Stony Brook University

Analysis of Resource Scheduling Strategies in Parallel, Distributed and Grid Computing Systems

A Dissertation Presented

by

Kijeung Choi

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

December 2009

Stony Brook University

The Graduate School

Kijeung Choi

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby recommend
acceptance of this dissertation.

Thomas G. Robertazzi – Dissertation Advisor
Professor, Department of Electrical and Computer Engineering

Wendy Tang – Chairperson of Defense
Associate Professor, Department of Electrical and Computer Engineering

John Murray
Associate Professor, Department of Electrical and Computer Engineering

Hussein Badr
Associate Professor, Department of Computer Science

This dissertation is accepted by the Graduate School.

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Analysis of Resource Scheduling Strategies in Parallel, Distributed and Grid Computing Systems

by

Kijeung Choi

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2009

Large-scale parallel and grid computing systems are becoming more mainstream in order to achieve high performance. The handling of large volumes of computational load on distributed and parallel computing systems is becoming a more and more challenging task. In this dissertation, we consider the problem of the analysis of resource scheduling strategies in parallel, distributed and grid computing systems. We employ a most promising algebraic means of determining the optimal allocations of load fractions to processors and links in a given interconnection topology, Divisible Load Theory (DLT). We investigate scheduling on a real-time networks with arbitrary processor release times. The theoretical analysis shows that generated solutions based on a conjectured timing model does not always admit a feasible solution. A heuristic algorithm for generating sufficient models for a feasible solution is developed. For the next topic, scheduling scenarios in wireless sensor networks, which are dynamically growing and promising distributed computing systems, are examined based on DLT. The performance of these scenarios is examined with respect to different sensing speeds, communication speeds, and information utility constant parameters. The special bounds for the ratio of speed parameters for the maintenance of the minimum round time for certain scenarios are derived. Mathematical analysis based on DLT contributes to interpret and evaluate performance of the wireless sensor networks. For the next study, we consider a simulation of the adaptive capacity utilization problem for data transfers between multiple source and destination nodes interconnected by modern, high-performance, hybrid networks

that support resource reservations. We account for a file transfer scenario using node capacity, file size, file transfer start time, and the deadline of files when determining the explicit capacity of Virtual Paths (VPs) across a backbone. Two opposite heuristic algorithms are designed to react to file transfers according to the temporal capacity of VPs for multiple sources and destinations networks. For the next topic, a special type of parallel computing system, "Grid" computing systems, has appeared as a promising trend for large-scale distributed parallel processing systems. An optimal computing power allocation solution is adapted to divisible load in a parallel computing grid is with an idealized two sources and a single sink computing processor. For the last topic, one of the major challenges to all processor requirements for high performance network systems now and in the future will be low cost. By applying DLT, we consider the problem of monetary network cost on a homogeneous tree networks. Our objective is to analyze quantitative and qualitative trends of monetary network cost against ratio of network speed parameters and to determine relationships between the network cost and the network environment. As going to Terascale / Petascale high performance computing (HPC) systems and beyond means that the number of components (cores, interconnect, storage) within such a system will grow enormously, it is obvious that these highly parallel systems will raise questions about reliable information about resource management and scheduling. Thus, it can be expected that adaptation of DLT is promising technique for a new era of HPC especially for resource allocation, computing power adjusted according to computing load, and network performance prediction in a parallel paradigm.

Contents

List of Figures

List of Tables

Acknowledgements

Foremost, I would like to express my deep and sincere gratitude to my advisor, Professor Thomas G. Robertazzi, Ph.D, Electrical and Computer Engineering Department, Stony Brook University. His wide knowledge and his logical way of thinking have been of great value for me. His understanding, encouraging and personal guidance have had a remarkable influence on my entire career in the field of parallel and distributed systems.

I owe my most sincere gratitude to Distinguished Professor, Gregory Belenky, Ph.D, who gave me important guidance during my first step into Master studies and gave me untiring help during my difficult moments.

I am also deeply grateful to Professor Petar M. Djurić, Ph.D, who gave me the opportunity to study particle filter. His kind support and guidance have been of great value in my research.

I wish to sincere thank Dr. Dimitrios Katramatos, Ph.D and Dr. Dantong Yu, Ph.D, Terapaths research team in Brookhaven National Laboratory, for their guidance and interesting discussions on network resource scheduling.

I also wish to specially thank Dr. Marcus M. Edvall, Tomlab Optimization Inc, for his kind assistance and guidance in utilizing nonlinear programming.

Last but not the least, my love goes to my families for the emotional support and encouragement underlying this dissertation. Thanks to my father, Taeyoung Choi, Professor, Ajou University., my mother, Jinja Kim, Pharmacist, Incheon International Airport, and my lovely younger sister, Youlan Choi for being a source of pride to me.

The financial support of Stony Brook University is gratefully acknowledged.

Chapter 1

Introduction

Future increases in computing network performance have been shown to be achieved through increases in system scale using a larger number of components, utilizing large computing resources for a single problem. Thus, the handling (scheduling) of large volumes of computational load on such large parallel and distributed computing networks is becoming a more and more challenging task. To fulfill an emerging need for the efficient scheduling of computing load, a new mathematical tool, called *divisible load theory* (DLT) [1],[2],[3], has been created to allow tractable and realistic performance analysis of systems incorporating communication and computation issues, as in parallel and distributed processing. In DLT, it is assumed that computation and communication loads can be arbitrarily partitioned into infinitesimally small fractions (arbitrarily divisible) and distributed among processors and links in the network. DLT basically adopts a load property that there are no precedence relations among the load fractions. A key feature of this divisible load scheduling theory is that it uses a linear mathematical model of the computation (processor) speed and communication (link) speed parameters in devising efficient load scheduling strategies. Basically, in DLT, the communication time is assumed to be linearly proportional to the amount of load that is transferred over the link, and the computation time is linearly proportional to the amount of load assigned to the processor to be computed. Considerable attention was focused on minimizing the total processing time of the entire load in a large amount

of studies. Based on the linear system modeling, a recursive deterministic mathematical formulation is often used to find the optimal fractions of the entire load such that the total processing finish time is a minimum. The optimal solution for a set of linear recursive equations characterizing time delay in communication and computation is imposed by the *optimality principle* [1], and is the key to DLT. Load scheduling strategies can be categorized according to the equipment of front-ends to processors and to possibility of simultaneous load distribution from root (originator) to processors. In the without front-end case, a processor can compute or communicate, but not do both at the same time (staggered start). On the other hand, in the with front-end case, a processor can begin computation as soon as it begins to receive load (simultaneous start). Simultaneous load distribution is possible as long as the speed of the CPU is fast enough to continually load buffers for each of its output ports.

1.1 Related Studies

DLT is a powerful concept for the performance analysis of networks of processors and links. The creation of DLT is motivated by modeling of a linear sensor network of communicating processors [4]. Because the theory's linearity and continuous mathematics framework, the tractability of DLT became apparent through early work. It has been examined for network topologies including linear daisy chain, bus network, and tree networks using a set of recursive equations [4],[5],[6],[7],[8],[9]. Load distribution strategies for multi-dimensional mesh networks was presented [10],[11],[12],[13],[14],[15]. In [16] the concept of time varying computational and communication environment was introduced. Optimal conditions were found which determine which processors have to be utilized in order to minimize the finish time [17]. The computational complexity in DLT is considered in [18],[19]. Systems with memory size limitations are analyzed based on DLT in [20],[21],[22]. DLT has been applied in various research areas such as computational biological [23], theoretical matrix-vector computation [24], and image processing techniques [25],[26].

1.2 Major Contribution

My research begins with scheduling on a real-time homogeneous single-level tree network (bus network) with arbitrary release times. A scheduling policy with sequential distribution and staggered start is considered. The theoretical analysis shows that generated solutions based on a conjectured timing model does not always admit a feasible solution. I suggest a heuristic algorithm for generating sufficient models for a feasible solution to be found. The proposed heuristic algorithm is quite promising as a starting point for generating appropriate timing models for networks composed of processors with arbitrary release times. It also seems of interest to extend the conjecture based approach to other network models based on a simultaneous distribution and the simultaneous start scheduling strategy. This optimization problem has a rich structure and open challenges includes developing scalable, rather than exhaustive, algorithms for load allocation optimization and a better understanding of the problems feasibility conditions. I believe that this work is significant for showing a method to take into account load scheduling with processors with arbitrary release times.
The second research area involves several scheduling scenarios of homogeneous single cluster wireless sensor networks (WSNs). The performance of these scenarios is examined with respect to the different sensing speed, communication speed, and information utility constant parameters. I derive the condition for the feasible measurement instruction assignment time constant. The special bounds for the ratio of speed parameters for the maintenance of the minimum round time for certain scenarios are also derived. As an extension to the work, an analysis of a multi-cluster based wireless sensor network topology with a speedup analysis for possible scenarios and a corresponding comprehensive interpretation with mathematical derivation are performed.
Another interesting research effort was performed with several research engineers at Brookhaven National Laboratory (BNL), handling large volumes of computational load on distributed and parallel high performance networks. In this work, we consider the adaptive capacity utilization problem for data transfers between multiple source and destination nodes

interconnected by modern, high-performance, hybrid networks that support resource reservations. We account for a file transfer scenario using node capacity, file size, file transfer start time, and the deadline of files when determining the explicit capacity of Virtual Paths (VPs) across a backbone. Our work is the first attempt to systematically investigate the nature of capacity reservation on VPs. We propose two heuristic algorithms referred to as the Most Conservative (MC) algorithm and the Load Balancing (LB) algorithm using a newly introduced parameter, capacity utilization. These two opposite heuristic algorithms are designed to react to file transfers according to the temporal capacity of VPs for multiple sources and destinations networks. The proposed heuristic algorithms are quite promising as a starting point for studying the scheduling of file transfers through modern hybrid high-performance networks.

Future increases in computing network performance have been shown to be achieved through increases in system scale using a larger number of components, utilizing large computing resources for a single problem. A promising trend in this regard is Grid scheduling. A special type of parallel computing system, "Grid" computing systems, have appeared as a promising trend for large-scale distributed parallel processing systems. Grid computing is applying the resources of many computers in a network to a single task at the same time. For collaborative grid computing across many computers, each computer is multiprogrammed. Multiprogramming is the technique of running several programs at a time using timesharing. It allows a computer to do several tasks at the same time. Thus, multiprogramming creates logical parallelism. In this dissertation, an optimal computing power allocation solution is adapted to divisible loads in a parallel computing grid with an idealized two source and a single sink processor model. We note that the analysis of the optimal computing power allocation is provided for the first time. A more comprehensive approach with more precise Grid models is worth addressing in future works.

