., p_m\}$ of identical processors and a set $V = \{1, 2, ..., n\}$ of partially ordered tasks. The set of tasks is usually described by a directed acyclic graph (dag) $\mathcal{G} = (V, E)$, referred as *task graph.* Tasks have unit execution times (UET), denoted by $p_j = 1$, and their execution is subject to precedence constraints and communication delays; whenever two communicating tasks i, j, with $(i,j) \in E$, are scheduled on different processors an identical communication delay $c_{ij} = c$ is introduced. A schedule is defined as a function $\sigma : V \to I\!N_+ \times P$, i.e. $\sigma(u) = (t_u, p_u)$ where t_u is the starting time and p_u the processor on which task u is scheduled. A schedule is *feasible* if:

- for all $u, v \in V$, $u \neq v$ implies $(t_u, p_u) \neq (t_v, p_v)$;
- if $(u,v) \in E$ then $t_u + 1 + c \times \mathbf{1}(p_u \neq p_v) \leq t_v$,

where $\mathbf{1}(.)$ is the indicator function.

The reverse function $\sigma^{-1}(t,p)$ gives the task that is executed on processor p during the time interval $[t, t+1]$. A processor is said to be *idle* in time slot t if no task is executed during this time slot. In this case we consider this processor executing a fictive task, labeled 0. By M we denote the makespan (length) of a schedule, that is the last time unit some task is executed on any processor, i.e:

$$M = \max\{\ t+1 \mid \sigma^{-1}(t, p_i) \neq 0, \quad i = 1, 2, ..., m\}.$$

According to the three-field notation scheme introduced in [9] and extended in [18] for scheduling problems with communication delays, our problem is denoted as

$$P2 \mid prec,\ p_j = 1,\ c_{i,j} = c(n) \mid C_{\max}$$

2 NP-completeness result

To prove that $P2|$ binary trees, $p_j = 1, c_{ij} = c|C_{max}$ is NP-hard, we give a reduction from the following special case of the well known NP-complete problem EXACT-COVER BY 3-SETS (X3C), that we call X3C1 [8]:

INSTANCE: A set $U = \{h_1, h_2, \ldots, h_{3m}\}$ and a family $F = \{S_1, S_2, \ldots, S_k\}$, of $k \leq 3m$ subsets of U such that $S_t = \{h_i, h_j, h_v\}$, $1 \leq t \leq k$, and every element of U belongs to no more than three elements of F.

QUESTION: Are there m elements of F whose union is exactly U?

For every instance of X3C1, we shall construct an instance of $P2|$ binary trees, $p_j = 1, c_{ij} = c|C_{max}$. We choose first constant integers G, G' and a such that

$$G >> a >> G' >> m^3 \tag{1}$$

and we construct a binary tree in the in the following way:

For every $i = 1, 2, ..., 3m, 3m+1$ we consider a chain of length $H_i = a(m^3 + i)$. Each chain H_i corresponds to an element h_i of U exept H_{3m+1} for which the corresponding h_i is not defined. For