One of the major challenges to all processor requirements for high performance network systems now and in the future will be low cost. By applying DLT, we consider the problem of monetary network cost on a homogeneous tree networks. Our objective is to analyze

trends of monetary network cost against ratio of network speed parameters and to determine relationships between the network cost and the network environment. To be specific, two strategies regarding load distribution (sequential load distribution and simultaneous load distribution) are discussed. The cost trends in various networks conditions are studied for the both cases of load distribution. It is apparent that quantitative monetary cost is important to scale cost demand and perform parallel processing under a limited monetary budget. Further, the quality of monetary usage is worth taking into account for the both cases of load distribution. In other words, how efficiently the monetary cost is spent for the parallel processing indicates the value of the unit cost. It is found that cost efficiency is a good indicator of the quality of cost consumption in high performance parallel processing. New enhanced metrics for cost analysis are worth being developed for evaluating new trends in cost performance.

Chapter 2

An Exhaustive Approach to Release Time Aware Divisible Load Scheduling

Most of the previous studies adopting different load distribution strategies so far had assumed a network in which there are no time constraints. In other words, it was assumed that all the processors in the system are available from the time instant at which the root processor starts the divisible load distribution. In practical scenarios, this is not always the case. We refer to the time instance at which a processor is available for processing the divisible load as *release time*. Some recent studies [27],[28] present real-time scheduling strategies involving the release time of processors and processing deadlines based on the classic DLT analysis. Load scheduling adopting multi-installment techniques [29] in bus networks with release times is analyzed in [30]. Release time aware load scheduling with buffer size constraints is considered [31]. A heuristic strategy is proposed for scheduling in linear daisy chain networks with release times [32].

The analysis of these preceding studies are implicitly restricted to the scenario that all distributions of load fractions from a root processor are completed before the earliest release time is reached. In this chapter, we release this restriction in that the distribution of the

load fractions starts once the first child processor is released. We assume that processors in the network are not equipped with front end processors. Hence, computation occurs only after communication ends (i.e., staggered start). The load distribution occurs in a sequential manner. With the sequential distribution and staggered start scenario, many possible models with associated timing diagrams restricted with two constraints need to be considered to obtain an optimal solution in terms of minimum processing finish time.

The intriguing fact here is that the developed closed form solutions from the possible timing models with two constraints may give us multiple feasible solutions, but not an optimal solution. In other words, the solutions may not follow the timing models of analysis. Put another way, optimal load allocations depend on release times but the timing of release times depends on the optimal load allocation solution. Then, we may be caught in a vicious circle. To resolve the circular reasoning problem, we propose an exhaustive search algorithm to search a timing model for an optimal solution. Several important aspects regarding the scheduling behavior of the exhaustive search algorithm are demonstrated through a simulation study.

The remainder of this chapter is organized as follows. Section 2.1 presents the types of notations and analytic background. Section 2.2 analyzes the diagrams based on the two timing constraints and demonstrates the exhaustive search algorithm. Simulation results showing the behavior of the exhaustive search algorithm are discussed in Section 2.3.

2.1 Problem Formulation and Preliminary Remarks

Consider the case where a bus network model consists of a root processor and n children processors. The children processors $p_1, p_2, \ldots, p_n$ are connected to the root processor through a bus. The root processor divides the total load into n parts, namely, $\alpha_1, \alpha_2, \ldots, \alpha_n$. Then, the root processor distributes the load fractions to the children processors, respectively (see Fig. 2.1).

In this chapter, the following assumptions are initially made:

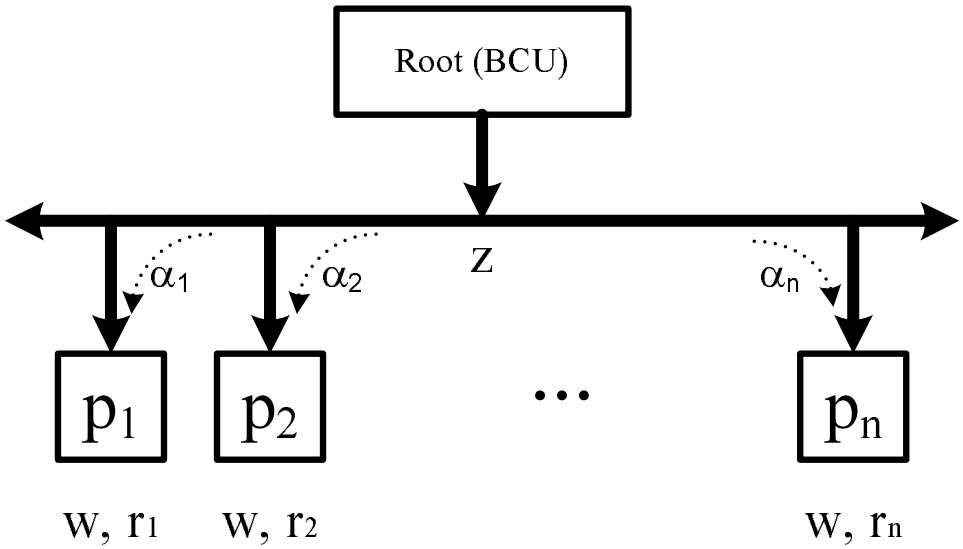


Figure 2.1: Bus network architecture.

1. The root processor (BCU) does not participate in load computation.

2. The root processor (BCU) distributes load fractions to all the children processors in sequential manner.

3. The children processors are not equipped with front-end processors. If a processor does not has a front-end processor, it can compute or communicate, but not do both at the same time (i.e., staggered start scenario). The load distribution from the root processor to a children processor can not take place before the children processor is released.

4. Each of the children processors have a capability to start processing as soon as the processor becomes available at release time r_i, where $i = 1, 2, \ldots, n$.

5. Each of the children processors stops computing at the same time. Intuitively, this is because otherwise some processors would be idle while others were still busy. A rigorous proof for this argument in the case of linear, bus, and tree network parameters, is given in [1].

6. The root processor (BCU) knows the release times of the processors.

7. Without loss of generality, the order of the children processors follows the order in which their release times increase, that is $r_1 \leq r_2 \leq \ldots \leq r_n$. In other words, the sequence of load distribution by the root processor (BCU) follows this order and let this order be

$p_1, p_2, \ldots, p_n$.

8. Compared to the size of the data, the time to report solutions back to the root processor (BCU) is negligible.

The following notation is used in this chapter.

α_i : The load fraction assigned to the ith children processor (where $i = 1, 2, \ldots, n$).

w : The inverse computing speed of the processors.

z : The inverse communication speed of the links.

T_{cp} : Computing intensity constant. The entire load can be processed on the i^{th} processor in time wT_{cp}.

T_{cm} : Communication intensity constant. The entire load can be transmitted over the i^{th} link in time zT_{cm}.

$T_{f,n}$: The finish time. Time at which n children processors complete computation.

T_i : The total time that elapses between the beginning of the process at $t = 0$ and the time when i^{th} processor completes its computation (where $i = 1, 2, \ldots, n$).

r_i : The release time. Time at which the i^{th} processor becomes ready for receiving assigned load from the root processor (BCU) (where $i = 1, 2, \ldots, n$).

s_i : The start time. Time at which the i^{th} processor actually starts receiving the assigned load from the root processor (BCU) (where $i = 1, 2, \ldots, n$). Here, $s_i \geq r_i$.

d : The absolute deadline. Time by which the load must complete.

D_i : The relative deadline. Time difference between the absolute deadline and the release time: $D_i = d - r_i$ (where $i = 1, 2, \ldots, n$).

Γ_i : The pair of i^{th} processor and $(i+1)^{th}$ processor, (p_i, p_{i+1}) (where $i = 1, 2, \ldots, n-1$).

2.2 A bus network with arbitrary release times

Consider a cluster with a bus network architecture where loads are distributed sequentially to the children processors as soon as the processor become available (i.e., *sequential distribution*). As mentioned earlier, processors are not equipped with front-end processors, so that processors start computing only after they receive the whole of the load assigned to them (i.e., *staggered start*), and initial communication starts at r_1 (i.e., s_1=r_1), the time when the whole load is assumed to be present at the root processor. Note that it is assumed that the release times of the processors, r_i (where $i = 1, 2, \ldots, n$) are given and fixed arbitrary constants. Accordingly, timing models with two rigid constraints are considered:

- $r_{i+1} - s_i \leq \alpha_i z Tcm$: The case that the release time of the $(i+1)^{th}$ processor, r_{i+1}, occurs before the $(i+1)^{th}$ load assignment from the root processor (BCU) starts or occurs during the middle of the communication time for the load assignment. Given the sequential load distribution and staggered start scenario, this timing constraint has no impact on the load scheduling. Here, $i = 1, 2, \ldots, n-1$.

- $r_{i+1} - s_i > \alpha_i z Tcm$: The case that the release time of the $(i+1)^{th}$ processor, r_{i+1}, occurs after the $(i+1)^{th}$ load assignment from the root processor (BCU) ends so that the $(i+1)^{th}$ processor needs to wait until its own release time, r_{i+1}, occurs to start communication with the root processor (BCU) to be assigned its own load. Hence, we can rewrite the constraint as $r_{i+1} - r_i > \alpha_i z Tcm$. Here, $i = 1, 2, \ldots, n-1$.

These constraints can exist for each pair of processors. The corresponding timing diagrams are as shown in Fig. 2.2 and Fig. 2.3.

The important condition for an optimal solution is that the minimum finish time $T_{f,n}$ occurs when all processors stop at the same time as we mentioned before. In other words, the identical finish time (i.e., $T_1 = T_2 = \ldots = T_n$) is the necessary condition to obtain the

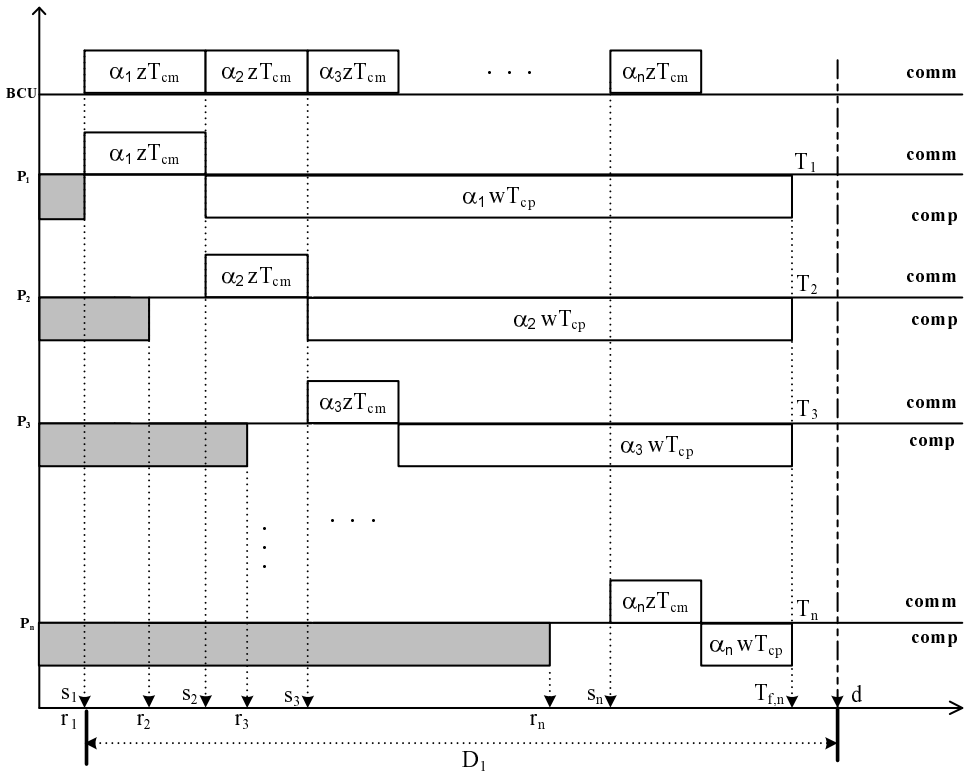


Figure 2.2: Sequential Distribution and Staggered Start, $r_{i+1} - s_i \leq \alpha_i zTcm$

optimal solutions for the minimum finish time.

Before we take into account the two inequality constraints, it is worth mentioning that the two constraints are assumed in that we do not have information about the values of α_i beforehand. The optimal solution for α_i can be obtained based on timing diagrams satisfying a certain scheduling strategy. Once the optimal solutions are achieved, it seems to be reasonable to mention the two constraints. Indeed, initially we assume that the timing model of given processors follows a certain constraint among the two constraints for each consecutive pair of processors over the following two subsections. From the timing diagrams (Fig. 2.2 and 2.3), the corresponding recursive load distribution equations so that all of the processors stop computing at the same time (i.e., $T_1 = T_2 = \ldots = T_n$) can be obtained as follows

$$s_i + \alpha_i zT_{cm} + \alpha_i wT_{cp} = s_{i+1} + \alpha_{i+1} zT_{cm} + \alpha_{i+1} wT_{cp} \qquad i = 1, 2, \ldots, n-1 \tag{2.1}$$

2.2.1 Constraint I : $r_{i+1} - s_i \leq \alpha_i z T cm$

In this subsection, we derive closed form solutions for an evaluation of the performance of a particular case. The sufficient condition for the feasible release times for the optimal solution is also derived. Initially we assume that the optimal scheduling scenario of n processors $(p_1, \ldots, p_n)$ satisfies *Constraint I* (see Fig. 2.2). Without this assumption, the derivation of the optimal solutions and corresponding performance evaluations is not possible. From Fig. 2.2, we can express the start times in terms of α_i as

$$s_i = r_1 + \left(\sum_{l=1}^{i-1} \alpha_l \right) z T_{cm} \qquad i = 2, 3, \ldots, n \tag{2.2}$$

By substituting (3.2) into *Constraint I*, we observe that the feasible release times must satisfy the following condition:

$$r_{i+1} \leq r_1 + \left(\sum_{l=1}^{i} \alpha_l \right) z T_{cm} \qquad i = 1, 2, \ldots, n-1 \tag{2.3}$$

Intuitively, the inequality (2.3) is recognizable from Fig. 2.2.

Using equation (2.1) and (2.2), we can obtain the value of α_i as

$$\alpha_i = q^{i-1} \alpha_1 \qquad i = 2, 3, \ldots, n \tag{2.4}$$

Here, $q = \frac{w T_{cp}}{z T_{cm} + w T_{cp}}$. The fractions of the total load should sum to one (normalization)

$$\sum_{i=1}^{n} \alpha_i = 1 \tag{2.5}$$

Then the normalization equation (2.5) becomes

$$\alpha_1 \sum_{i=1}^{n} q^{i-1} = 1, \quad \alpha_1 = \frac{1-q}{1-q^n} \tag{2.6}$$

From equation (2.6), the minimum finish time is given by

$$\begin{aligned} T_{f,n} = T_1 &= r_1 + \alpha_1 z T_{cm} + \alpha_1 w T_{cp} \\ &= r_1 + \frac{1-q}{1-q^n}\left(w T_{cp} + z T_{cm}\right) \end{aligned} \tag{2.7}$$

The preceding derivation of the optimal solution has been already appeared in [7] introducing the load scheduling scenario with sequential distribution and staggered start. The derivational similarity occurs because the timing diagram of sequential distribution and staggered start does not vary according to the release time under *Constraint I* as we mentioned previously.

The minimum number of processors, n^{min} that the whole load needs at start time $s_1(= r_1)$ to meet the absolute deadline, d can be derived as follow

$$T_{f,n} \leq r_1 + D_1 \tag{2.8}$$

This implies that

$$r_1 + \frac{1-q}{1-q^n}\left(w T_{cp} + z T_{cm}\right) \leq r_1 + D_1 \tag{2.9}$$

since $1 - q^n > 0$

$$\begin{aligned} 1 - q^n &\geq \frac{(1-q)\left(w T_{cp} + z T_{cm}\right)}{D_1} \\ q^n &\leq 1 - \frac{(1-q)\left(w T_{cp} + z T_{cm}\right)}{D_1} \\ q^n &\leq 1 - \frac{z T_{cm}}{D_1} \end{aligned} \tag{2.10}$$

Note that $0 < q < 1$. Thus, we have

$$n \geq \frac{\ln \eta}{\ln q}, \quad where \quad \eta = 1 - \frac{z T_{cm}}{D_1} \tag{2.11}$$

Here, $0 < \eta < 1$ otherwise the distribution of the partitioned load can not meet the deadline and even can not have enough time for computing (see Fig. 2.2). Therefore, the minimum number of children processors that the load needs at time $s_1 (= r_1)$ to meet its deadline is

$$n^{min} = \lceil \frac{\ln \eta}{\ln q} \rceil \tag{2.12}$$

2.2.2 Constraint II : $r_{i+1} - s_i > \alpha_i z T cm$

In this subsection, we derive closed form solutions for an evaluation of the performance of a different case by following a similar procedure described in the earlier case. Note that the release time r_i is the same as the start time s_i for $i = 1, 2, \ldots, n$ under *Constraint II* as shown in Fig. 2.3. Under this constraint, it is quite possible that some processors have too large a release time so that the processors can not be assigned any fraction of the load from the root processor (BCU). The criteria for the feasible release times is also derived in this section. We assume that n processors participating in distributed computing satisfy *Constraint II* initially. As in the previous subsection, load distribution equations in this subsection will not be reasonable without the assumption. Rewriting (2.1) based on the timing diagram (see Fig. 2.3) as

$$\alpha_i = \alpha_1 + \frac{r_1 - r_i}{wT_{cp} + zT_{cm}} \qquad i = 2, 3, \ldots, n \tag{2.13}$$

Also, the fractions of the total load should sum to one (normalization). This gives n linear equations with n unknowns. Then the normalization (2.5) becomes

$$\begin{aligned} n\alpha_1 + \frac{(n-1)\, r_1 - \sum_{i=2}^{n} r_i}{wT_{cp} + zT_{cm}} &= 1 \\ \alpha_1 = \frac{1}{n}\left(1 - \frac{(n-1)\, r_1 - \sum_{i=2}^{n} r_i}{wT_{cp} + zT_{cm}}\right) &= \frac{1}{n}\left(1 + \frac{-nr_1 + \sum_{i=1}^{n} r_i}{wT_{cp} + zT_{cm}}\right) \end{aligned} \tag{2.14}$$

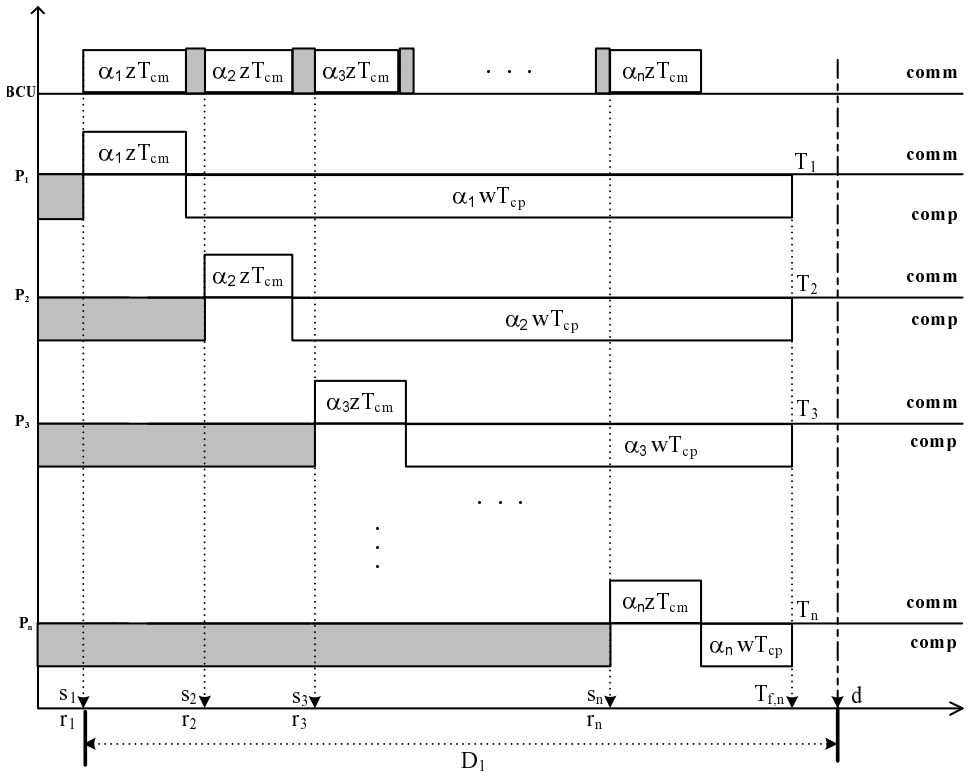


Figure 2.3: Sequential Distribution and Staggered Start, $r_{i+1} - s_i > \alpha_i z T cm$

Substituting (2.15) into (2.14) yields

$$\begin{aligned} \alpha_i &= \frac{1}{n}\left(1 + \frac{-nr_1 + \sum_{i=1}^{n} r_i}{wT_{cp} + zT_{cm}}\right) + \frac{r_1 - r_i}{wT_{cp} + zT_{cm}} \\ &= \frac{1}{n}\left(1 + \frac{-nr_i + \sum_{i=1}^{n} r_i}{wT_{cp} + zT_{cm}}\right) \qquad i = 2, 3, \ldots, n \end{aligned} \tag{2.15}$$

From (2.15), we can see that the α_i just depends on the prior values of release time and the number of the processors as comparing to (2.4) given in the case of *Constraint I*. This speciality for the case of *Constraint II* opens the possibility that a condition for the normalized solution, $0 < \alpha_i \leq 1$, is violated. To generate the condition for the feasible processor's release times, we can use the necessary condition $0 < \alpha_i \leq 1$ for $i = 1, 2, \ldots, n$. This leads to the following important criterion for the feasible release times

$$0 < \frac{1}{n}\left(1 + \frac{-nr_i + \sum_{i=1}^{n} r_i}{wT_{cp} + zT_{cm}}\right) \leq 1 \tag{2.16}$$

We can rewrite above inequality as follows

$$C_1 \leq r_i < C_2 \tag{2.17}$$

where

$$\begin{aligned} C_1 &= \frac{wT_{cp} + zT_{cm} + \sum_{i=1}^{n} r_i}{n} - (wT_{cp} + zT_{cm}) \\ C_2 &= \frac{wT_{cp} + zT_{cm} + \sum_{i=1}^{n} r_i}{n} \end{aligned} \tag{2.18}$$

In order to obtain the feasible release times, it is required to apply the condition (2.17) iteratively by reducing the number of processors, n until all release times satisfy the necessary condition (2.17). Note that for the optimal solutions, the release times should admit *Constraint II*, which is the sufficient condition for the optimal solutions.