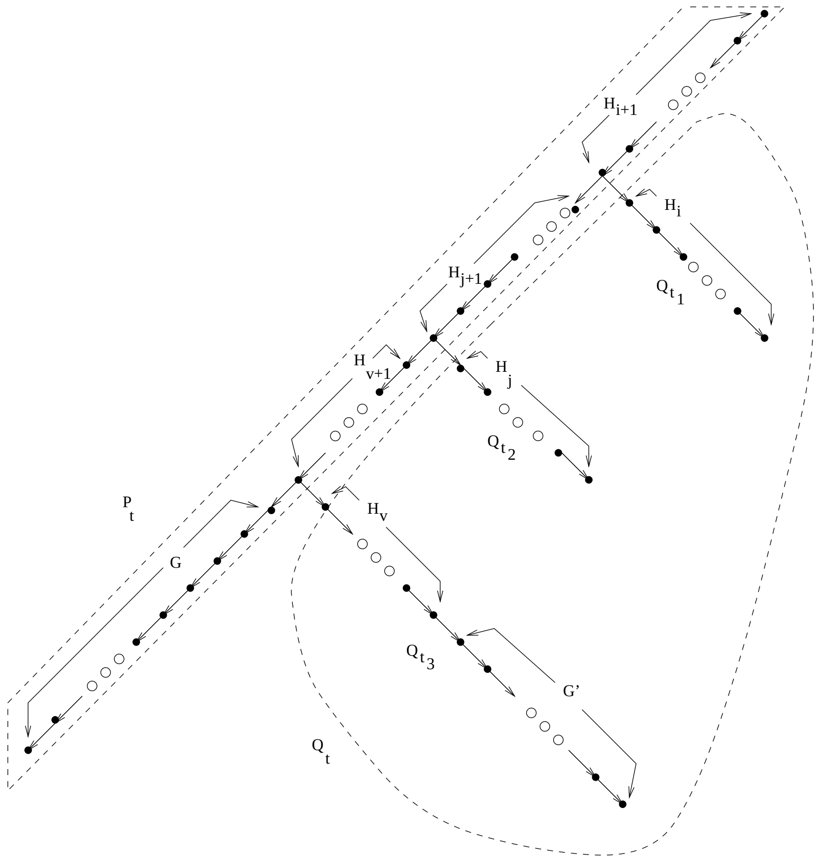

Figure 1: *The subtree T_t corresponding to the element $S_t = \{h_i, h_j, h_v\}$ of F.*

every element $S_t = \{h_i, h_j, h_v\}$ of F (w.l.o.g. we assume that $i > j > v$), we construct a subtree T_t as shown in Figure 1. The lengths of T_t's chains are depicted in the figure.

Given the subtrees T_t, $t = 1, \ldots, k$, we construct the tree $\mathcal{T}$ as shown in Figure 2. Let us denote by $\mathcal{B}$ the total number of tasks in all subtrees T_t, $t = 1, \ldots, k$, i.e.

$$\mathcal{B} = \sum_{t=1}^{k} |T_t| = kG + kG' + \sum_{\forall S_t = \{h_i, h_j, h_v\} \in S} (H_i + H_j + H_v + H_{i+1} + H_{j+1} + H_{v+1}).$$

Then we consider the following instance of the $P2|$ binary trees, $p_j = 1, c_{ij} = c|C_{max}$ scheduling problem:

Can we schedule $\mathcal{T}$ with communication delay

$$c = \mathcal{B} + G - 2(\sum_{i=1}^{3m} H_i + mG') - H_{3m+1},$$

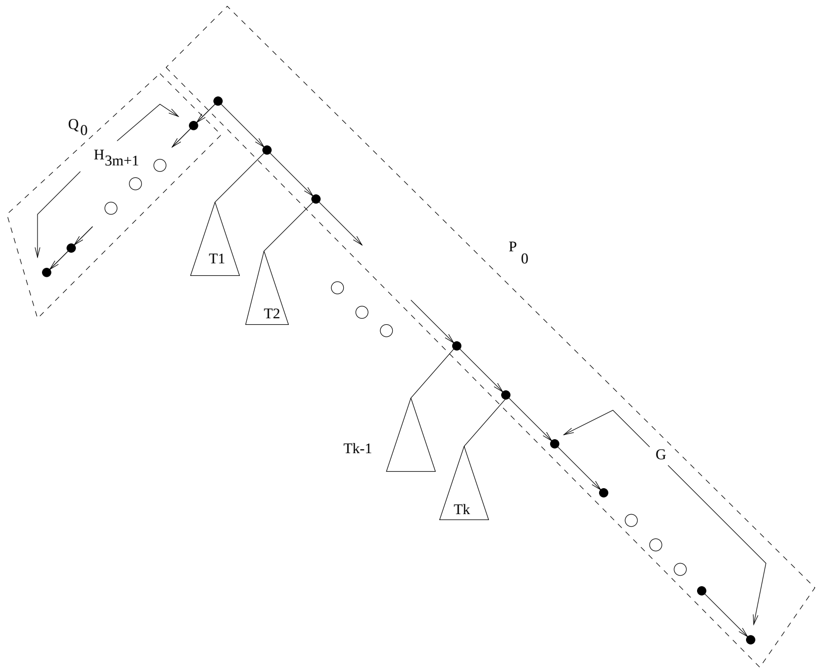

Figure 2: *The tree $\mathcal{T}$ corresponding to the instance of X3C.*

on two identical processors p_1 and p_2 within deadline

$$D_m = k + 1 + \mathcal{B} - (\sum_{i=1}^{3m} H_i + mG') + G?$$

It is easy to verify that if X3C1 has a positive answer, then there exists a feasible schedule $\mathcal{S}$ of $\mathcal{T}$ on two indentical processors such that $C_{\max}(\mathcal{S}) \leq D_m$. Indeed, let, w.l.o.g., $F^* = \{S_1, S_2, \ldots, S_m\}$ be a solution of X3C1, i.e. $|F^*| = m$, $F^* \subset F$ and $\bigcup_{i=1}^{m} S_i = U$. Let us also denote by $T_1, \ldots, T_m$ the subtrees of $\mathcal{T}$ corresponding to the elements of F^*. Then, a valid schedule is the following:

Processor p_1, starts at time 0 by executing the root of the tree. Then it executes the first k tasks of the path P_0 of $\mathcal{T}$ and then the chains $H_{i+1}, H_{j+1}, H_{v+1}$ of every P_t, $1 \leq t \leq m$, in decreasing order of their lengths i.e. $H_{3m+1}, H_{3m}, \ldots, H_2$. These $3m$ chains can be executed in that way since by the construction of $\mathcal{T}$, after the execution of some chain, say H_{i+1} of some P_t, $1 \leq t \leq m$, the chain H_i of some $P_{t'}$, $1 \leq t' \leq m$, is always available. Processor p_1 finishes by executing the remaining G tasks of every path P_t, $t = 1, \ldots, m$, the tasks of the remaining subtrees T_t, of $\mathcal{T}$, $t = m+1, \ldots, k$, and the last G tasks of P_0.

Processor p_2 starts executing at time $c + k + 1$ the H_{3m+1} tasks of Q_0 and then the tasks of the branches $Q_{t_1}, Q_{t_2}, Q_{t_3}$ of all trees $T_1, \ldots, T_m$, in decreasing order of their lengths i.e. H_{3m}, H_{3m-1}, $\ldots, H_1$. This is always possible since the execution of chain H_{l+1} of P_t, $t = 1, \ldots, m$, on processor

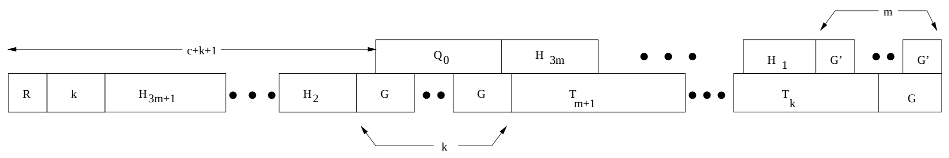

Figure 3: *A feasible schedule (R represents the root of the $\mathcal{T}$, and k the next k tasks in path P_0.)*

p_1 allows the execution of the corresponding branch of length H_l of Q_t on processor p_2, c time units after the end of the execution of the last task of H_{l+1} on p_1. Finally, it executes the remaining G' tasks in the end of the Q_{t_3}'s branches of every T_t, $t = 1, \ldots, m$ (the order is not important). Figure 3 illustrates the Gantt-chart of such a schedule.

Let us now show that if $\mathcal{T}$ can be scheduled within time D_m, then X3C1 has a solution F^*. We proceed by proving a series of claims concerning such a schedule:

Claim 1: Communications can be afforded only from processor p_1 to processor p_2.

Proof Notice that processor p_2 cannot start execution before time $c + 1$. With respect to the chosen value of c and the definition of D_m, it is clear that every schedule that contains a communication transfer from p_1 to p_2 as well as vice versa has a length of at least $2c + 3 > D_m$ time units, which is a contradiction to the assumption of $C_{max}(\mathcal{S}) \leq D_m$. Due to the symmetry, we can assume that the root of $\mathcal{T}$ is assigned to processor p_1. This concludes the proof of Claim 1. ∎

Claim 2: Processor p_2 may remain idle for at most k time units after time unit $c + 1$.

Proof If this is not the case, then processor p_1 has to execute more than D_m tasks and thus, the given deadline cannot be satisfied. This is a contradiction to the assumptions above. ∎

Claim 3: All the tasks of every path P_i, $0 \leq i \leq k$ are executed by processor p_1. Therefore, p_2 executes only tasks of Q_i's.

Proof By Claim 1 only one communication is allowed in any path from the root of $\mathcal{T}$ to a leaf. Thus, if a task of some P_i, $0 \leq i \leq k$, is executed on p_2, then all successors of this task must be executed on p_2 as well. Since all paths P_i end up with a chain of length G, at least $c + G > D_m$ tasks have to be executed by processor p_2. This stands in contradiction to our assumptions. ∎

Claim 4: Processor p_2 starts by executing at least $(H_{3m+1} - k)$ tasks of Q_0.