From (2.14), the minimum finish time is given by

$$\begin{aligned} T_{f,n} = T_1 &= r_1 + \alpha_1 zT_{cm} + \alpha_1 wT_{cp} \\ &= r_1 + \frac{1}{n}\left(1 + \frac{-nr_1 + \sum_{i=1}^{n} r_i}{wT_{cp} + zT_{cm}}\right)(wT_{cp} + zT_{cm}) \\ &= \frac{1}{n}\left(wT_{cp} + zT_{cm} + \sum_{i=1}^{n} r_i\right) = \frac{1}{n}(wT_{cp} + zT_{cm}) + \mu \end{aligned} \tag{2.19}$$

Here, let $\mu = \frac{1}{n}\sum_{i=1}^{n} r_i$, which is an average value of n release time points.

Using the similar method shown in the previous section, the minimum number of processors, n^{min} that the load needs at start time $s_1 (= r_1)$ to meet the absolute deadline, d can be derived as follows

$$T_{f,n} \leq r_1 + D_1 \tag{2.20}$$

This implies that

$$\frac{1}{n}(wT_{cp} + zT_{cm}) + \mu \leq r_1 + D_1 \tag{2.21}$$

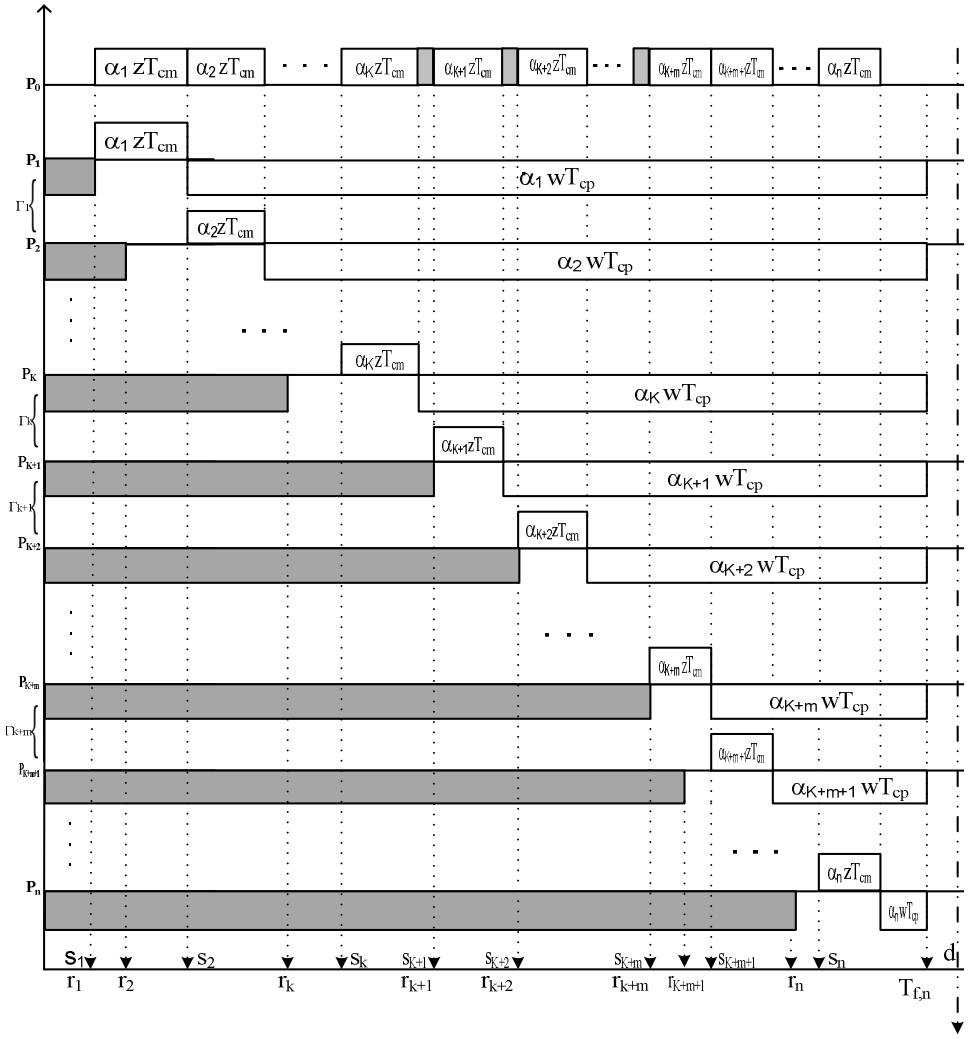


Figure 2.4: An example of a timing model, one of 2^{n-1} cases

Thus, we have

$$n \geq \frac{wT_{cp} + zT_{cm}}{r_1 + D_1 - \mu} \tag{2.22}$$

Here, it is obvious that $r_1 + D_1$, which is the absolute deadline, d is larger than μ. This is because all release times r_i of the eligible processors are surely less than d. Therefore, the minimum number of children processors that the load needs at time $s_1(= r_1)$ to meet its deadline is

$$n^{min} = \lceil \frac{wT_{cp} + zT_{cm}}{r_1 + D_1 - \mu} \rceil \tag{2.23}$$

2.2.3 Exhaustive search algorithm

One may recognize that a schedule will not necessarily involve only one of the two constraints (i.e., *Constraint I* and *Constraint II*). An actual scheduling may consist of a mixture of both constraints. In this section, we focus on scheduling divisible loads with arbitrary release time with no prior knowledge of constraints. In case that n processors get involved in the distributed computing, $2^{(n-1)}$ possible models should be taken into account. Fig. 2.4 shows an example of a timing diagram among $2^{(n-1)}$ possible cases which the exhaustive search algorithm takes into account. Note that the diagram shows one of the possible cases that the load distribution follows the decreasing order of the release time. From the Fig. 2.4, one can see that the example diagram is composed of $\Gamma_1(I) \ldots \Gamma_{k-1}(I)$, $\Gamma_k(II) \ldots \Gamma_{k+m-1}(II)$, and $\Gamma_{k+m}(I) \ldots \Gamma_{n-1}(I)$. Here, we denote the i^{th} pair of processors, Γ_i, satisfying *Constraint I* and *Constraint II* as $\Gamma_i(I)$ and $\Gamma_i(II)$ respectively. From the timing diagram (Fig. 2.4), one can write the following set of deterministic equations in the system

$$
\begin{aligned}
T_1 &= r_1 + \alpha_1 (zT_{cm} + wT_{cp}) \\
T_2 &= s_2 + \alpha_2 (zT_{cm} + wT_{cp}) \\
&\vdots \\
T_k &= s_k + \alpha_k (zT_{cm} + wT_{cp}) \\
T_{k+1} &= r_{k+1} + \alpha_{k+1} (zT_{cm} + wT_{cp}) \\
T_{k+2} &= r_{k+2} + \alpha_{k+2} (zT_{cm} + wT_{cp}) \\
&\vdots \\
T_{k+m} &= r_{k+m} + \alpha_{k+m} (zT_{cm} + wT_{cp}) \\
T_{k+m+1} &= s_{k+m+1} + \alpha_{k+m+1} (zT_{cm} + wT_{cp}) \\
&\vdots \\
T_n &= s_n + \alpha_n (zT_{cm} + wT_{cp})
\end{aligned}
\tag{2.24}
$$

In order to achieve the optimal solutions for the minimum finish time, all processors stop processing at the same time as we mentioned earlier (i.e., $T_1 = T_2 = T_3 = \ldots = T_n$). Based on the above set of equations, one can write the following set of $n-1$ equations

$$\begin{aligned}
\alpha_2 &= q\alpha_1 \\
\alpha_3 &= q\alpha_2 \\
&\vdots \\
\alpha_k &= q\alpha_{k-1} \\
\alpha_{k+1} &= \alpha_1 + \tfrac{r_1 - r_{k+1}}{wT_{cp}+zT_{cm}} \\
\alpha_{k+2} &= \alpha_1 + \tfrac{r_1 - r_{k+2}}{wT_{cp}+zT_{cm}} \\
&\vdots \\
\alpha_{k+m} &= \alpha_1 + \tfrac{r_1 - r_{k+m}}{wT_{cp}+zT_{cm}} \\
\alpha_{k+m+1} &= q\alpha_{k+m} \\
&\vdots \\
\alpha_{n-1} &= q\alpha_{n-2} \\
\alpha_n &= q\alpha_{n-1}
\end{aligned} \tag{2.25}$$

The pattern of the series of equations under assumption of either *Constraint I* or *Constraint II* can be also seen as shown in section *2.2.1* and *2.2.2*. As we mentioned earlier, the fractions of the total load should sum to one

$$\sum_{i=1}^{n} \alpha_i = 1 \tag{2.26}$$

Based on the above total n equations (i.e., (2.25) and (2.26)), we can obtain the solutions for the fraction of load to be assigned to n processors. By using this analytical method, we can obtain solutions for all $2^{(n-1)}$ possible cases. Then, we can find an optimal case based on the obtained values of $\alpha's$. The sets of deterministic equations obtained from 2^{n-1} possible cases are solved via linear programming (LP) technique. In the manner of

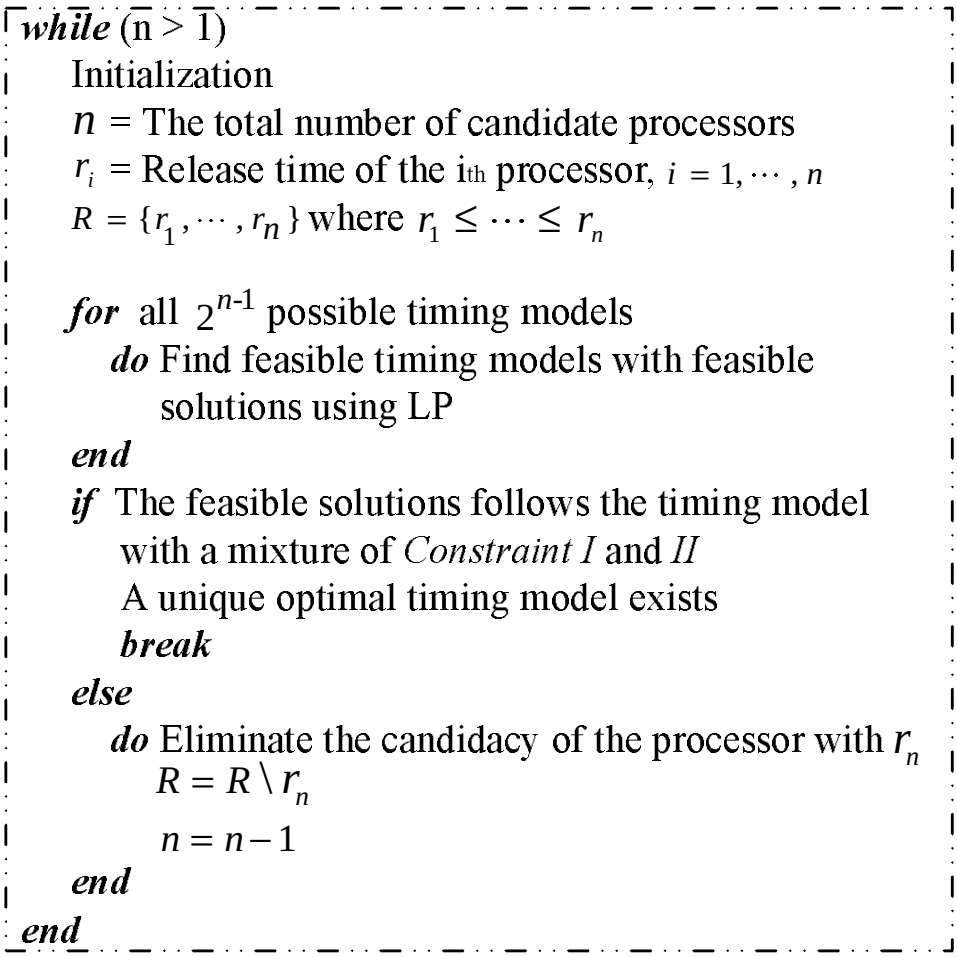


Figure 2.5: Exhaustive Search Algorithm

the required computing power, linear programming is a quite useful method for drastically improved computing speed to find the optimal solution. For the linear programming, (2.25) can be expressed in canonical form as

$$\mathbf{A} \cdot \alpha = \mathbf{b} \tag{2.27}$$

α represents the $n \times 1$ vector of solution variables to be determined. From (2.25), the $n \times n$ known coefficient matrix, $\mathbf{A}$ and the $n \times 1$ vector of known coefficient, $\mathbf{A}$ can be written as

$$\mathbf{A} = \begin{pmatrix}
-q & 1 & 0 & \cdots & & & & & & & & 0 \\
0 & -q & 1 & 0 & \cdots & & & & & & & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \cdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & -q & 1 & 0 & \cdots & & & & & & 0 \\
-1 & 0 & \cdots & & 0 & 1 & 0 & \cdots & & & & & 0 \\
-1 & 0 & & \cdots & & 0 & 1 & 0 & \cdots & & & & 0 \\
\vdots & \vdots & & & \cdots & & \ddots & \ddots & \ddots & \cdots & & & \vdots \\
-1 & 0 & & & & \cdots & & 0 & 1 & 0 & \cdots & & 0 \\
0 & 0 & & & & & \cdots & 0 & -q & 1 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \cdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
0 & & & & & & & & \cdots & 0 & -q & 1 & 0 \\
0 & 0 & & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & -q & 1 \\
1 & 1 & & & \cdots & 1 & 1 & & & & \cdots & 1 & 1
\end{pmatrix} \quad \mathbf{b} = \begin{pmatrix}
0 \\ 0 \\ \vdots \\ 0 \\ \frac{r_1 - r_{k+1}}{wT_{cp} + zT_{cm}} \\ \frac{r_1 - r_{k+2}}{wT_{cp} + zT_{cm}} \\ \vdots \\ \frac{r_1 - r_{k+m}}{wT_{cp} + zT_{cm}} \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1
\end{pmatrix}$$

Similarly, other candidates of the timing model can be expressed in the canonical form. The linear programming problem can be defined as the problem of minimizing the objective function, the finish time, $T_{f,n}$ subject to linear constraints. Here, the objective function is

$$T_{f,n} = r_1 + \alpha_1(zT_{cm} + wT_{cp}) \tag{2.28}$$

The constraints are n equalities (2.25) and (2.26) and the bounded nonnegative constraints

$$0 \leq \alpha_i \leq 1 \qquad i = 1, 2, \ldots, n \tag{2.29}$$

From Fig. 2.4, additional $n-1$ inequality constraints for the release times

$$r_i \leq T_{f,n} - \alpha_i(zT_{cm} + wT_{cp}) \qquad i = 2, 3, \ldots, n \tag{2.30}$$

is derived in order to check the feasibility of the release times for the case that mixture of *Constraint I* and *Constraint II* happens. Using (2.28), the inequality constraints, (2.30) can be reformulated as

$$\alpha_i - \alpha_1 \leq \frac{r_1 - r_i}{zT_{cm} + wT_{cp}} \qquad i = 2, 3, \ldots, n \tag{2.31}$$

Another inequality constraint can be obtained from the time consumption for the total load distribution from BCU to all participating processors as

$$r_1 + zT_{cm} \sum_{i=1}^{n} \alpha_i \leq T_{f,n} - \alpha_n wT_{cp} \tag{2.32}$$

Using (2.28), the inequality constraints, (2.32) can be reformulated as

$$\alpha_n wT_{cp} - \alpha_1(zT_{cm} + wT_{cp}) \leq -zT_{cm} \tag{2.33}$$

Based on the n equality constraints (i.e., (2.25) and (2.26) and inequality constraints (i.e., (2.31) and (2.33), the feasibility of the solutions based on the candidates of the timing model can be checked. One critical point here is the feasible solutions obtained by LP based on its own timing model may not satisfy the timing model, ironically. This is because the issue of the release time's feasibility studied in the section of *Constraint II* implicitly arises. To deal with this issue, an exhaustive search algorithm (see Fig. 2.5) to obtain an

optimal timing diagram for a certain network with arbitrary release time is proposed. To clarify the feasible release time issue, it is required to check if the feasible solution obtained by LP satisfies with the timing model for its own solution. If there exists no feasible solutions satisfying own timing model, the algorithm concludes that there is no optimal timing model for the given release times. Then, the candidacy of the n^{th} processor with the largest candidate release time, r_n is rejected. The largest release time, r_n, is regarded as an infeasible release time. Then, the new loop of the algorithm with $n-1$ candidate processors without n^{th} processor will be considered. The algorithm continues running until finding an unique optimal solution. The obtained unique timing model is the optimized model for the minimum finish time with the generalized scenario containing both cases of *Constraint I* and *Constraint II*. Note that the number of candidate processors, n is chosen implicitly based on the n processors's load finish time within the absolute deadline (i.e., $T_{f,n} \leq d$). Naturally this is not a scalable solution-something that awaits further research. One critical drawback of the algorithm is that the algorithm requires high-computational power to consider the possible cases, especially as the number of processors increases and the number of investigated loops increases. Our contribution is that the analysis of the generalized case will provide the flexibility for studying various scenarios with arbitrary release times.

2.3 Performance evaluation

To investigate the behavior of the algorithm, we assume 10 candidate processors are initially given with 10 release times (i.e., $r_1, r_2, \ldots, r_{10}$) generated by the exponential distribution. Simulation is performed 100 times with inverse speed parameters, $wT_{cp} = 1.2$ and $zT_{cm} = 0.8$ sec/load. The sets of deterministic equations obtained from 2^9 possible timing models are solved via MATLAB linear programming function, ***linprog()***. Fig. 2.6 describes the average number of processors actually utilized in the processing. From Fig. 2.6, we can observe that the average number of processors participating in the processing decreases as