Proof Processor p_2, by Claim 3, executes only tasks of Q_i's and by Claim 2, must start before time $c + 1 + k$. It is clear that p_2 can not start by this time executing tasks of some Q_{t_1} since in this case H_{i+1} tasks of the corresponding path P_t, have to finish their execution on p_1 by time k, a contradiction since $a >> m$. However, p_1 can execute at most k tasks of Q_0. By Claim 1, a communication from p_2 to p_1 is not allowed and therefore p_2 executes at least $(H_{3m+1} - k)$ tasks of Q_0. ∎

Let us now call *stage* the time during which processor p_1 executes the tasks of a chain of length H_i of some P_t.

Claim 5: At every stage i, processor p_1 has to execute more tasks than at stage $i+1$.

Proof Clearly p_2 must execute at most $D_m - (c+1) = \sum_{i=1}^{3m+1} H_i + mG' + k$ tasks, and at least, by Claim 2, $\sum_{i=1}^{3m+1} H_i + mG'$ tasks. In order to prove this claim, we shall argue on the potential work of p_2. In fact we shall prove that processor p_2 can execute E tasks, $\sum_{i=1}^{3m+1} H_i + mG' \leq E \leq \sum_{i=1}^{3m+1} H_i + mG' + k$, only by executing in a specific order the Q_{t_i}'s that correspond to a solution of X3C1.

Let us consider the first stage. Let H_{i_1} be the length of the chain executed by p_1. Then p_2 could execute (c times after) a chain of length at most $H_{i_1 - 1}$. Clerly this chain is available because of the construction of $\mathcal{T}$. Now, what p_1 can execute at stage 2, in order to prepare some work for p_2 without leaving processor p_2 idle for more than k time units, is at most:

$$x_1 = H_{3m+1} + H_{i_1-1} - H_{i_1} = H_{3m+1} - a = H_{3m},$$

tasks. If $i_1 \neq 3m+1$ then $H_{i_1-1} < H_{3m}$, otherwise p_2 can execute $H_{i_1-1} = H_{3m}$ tasks.

Let H_{i_2} be the length of the chain executed by p_1 at stage 2. So the induced potential work for p_2 is a chain of length at most H_{i_2-1}. As before, what p_1 could execute in order to prepare some work for p_2 is a chain of length at most:

$$x_2 = H_{3m+1} + H_{i_1-1} + H_{i_2-1} - H_{i_1} - H_{i_2} = H_{3m+1} - 2a = H_{3m-1}.$$

If $i_2 \neq 3m$, then $H_{i_2-1} < H_{3m-1}$, otherwise p_2 can execute $H_{i_2-1} = H_{3m-1}$ tasks.

In general in the s^{th}, $s \leq 3m$, stage, what p_1 can execute is at most:

$$x_s = H_{3m+1} + \sum_{j=1}^{s} H_{i_j-1} - \sum_{j=1}^{s} H_{i_j} = H_{3m+1} - sa.$$

If $i_s \neq 3m - s + 2$, then $H_{i_s-1} < H_{3m-s+1}$, otherwise p_2 can execute $H_{i_s-1} = H_{3m-s+1}$ tasks.

If at some stage, say s, we have $H_{i_s-1} < H_{3m-s+1}$, then p_2 will never have potential work equal to $\sum_{i=1}^{3m+1} H_i + mG'$ without being idle for more than $c+1+k$ time units. Since for every i, $mG' << H_i$, a chain with H_{3m-s+1} tasks executed out of its order will create at least a idles on p_2. This completes the proof of the claim. ∎

Claim 6: Processor p_2 executes the tasks of exactly m Q_t's.

Proof By Claim 5, processor p_2 executes initially $\sum_{i=1}^{3m+1} H_i$ tasks without remaining idle for more than $c+1+k$ idle times. Since $mG' << H_i$, $1 \leq i \leq 3m$, p_2 has not the time to execute any other Q_{t_i} and therefore the only tasks processor p_2 can execute are the last parts of the branches started until the $3m^{th}$ stage. ∎

By Claim 5, in order to execute on p_2 $\sum_{i=1}^{3m+1} H_i + mG'$ tasks, X3C1 must have a solution F^* corresponding to the m Q_t's that processor p_2 has to execute.

Thus we can state the following theorem.