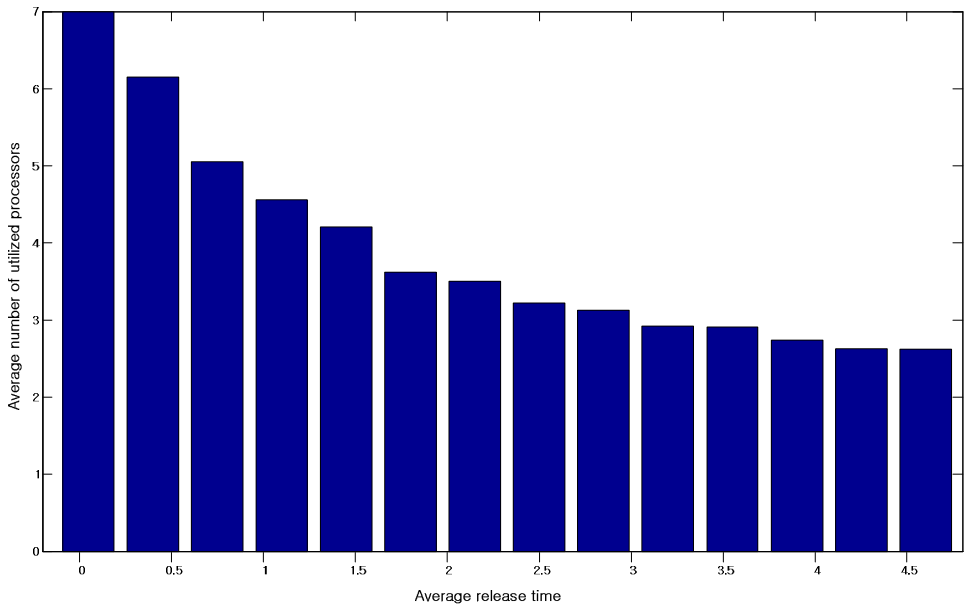


Figure 2.6: Average number of utilized processors vs. average release time

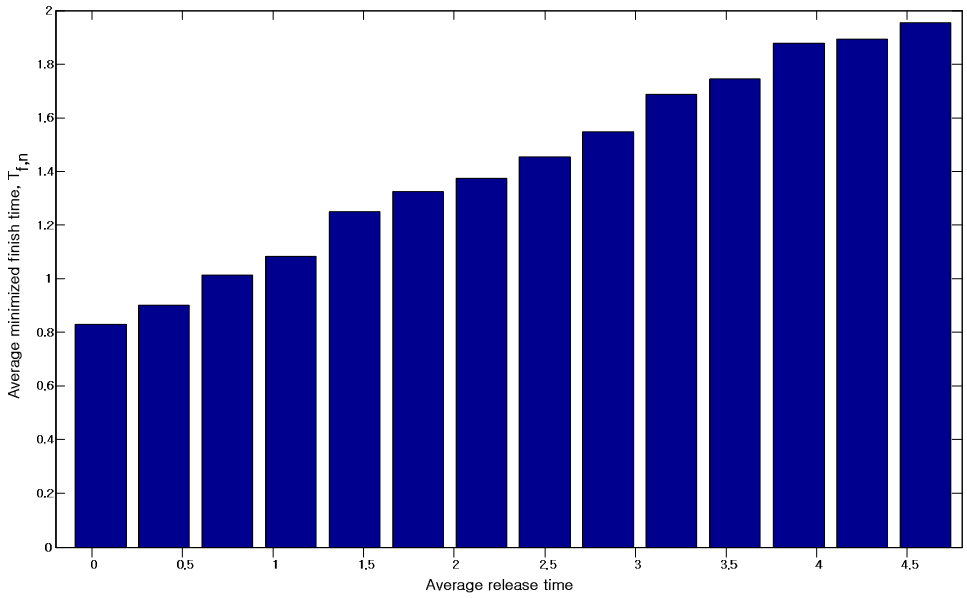


Figure 2.7: Average minimized finish time vs. average release time

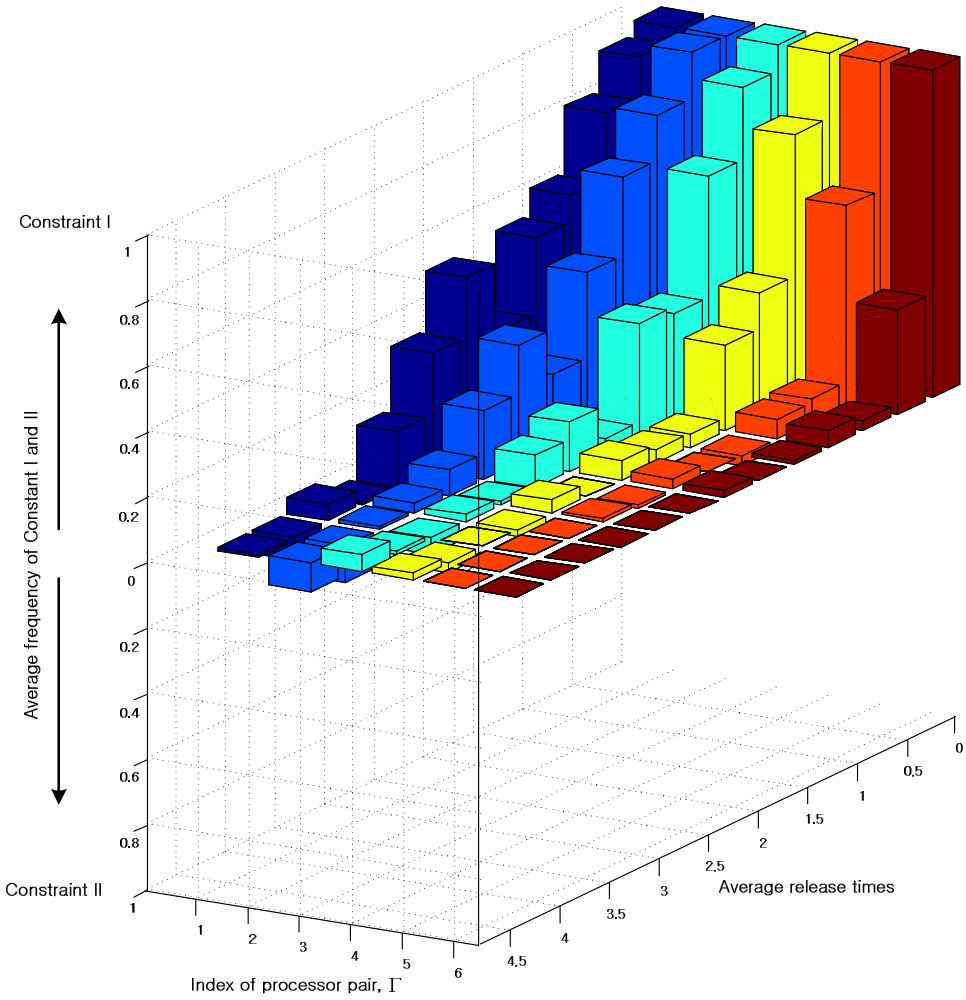


Figure 2.8: Trend of the Constraint I and II

the average release time increases. For the exhaustive search algorithm, the number of candidate processors decreases by one per loop in case that no optimal timing diagram is found. This corresponds the rejection of the candidacy of the largest release time as we mentioned before. It can be expected that the possibility that the release time is infeasible increases as the average release time increases. Thus, intuitively the number of rejected processors increases as the average release time increases.

Fig. 2.7 shows the average finish time performance against the average release time. As the time until the candidate processors are available to receive their own divisible load takes longer, the average time to finish a whole load increases. Also, the decreasing participated processors as the release time increases (see Fig. 2.6) contributes to the longer time spend to end the processing of the whole load.

The occurrence trend of the processor pair satisfying the *Constraint I*, $\Gamma(I)$ and satisfying the *Constraint II*, $\Gamma(II)$ is illustrated in Fig. 2.8. The values on the upper side and lower side show the normalized average occurrence frequency of the $\Gamma(I)$ and $\Gamma(II)$ respectively over the z-axis. It is shown that processor pairs having smaller release time are likely to satisfy the *Constraint I*. That is, the smaller the release time, the more *Constraint I* dominates. It can be expected that the probability that the root processor waits until processors are available to distribute the optimally divided loads to the children processors (i.e., the probability of $\Gamma(II)$) decreases as the release times of the children processors become smaller. The lower index of the processor pair implies the smaller release time as the distribution order of processor (i.e., $r_1 \leq r_2 \leq \ldots \leq r_n$). By similar intuition, it seems reasonable that the network with the higher indexed processor pairs with larger release times experiences the delay shown in the case of the *Constraint II*. From the simulation results, we can observe that the exhaustive search algorithm behaves in a reasonable manner over the generalized models which are a mixture of *Constraint I* and *Constraint II*.

2.4 Concluding remarks

In this chapter, the divisible load distribution problem for a bus network adopting sequential distribution and a staggered start strategy is examined. An analysis was provided based on the real-time assumption that processers in the network have arbitrary release times. Unlike the previous literature, where a distinct scenario of the divisible load distribution is taken into account, we can see that the scenario here involving arbitrary release times can be restricted in two ways. The performance analysis of the network following each of the two constraints exclusively is derived. Based on the analysis of the case of two constraints, we showed an analysis for one of the possible models, mixed two constraints, in a realistic situation. Based on the analysis of a realistic case, we proposed an exhaustive search algorithm for obtaining an optimal solution of the scheduling of divisible loads.

The approach showed in this chapter can initiate a way for solving the divisible load scheduling problem under actual circumstances. It also seems of interest to extend this constraint-based approach to the other network models based on the simultaneous distribution and simultaneous start strategy. One of the most intriguing problems is to develop a computationally efficient heuristic algorithm, which is unlike the one here and scalable, to deal with the exponentially growing possible timing models.

Chapter 3

Divisible Load Scheduling in Clustered Wireless Sensor Networks

In recent years, wireless sensor networks (WSNs) have been a dynamically growing and promising research area due to the great technological progress in the field of wireless communication protocols [33]. Miniaturized low power wireless sensor devices have been used in various potential applications such as for military / aerospace communications and tracking, environmental monitoring, and avionics controls, etc. These sensors, capable of sensing, processing (aggregating) data, and short range communication, form WSNs to report aggregated data across platforms on or above geographically diverse terrain with highly constrained resources.

The applications of sensors in aerospace applications are quite diverse such as pressure sensors, speed sensors, surface temperature sensors, proximity sensors and remote sensors, etc. Such sensor applications operate as a network with various sized independent sensor clusters feeding information to a sink. In sensing, it is inevitable to obtain redundant information such as overlapped data and noninformative data. Considering that an extremely fast response is generally required in aerospace applications, data aggregation in the clusterhead is highly desired for reducing unnecessary load under data communication and computation.

As a WSN configuration, a WSN can be *flat* or *hierarchical* [34]. In flat networks, every

sensor has the same functionality. On the other hand, in hierarchical networks, there are two types of sensor: a cluster head and a non clusterhead sensor. The clusterhead is usually an optimally elected high energy sensor which has a larger role than other sensors [35]. Clusterheads play an important role in data aggregation (data fusion) [36].

As compared with earlier work when most deployed WSNs involved relatively small numbers of sensors, nowadays, WSNs are considered as large scale randomly and densely distributed sensor networks, which do not need infrastructure. Due to the nature of distributed sensing, it is expected that Divisible Load Theory (DLT) could play an important role in providing an optimal solution for load distribution under WSN environments, especially those having various energy and computational constraints.

Recently, studies about DLT based WSN scheduling for data reporting have been published [37],[38]. To our knowledge, the naive data collected by clusterhead from non-clusterhead nodes would not contain the most critical information. For this reason, we introduce a parameter, the information utility constant, showing the accuracy of the collected data from each sensor. From a mathematical point of view, the information utility constant is required to be a deterministic a priori known variable to be applied to DLT. This is feasible by using a technique of information accuracy estimation [39] with a crucial assumption that the clusterhead has an accurate knowledge of position of each sensor nodes in the cluster [40]. By eliminating the noninformative part of the collected data, unexpected communication and processing delays can be reduced. We assume that the aggregation scheme perfectly filters the most critical information so that a clusterhead reports the most critical information to the sink.

The remainder of this chapter is organized as follows. Section 3.1 present the types of notations and analytic background. In section 3.2, four different single cluster WSN scheduling models are analyzed. Muti-cluster WSN scheduling scheme is examined in Section 3.3. Section 3.4 presents performance evaluation curves. Section 3.5 is a conclusion and a discussion of open questions.

3.1 Problem Formulation and Preliminary Remarks

In a hierarchical WSN, the network sensor nodes are partitioned into groups called clusters. A cluster is composed of a single *clusterhead* and sensor nodes (non-clusterhead nodes) in the lowest tier as shown in Fig. 3.1(a). Once a cluster is created, the clusterhead distributes measurement instructions to the sensor nodes deployed in its cluster region. Sensory data from each sensor node is reported to the clusterhead in the higher tier for data aggregation (data fusion). The idea of data aggregation is to gather the data reported from different sensor nodes while eliminating redundancy, minimizing the amount of reporting, and thus achieving energy efficiency [36]. The aggregated data is then transmitted to the high power sink (Base Station, BS) in the highest tier. The BS usually routes the received data via wired infrastructure connected to Internet-based user applications. However, the networking via the wired interface between the BS and end-user is beyond our scope. Thus, the structure of WSN topology discussed in this chapter is a three tier hierarchical WSN.

In this chapter, we ignore the capability of sensing of clusterheads in spite of the fact that a clusterhead is an optimally elected sensor in the cluster region as we mentioned before. Here, there is no direct communication between sensor nodes. In other words, sensor nodes can communicates only through the clusterhead. We will not discuss the case of communication between node sensors in this chapter. However, this important point gives us an idea that the network topology of a cluster can be considered as a centralized single level tree (star) network as shown in Fig. 3.1(a). The communication delay caused by the initial deployment of the measurement instruction from the sink to the clusterhead is ignored in the analysis in this chapter. This is the based on the assumption that channel speed between sink and clusterhead is much faster than channel speed in the cluster and also the amount of instruction data is much smaller compared to the amount of sensory data. In this chapter, we ignore possible limitations subject to various sources of unreliability and other issues related with wireless data transmission such as channel noise and interference during wireless transmission, etc.

The following notation is used in this chapter.

t : Constant time for the instruction assignment from clusterhead to sensor nodes.

α_{si} : The sensory load fraction that the i^{th} sensor node collects.

α_{ti} : The informative load fraction (which is the critical data in response to the instruction) in the sensory load fraction that the i^{th} sensor node collects.

ρ_i : The information utility constant of the i^{th} sensor node: $\rho_i = \frac{\alpha_{ti}}{\alpha_{si}}$, the transmit to sense ratio. The parameter shows the accuracy of the data provided by the i^{th} sensor.

y_i : The inverse sensing speed of the i^{th} sensor node.

z_i : The inverse communication (reporting) speed of the link (channel) between the i^{th} sensor node and the clusterhead. We use index 0 for the clusterhead.

w_0 : The inverse computing speed of the clusterhead.

$T_{so(cp,cm)}$: Sensing operation intensity (Computing intensity, Communication intensity) constant. The entire load can be sensed (processed, transmitted) over the i^{th} channel in time $y_i T_{so}$ ($w_0 T_{cp}$, $z_i T_{cm}$).

w_0^k : The inverse computing (data aggregation) speed of the k^{th} clusterhead (where $k = 1, 2, \ldots, n$). Note that we use w_0 for analyzing a single cluster topology in this chapter.

z_0^k : The inverse communication (reporting) speed of the link (channel) between the k^{th} clusterhead and a sink. Note that we use z_0 for analyzing a single cluster topology in this chapter.

T_i : The total time that elapses between the beginning of the process at $t = 0$ and the time when the i^{th} sensor node completes the report of its own sensing data to the clusterhead (where $i = 1, 2, \ldots, n$). We use index 0 for the clusterhead.

$T_{r,n}$: The round time. Time at which the clusterhead for n sensors finishes transmitting the aggregated data to the sink ($=T_0$).

$T_{r,1}$: The round time. Time at which the clusterhead in a cluster with a single sensor node finishes transmitting the aggregated data to the sink.

We will distinguish in the following sections between different channel characteristic ($Single$, $Multi$) and whether either is equipped with front-end processor ($Simultaneous$ reporting)

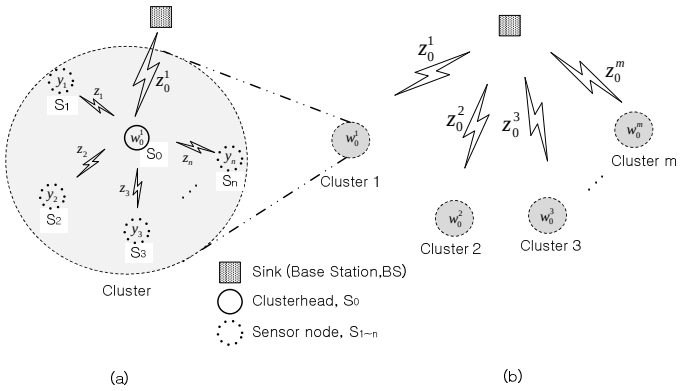


Figure 3.1: Three tier hierarchical wireless sensor network topology.

or not (*Sequential* reporting). There are thus four scheduling scenarios with these two sets of possible features.

3.2 Single Cluster based hierarchical WSN scheduling

3.2.1 Single Channel with no front-end processor, SCnP

Consider a single channel cluster composed of single clusterhead and n sensor nodes in the wireless network (star topology). This is a simple network scenario where the n sensor nodes $S_1, S_2, \ldots, S_n$ can report their own sensory data directly to the clusterhead, S_0 via a single channel. Conversely, the clusterhead can assign measurement instructions directly to the n sensor nodes via a single channel. Since all of the data flows through the single channel, sensing instruction assignment and reporting are performed sequentially as shown in Fig. 3.2. In this section, the clusterhead and sensor nodes are not equipped with front-end processors for communication off-loading. That is, nodes can either communicate (report) or sense but not do both at the same time (i.e., staggered start). In the same manner, the data aggregation is performed only after the last report from the sensors terminates, and the data aggregation and its reporting to the sink are also performed sequentially. Each sensor node has a capability to start sensing as soon as the sensor receives its sensing instructions from the sink node. The placement of nodes $S_0, S_1, S_2, \ldots$,and S_n is unconstrained. Initially,

we consider that the position of the clusterhead is optimized.

From the timing diagram (Fig. 3.2), one can set up the following corresponding recursive load distribution equations

$$\alpha_{si} y_i T_{so} = t + \alpha_{si+1}(y_{i+1} T_{so} + z_{i+1} T_{cm})$$
$$i = 1, 2, \ldots, n-1 \tag{3.1}$$

Rewriting the above set of equations as

$$\alpha_{si+1} = f_i \alpha_{si} - g_i \qquad i = 1, 2, \ldots, n-1 \tag{3.2}$$

where

$$f_i = \frac{y_i T_{so}}{y_{i+1} T_{so} + z_{i+1} T_{cm}}, \quad g_i = \frac{t}{y_{i+1} T_{so} + z_{i+1} T_{cm}}$$
$$i = 1, 2, \ldots, n-1 \tag{3.3}$$

These equations can be solved as follows:

$$\begin{aligned}
\alpha_{s2} &= f_1 \alpha_{s1} - g_1 \\
\alpha_{s3} &= f_2 \alpha_{s2} - g_2 = f_1 f_2 \alpha_{s1} - g_1 f_2 - g_2 \\
\alpha_{s4} &= f_3 \alpha_{s3} - g_3 = f_1 f_2 f_3 \alpha_{s1} - g_1 f_2 f_3 - g_2 f_3 - g_3 \\
&\vdots \\
\alpha_{sn} &= f_{n-1} \alpha_{sn-1} - g_{n-1} \\
&= f_1 f_2 \ldots f_{n-1} \alpha_{s1} - g_1 f_2 f_3 \ldots f_{n-1} - g_2 f_3 f_4 \ldots f_{n-1} - \ldots - g_{n-2} f_{n-1} - g_{n-1}
\end{aligned} \tag{3.4}$$

alternatively,

$$\alpha_{si} = \alpha_{s1} \prod_{k=1}^{i-1} f_k - \sum_{k=1}^{i-1} \frac{g_k}{f_k} \prod_{l=k}^{i-1} f_l \qquad i = 2, 3, \ldots, n \tag{3.5}$$

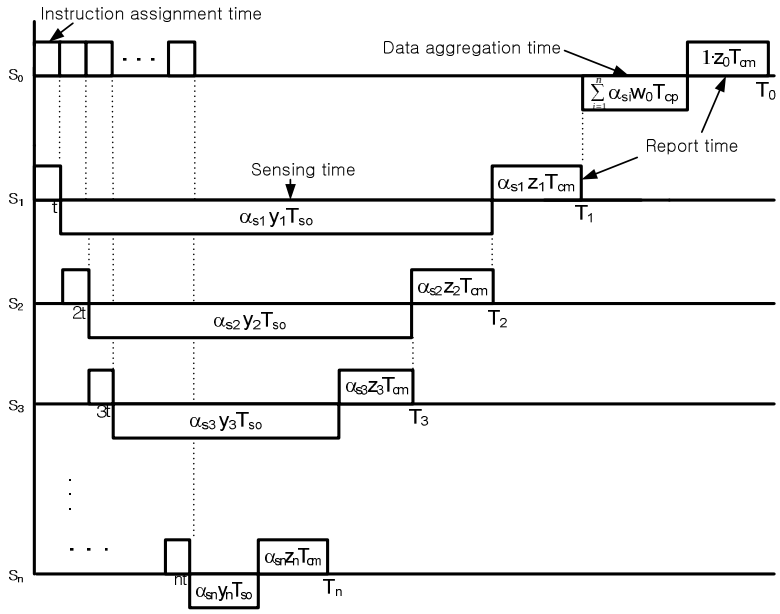


Figure 3.2: Timing diagram for single channel hierarchical wireless sensor network with no front-end processors.

or by definition of the information utility constant

$$\alpha_{ti} = \rho_i \alpha_{si} = \alpha_{s1} \rho_i \prod_{k=1}^{i-1} f_k - \rho_i \sum_{k=1}^{i-1} \frac{g_k}{f_k} \prod_{l=k}^{i-1} f_l \qquad i = 2, 3, \ldots, n \tag{3.6}$$

Since the fractions of total informative data, α_ts should sum to one (normalization),

$$1 = \sum_{i=1}^{n} \alpha_{ti} = \sum_{i=1}^{n} \rho_i \alpha_{si} = \alpha_{s1} \left(\rho_1 + \sum_{i=2}^{n} \rho_i \prod_{k=1}^{i-1} f_k \right) - \sum_{i=2}^{n} \rho_i \sum_{k=1}^{i-1} \frac{g_k}{f_k} \prod_{l=k}^{i-1} f_l \tag{3.7}$$

Therefore,

$$\alpha_{t1} = \rho_1 \alpha_{s1} = \rho_1 \frac{1 + C_1}{\rho_1 + C_2} \tag{3.8}$$

where

$$C_1 = \sum_{i=2}^{n} \rho_i \left(\sum_{k=1}^{i-1} \frac{g_k}{f_k} \prod_{l=k}^{i-1} f_l \right), \; C_2 = \sum_{i=2}^{n} \rho_i \prod_{k=1}^{i-1} f_k \tag{3.9}$$

From the (3.6) and (3.8), the optimal values of α_t's, which is certain fraction of informative data collected during measurement time can be obtained. Here, we say that the network completes a *round* when the clusterhead finishes reporting the aggregated data to the sink node. Referring the Fig. 3.2, the minimum round time of the network can be achieved using (3.5) as follows

$$\begin{aligned} T_{r,n} &= T_0 = T_1 + \sum_{i=1}^{n} \alpha_{si} w_0 T_{cp} + 1 \cdot z_0 T_{cm} \\ &= t + \alpha_{s1}(y_1 T_{so} + z_1 T_{cm}) + \left\{ \alpha_{s1} \left(1 + \sum_{i=2}^{n} \prod_{k=1}^{i-1} f_k \right) - \sum_{i=2}^{n} \sum_{k=1}^{i-1} \frac{g_k}{f_k} \prod_{l=k}^{i-1} f_l \right\} w_0 T_{cp} + z_0 T_{cm} \end{aligned} \tag{3.10}$$

From (3.8), $T_{r,n}$ can be rewritten as follow

$$T_{r,n} = t + \frac{1 + C_1}{\rho_1 + C_2}(y_1 T_{so} + z_1 T_{cm}) + C_3 w_0 T_{cp} + z_0 T_{cm} \tag{3.11}$$

Here,

$$C_3 = \frac{(1 + C_1)(1 + C_2')}{\rho_1 + C_2} - C_1' \tag{3.12}$$

where,

$$C_1' = \sum_{i=2}^{n} \sum_{k=1}^{i-1} \frac{g_k}{f_k} \prod_{l=k}^{i-1} f_l, \;\; C_2' = \sum_{i=2}^{n} \prod_{k=1}^{i-1} f_k \tag{3.13}$$

As a special case, consider the situation of a homogeneous cluster where all sensor nodes have same inverse sensing speed and inverse communication speed (i.e., $y_i = y$, $z_i = z$ for $i = 1, 2, \ldots, n$). Note here that the inverse communication speed of the clusterhead, z_0 can be different. Consequently,

$$f = \frac{y T_{so}}{y T_{so} + z T_{cm}}, \;\; g = \frac{t}{y T_{so} + z T_{cm}} \tag{3.14}$$