Theorem 1 *Finding an optimal schedule for $P2|$ binary trees, $p_j = 1, c_{ij} = c|C_{max}$ is NP-hard in the strong sense.*

3 Polynomial time algorithm for complete trees

Let us consider a complete binary out-tree (c.b.t.) of height h (the root of the tree has a height value of 1) with $n = 2^h - 1$ nodes, denoted by T_h, to be scheduled on two processors p_1 and p_2. By convention we assume that p_1 executes the root of the tree. By M_{p_i}, $i = 1, 2$, we denote the last time unit some task is executed on processor p_i, i.e. $M_{p_i} = \max\{ t \mid \sigma^{-1}(t, p_i) \neq 0\}$ for $i = 1, 2$. A schedule is called *balanced* if $|M_{p_1} - M_{p_2}| \leq 1$.

The key point of an optimal schedule is the number of communications allowed in a path from the root of the tree to a leaf. Roughly speaking we distinguish three cases depending on the magnitude of c with respect to n:

- For "long" communication delays no communication is allowed.
- For "medium" communication delays only one way communications, i.e. from p_1 to p_2, are allowed.
- For "small" communication delays both ways communications, i.e. from p_1 to p_2 and from p_2 to p_1, are allowed.

For long communications delays we are in the extreme case of a sequential schedule:

Lemma 1 *If $c \geq 2^h - 1 - h$, then the sequential schedule is optimal.*

Proof. It is easy to observe that executing some task on p_2 introduces a communication delay on some path from the root to a leaf. The resulting schedule is of length $M \geq h + c \geq h + 2^h - 1 > n = 2^h - 1$. ∎

A lower bound M^l for the length of any schedule using both processors can be obtained by considering a balanced schedule with the smallest number of idle time units. It is clear that processor p_2 can start execution not before time $c+1$. Since this also is a lower bound on the number of idle time units, we obtain:

$$C_{max}(\mathcal{S}) \geq M^l = \left\lceil \frac{n+c+1}{2} \right\rceil = \left\lceil \frac{2^h+c}{2} \right\rceil = 2^{h-1} + \left\lceil \frac{c}{2} \right\rceil .$$

In the following, by $x_h, x_{h-1}, ..., x_2, x_1$ we denote the leftmost path in a c.b.t. from its root, identified by x_h, to the leftmost leaf, identified by x_1. By y_i we denote the brother of task x_i, $1 \leq i \leq h-1$, i.e. x_i and y_i are the left and the right child, respectively, of x_{i+1}.

We consider first the case of "small" and even c's. We construct a *two ways communication schedule*, i.e. a schedule where there are communications from p_1 to p_2 and from p_2 to p_1.
The algorithm below constructs such an optimal schedule.

procedure SchTwoWaysCom (*c.b.t. of height* h, *communication delay* c *even*);
begin
Find the greatest value h_1, $1 \leq h_1 \leq h-2$, such that $c/2 \geq 2^{h_1} - 1$
Let $i = 1$
Let $M_{p_2} = c + 2^{h-1} - (2^{h_1} - 1)$ and $M^l = \left\lceil \frac{2^h+c}{2} \right\rceil$
While $M_{p_2} > M^l$ **do**
  Let $i = i + 1$
  Find the greatest h_i, $1 \leq h_i \leq h_{i-1}$, such that $M_{p_2} - (2^{h_i} - 1) \geq M^l$
  Let $M_{p_2} = M_{p_2} - (2^{h_i} - 1)$
enddo
Schedule on p_2 the path $x_{h-1}, ..., x_{h_i+1}$ in consecutive time units starting from time unit $c+2$.
Schedule on p_1 as soon as possible the subtrees rooted in y_{h_j}, $1 \leq j < i$.
If $h_i = h_{i-1}$ **then** schedule on p_1 as soon as possible the subtree rooted in x_{h_i}
  else schedule on p_1 as soon as possible the subtree rooted in y_{h_i}
Schedule on p_1 the tasks of the subtree rooted in y_{h-1}.
Schedule on p_2 the rest tasks of the subtree rooted in x_{h-1}.
end

Lemma 2 *The algorithm SchTwoWaysCom constructs an optimal schedule for even communication delays $c \leq 2^{h-2} - 2$.*

Proof. The aim of algorithm SchTwoWaysCom is to construct a schedule of length equal to the lower bound M^l. To this end, the subtree rooted in y_{h-1} is scheduled on p_1. The subtree rooted in x_{h-1} is scheduled on p_2, but some of its subtrees (those rooted in $y_{h_1}, y_{h_2}, ..., y_{h_i}$) containing in total $c/2$ tasks, are scheduled on p_1 in order to have a balanced schedule.
It is enough to prove that the subtree rooted in y_{h_1} is scheduled on p_1 not later than time unit $2^{h-1}+1$, since 2^{h-1} is the last time unit occupied by the root and the tasks of the subtree rooted in y_{h-1}, executed in a non-idle manner. If the communication delay is $c = 2^{h-2} - 2$, then the subtree

rooted in y_{h_1} has height $h-3$, i.e. $h_1 = h-3$. Moreover, it is the only subtree ($i=1$) that must be executed on p_1. Since x_{h-2} is executed on p_1 at time unit $c+3$, y_{h-3} is available for execution on p_1 in time unit $2c+4$, and we have $2c+4 \leq 2^{h-1}+1$ that is true.
For communications $c < 2^{h-2}-2$ the same argument holds and allows us to conclude. ∎

Corollary 1 *The algorithm SchTwoWaysCom constructs a schedule longer than M^l+1 for even communication delays $c \geq 2^{h-2}$.*

Corollary 2 *Any algorithm with two ways communication constructs a schedule longer than M^l+1 for even communication delays $c \geq 2^{h-2}$.*

Proof. Any two ways communication algorithm either do not start execution on p_2 in time unit $c+2$ or has p_1 idle during at least one time unit (before executing the subtree rooted in y_{h_1}). ∎

We consider next the case of "medium" c's. We construct a *one way communication schedule*, i.e. a schedule where there are communications only from p_1 to p_2.
The algorithm below constructs such an optimal schedule.

procedure SchOneWayCom (*c.b.t. of height h, communication delay c*);
begin
Find the greatest value h_1, $1 \leq h_1 \leq h-1$, such that $h - h_1 + c + 2^{h_1} - 1 \leq 2^h - 1 - (2^{h_1}-1)$
Let $i = 1$
Let $M_{p_1} = 2^h - 1 - (2^{h_1}-1)$ and $M_{p_2} = h - h_1 + c + 2^{h_1} - 1$
While $M_{p_1} > M_{p_2} + 1$ **do**
    Let $i = i+1$
    Find the greatest h_i, $1 \leq h_i \leq h_{i-1}$, such that $M_{p_2} + 2^{h_i} - 1 \leq M_{p_1} - (2^{h_i}-1)$
    Let $M_{p_1} = M_{p_1} - (2^{h_i}-1)$ and $M_{p_2} = M_{p_2} + 2^{h_i} - 1$
enddo
Schedule on p_1 the path $x_h, ..., x_{h_i+1}$ in the first $h-h_i$ time units.
Schedule on p_2 as soon as possible the subtrees rooted in y_{h_j}, $1 \leq j < i$.
If $h_i = h_{i-1}$ **then** schedule on p_2 as soon as possible the subtree rooted in x_{h_i}
    else schedule on p_2 as soon as possible the subtree rooted in y_{h_i}
Schedule on p_1 the rest tasks of the initial tree.
end

Lemma 3 *The algorithm SchOneWayCom constructs an optimal schedule for:*
- *Odd communication delays $c < 2^h - 1 - h$, and*
- *Even communication delays $2^{h-2} \leq c < 2^h - 1 - h$.*

Proof.
The length of the constructed schedule by algorithm SchOneWayCom is $M = \left\lceil \frac{2^h + c + h - h_1}{2} \right\rceil$. We distinguish between two cases depending on the magnitude of c:

(i) If $2^{h-1}-1 \leq c < 2^h - 1 - h$ it is easy to observe that any two ways communication schedule is longer than the one way communication schedule produced by algorithm SchOneWayCom since there is at least one path in the tree containing two communications, i.e. $M_{SchTwoWaysCom} \geq h + 2c \geq \left\lceil \frac{2^h+c+h}{2} \right\rceil > M$.

Thus we have only to prove that the algorithm constructs the shortest schedule with one way communications. Consider first a one way communication algorithm such that the first subtree to be executed on p_2 is of height $h'_1 > h_1$. Clearly the obtained schedule will be unbalanced and longer than the one produced by our algorithm. Consider next the case where the first subtree to be executed on p_2 is of height $h'_1 < h_1$. Then processor p_2 starts execution in time unit $h - h'_1 + c$ that is latter than in algorithm SchOneWayCom and therefore the obtained schedule is longer.

(ii) If $c < 2^{h-1} - 1$ and c is odd, then the first subtree to be executed on p_2 is of height $h_1 = h - 2$. Hence, the length of the schedule is $\left\lceil \frac{2^h+c+1}{2} \right\rceil = M^l$, that is optimal, since c is odd.

If $2^{h-2} \leq c < 2^{h-1} - 1$ and c is even, then the length of the schedule is is equal to $M^l + 1$. Using Corollary 2 we conclude that this schedule is optimal. ∎

Combining Lemmas 1, 2 and 3 we obtain the next theorem:

Theorem 2 *A complete binary tree of height h, with communication delay c, can be scheduled optimally on two processors in $O(n \log n)$ time.*

References

[1] H. Ali, H. El-Rewini, *The time complexity of scheduledduling interval orders with communication is polynomial,* Parallel Processing Letters **3** (1) (1993) 53-58.

[2] C. G. Bell, *Gordon Bell on the Future of Computers,* interview in SIAM NEWS, February 1987.

[3] P. Chrétienne, C. Picouleau, "Scheduling with Communication Delays: A Survey", In *Scheduling Theory and Its Applications*, P. Chrétienne et al. (Eds.), J. Wiley, 1995.

[4] H. Jung, L. Kirousis, P. Spirakis, *Lower bounds and efficient algorithms for multiprocessor scheduling of DAGs with communication delays,* Information and Computation **105** (1993) 94-104.

[5] A. Jakoby and R. Reischuk, *The complexity of scheduling problems with communication delays for trees,* In Proc. Scandinavian Workshop on Algorithm Theory, 1992, (SWAT'92), Springer Verlag LNCS-621 (1992) 165-177.

[6] L. Finta, Z. Liu, I. Milis, E. Bampis *Scheduling UET-UCT series parallel graphs on two processors,* Theoretical Computer Science 162 (1996) 323-340.

[7] L. Gao, A. L. Rosenberg and R. K. Sitaraman, *Optimal architecture-independent scheduling of fine-grain tree-sweep computations,* In Proc. 7^{th} IEEE Symposium on Parallel and distributed Processing (1995) 620-629.

[8] M.R. Garey, D. S. Johnson, "Computers and Intractability, A Guide to the Theory of NP-completeness", Ed. Freeman, 1979.

[9] R. L. Graham, E. L. Lawler, J. K. Lenstra, K. Rinnooy Kan, "Optimization and Approximation in Deterministic Scheduling: A Survey", *Ann. Disc. Math.*, **5** (1979) 287-326.

[10] F. Guinand, D. Trystram, "Optimal Scheduling of UECT Trees on Two Processors". Technical Report APACHE RR-93-03, IMAG, Grenoble, 1993.

[11] C. Papadimitriou, J. Ullman, *A Communication Time Tradeoff,* SIAM J. on Computing, 4 (1987) 639-646.

[12] C. Papadimitriou, M. Yannakakis, *Towards an Architecture-Independent Analysis of Parallel Algorithms,* SIAM J. on Computing, 2 (1990) 322-328.

[13] C. Picouleau, *Etude des problèmes d'optimisation dans les systèmes distribués*, Ph.D. Thesis, Université Pierre et Marie Curie, France, 1992.

[14] V. J. Rayward-Smith, "UET Scheduling with Unit Interprocessor Communication Delays", *Disc. Appl. Math.*, **18** (1987), 55-71.

[15] T. Varvarigou, V. P. Roychowdhury, T. Kailath, E. Lawler. "Scheduling In and Out Forests in the Presence of Communication Delays". In *Proc. Intern. Parallel Processing Symposium*, Newport Beach, CA,(1993) 222-229, IEEE Trans. on Parallel and Distributed Systems, **7** (10) (1996) 1065-1074.

[16] M. Veldhorst, "A linear Time Algorithm to Schedule Trees with Communication Delays Optimally on Two Machines". Technical Report COSOR 93-07, Dep. of Math. and Comp. Sci., Eindhoven Univ. of Technology, 1993.

[17] B. Veltman, "Multiprocessor Scheduling with Communication Delays", Ph.D. Thesis, CWI-Amsterdam, (1993).

[18] B. Veltman, B. J. Lageweg, J. K. Lenstra, "Multiprocessor Scheduling with Communication Delays", *Parallel Computing*, **16** (1990) 173-182.