Here, $0 < f < 1$. Then, the (3.9) can be modified as

$$C_1 = \frac{g}{1-f}\sum_{i=2}^{n}\rho_i(1-f^{i-1}),\ C_2 = \sum_{i=2}^{n}\rho_i f^{i-1} \tag{3.15}$$

Hence, under the homogeneous condition, $T_{r,n}$ can be obtained as follow

$$T_{r,n} = t + \frac{1+C_1}{\rho_1 + C_2}(yT_{so} + zT_{cm}) + C_3 w_0 T_{cp} + z_0 T_{cm} \tag{3.16}$$

Here, C_3 is identical form with (3.12), where

$$\begin{aligned} C_1' &= \frac{g}{f}\sum_{i=2}^{n}\left(\sum_{k=1}^{i-1}\prod_{l=k}^{i-1} f\right) = \frac{g}{f}\sum_{i=2}^{n}\sum_{k=1}^{i-1} f^{i-1-k} = \frac{g}{1-f}\left\{(n-1) - \frac{f-f^n}{1-f}\right\} \\ C_2' &= \sum_{i=2}^{n}\prod_{k=1}^{i-1} f = \sum_{i=1}^{n-1} f^i = \left(\frac{f-f^n}{1-f}\right) \end{aligned} \tag{3.17}$$

For the further simplification, here we add another special condition, $\rho_i = \rho$ for $i = 1, 2, \ldots, n$, the homogeneous information utility constant. Since $C_1 = \rho C_1'$, $C_2 = \rho C_2'$ (i.e., $C_3 = \frac{1}{\rho}$ from (3.12)) under the special case of the homogeneous (fully homogeneous, i.e. $y_i = y$, $z_i = z$, $\rho_s = \rho$ and $\rho_i = \rho$ for $i = 1, 2, \ldots, n$) condition one can solve for α_{s1} as

$$\alpha_{s1} = \frac{\frac{1}{\rho} + C_1'}{1 + C_2'} \tag{3.18}$$

Hence, $T_{r,n}$ can be obtained as follow

$$T_{r,n} = t + \frac{\frac{1}{\rho} + C_1'}{1 + C_2'} yT_{so} + \frac{\frac{1}{\rho} + C_1'}{1 + C_2'} zT_{cm} + \frac{1}{\rho} w_0 T_{cp} + z_0 T_{cm} \tag{3.19}$$

From the above equation, under the fully homogeneous condition, we can readily see the minimum round time can be achieved when information utility constant equals to unity which is an ideal case. This makes intuitive sense as no redundant data is generated, time delays for sensing, reporting, and data aggregation can be reduced.

Since the minimum round time on a single sensor node under the homogeneous condition is

$$T_{r,1} = t + \frac{1}{\rho} y T_{so} + \frac{1}{\rho} z T_{cm} + \frac{1}{\rho} w_0 T_{cp} + z_0 T_{cm} \tag{3.20}$$

the speedup is then

$$Speedup = \frac{T_{r,1}}{T_{r,n}} = \frac{t + \frac{1}{\rho} y T_{so} + \frac{1}{\rho} z T_{cm} + \frac{1}{\rho} w_0 T_{cp} + z_0 T_{cm}}{t + \frac{\frac{1}{\rho}+C_1'}{1+C_2'} y T_{so} + \frac{\frac{1}{\rho}+C_1'}{1+C_2'} z T_{cm} + \frac{1}{\rho} w_0 T_{cp} + z_0 T_{cm}} \tag{3.21}$$

Under the fully homogeneous condition, the condition for a feasible instruction assignment time, t can be described. From the Fig. 2, under the case of the sequential distribution under the single channel scenario, it can be possible some sensors cannot be assigned the instruction under the certain condition that the instruction assignment time is large. Referring to (3.5), the upper bound of a feasible instruction assignment time constant can be obtained according to the intuitive condition as

$$\begin{aligned} & 0 < \alpha_{si} < 1 \qquad i = 2, 3, \ldots, n \\ & < \alpha_{s1} f^{i-1} - \frac{g}{f}\left(\frac{f - f^{i-1}}{1-f}\right) < 1 \\ & < \frac{\frac{1}{\rho} + C_1'}{1 + C_2'} f^{i-1} - \frac{g}{f}\left(\frac{f - f^{i-1}}{1-f}\right) < 1 \\ & < \frac{\frac{1}{\rho}}{1 + C_2'} f^{i-1} + \left\{ \frac{C_1''}{1 + C_2'} f^{i-1} + \frac{1}{y T_{so}}\left(\frac{f - f^{i-1}}{1-f}\right) \right\} t < 1 \end{aligned} \tag{3.22}$$

where, $C_1'' = C_1'/t$ (refer to (3.17)).

Thus, we have

$$0 < \eta_i + \gamma_i t < 1 \qquad i = 2, 3, \ldots, n \tag{3.23}$$

where,

$$\eta_i = \frac{\frac{1}{\rho}}{1+C_2'} f^{i-1}, \quad \gamma_i = \frac{C_1''}{1+C_2'} f^{i-1} + \frac{1}{yT_{so}} \left(\frac{f - f^{i-1}}{1-f} \right) \tag{3.24}$$

Since, $\eta_i > 0$, if $\gamma_i < 0$, the (3.23) can be rewritten as follows

$$0 \leq t < \frac{\eta_k}{\gamma_k} \tag{3.25}$$

Here, k is the index of the sensor node number where $\gamma_i < 0$. If $\gamma_i > 0$, the (3.23) can be rewritten as follows

$$0 \leq t < \frac{1-\eta_l}{\gamma_l} \tag{3.26}$$

Here, l is the index of the sensor node number where $\gamma_i > 0$. From the (3.25) and (3.26), the condition for the feasible instruction assignment time, t_{feas} can be written as

$$0 \leq t_{feas} < min(\frac{\eta_k}{\gamma_k}, \frac{1-\eta_l}{\gamma_l}) \tag{3.27}$$

Here, if one let $n \to \infty$, we can superficially recognize that $T_{r,n} \to \infty$, which is not reasonable since the factor, C_1' is a the first order function of n (see (3.17)). However, as referring the bound for feasible t (see (3.25) and (3.26)), we would also see $t \to 0$ (i.e., $C_1'' \to \infty$ or $\gamma_i \to \infty$) as one is adding more sensor nodes in the cluster.

Hence, using the fact $C_1' \to 0$ and $C_2' \to \frac{yT_{so}}{zT_{cm}}$, one can obtain the asymptotic minimum round time, $T_{r,\infty}$ as

$$\begin{aligned} T_{r,\infty} &= \frac{\sigma}{\rho(\sigma+1)} yT_{so} + \frac{\sigma}{\rho(\sigma+1)} zT_{cm} + \frac{1}{\rho} w_0 T_{cp} + z_0 T_{cm} \\ &= \frac{1}{\rho} zT_{cm} + \frac{1}{\rho} w_0 T_{cp} + z_0 T_{cm} \end{aligned} \tag{3.28}$$

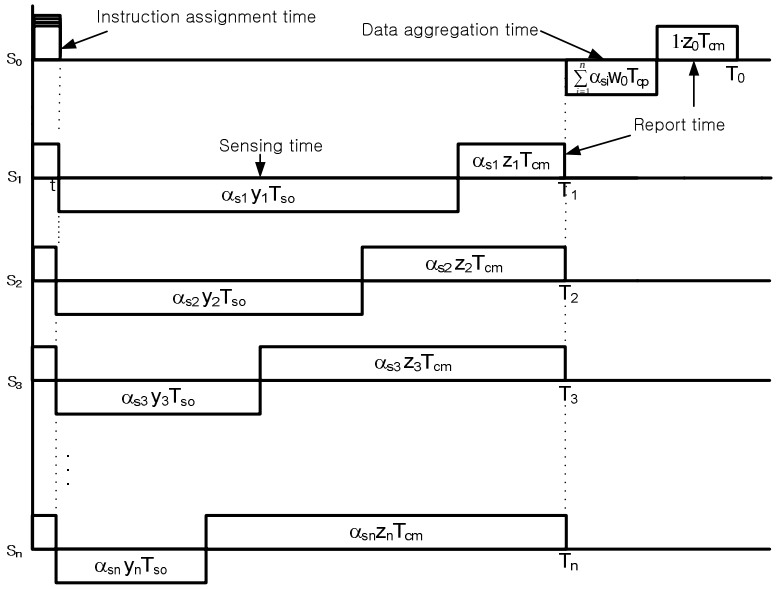


Figure 3.3: Timing diagram for multi channel hierarchical wireless sensor network with no front-end processors.

where $\sigma = \frac{zT_{cm}}{yT_{so}}$

From (3.21), we can obtain asymptotic speedup as follows

$$Speedup_{n\to\infty} = 1 + \frac{yT_{so}}{zT_{cm} + w_0 T_{cp} + \rho z_0 T_{cm}} \tag{3.29}$$

3.2.2 Multi Channel with no front-end processor, MCnP

The network model that is discussed in this subsection is similar to that discussed in the previous one except for the fact that the communication between the clusterhead and sensor nodes takes place under multiple independent channels. Therefore, multiple access to the clusterhead from the sensor nodes can take place at the same time. Conversely, concurrent instruction assignment is feasible as described in Fig. 3.3. As a reminder, the single channel case gives rises to considerable idle time for almost all of the sensor nodes due to the sequential communication (instruction assignment and reporting) involved in communicating data from each sensor node to the clusterhead. On the other hand, communication

delay is reduced through simultaneous instruction assignment and mainly simultaneous reporting termination over all sensor nodes through multiple channels. It has been known on a intuitive basis that network elements should be kept constantly busy for good performance [17]. Thus it can be expected that multi channel communication contributes to better performance than the single channel scenario in that each sensor finish the reporting simultaneously.

The timing diagram of the network is plotted in Fig. 3.3. The fundamental recursive equation of the network can be formulated as follows

$$t + \alpha_{si}(y_i T_{so} + z_i T_{cm}) = t + \alpha_{si+1}(y_{i+1} T_{so} + z_{i+1} T_{cm}) \qquad i = 1, 2, \ldots, n-1 \tag{3.30}$$

Rewriting the above set of equations as

$$\alpha_{si+1} = f_i \alpha_{si} \qquad i = 1, 2, \ldots, n-1 \tag{3.31}$$

where

$$f_i = \frac{y_i T_{so} + z_i T_{cm}}{y_{i+1} T_{so} + z_{i+1} T_{cm}} \qquad i = 1, 2, \ldots, n-1 \tag{3.32}$$

These equations can be solved as follows

$$\alpha_{si} = \alpha_{s1} \prod_{k=1}^{i-1} f_k \qquad i = 2, 3, \ldots, n \tag{3.33}$$

Alternatively, by definition of information the utility constant

$$\alpha_{ti} = \rho_i \alpha_{si} = \alpha_{s1} \rho_i \prod_{k=1}^{i-1} f_k \qquad i = 2, 3, \ldots, n \tag{3.34}$$

As mentioned earlier, the fractions of the total informative data should sum to one

$$1 = \sum_{i=1}^{n} \alpha_{ti} = \sum_{i=1}^{n} \rho_i \alpha_{si} = \alpha_{s1} \left(\rho_1 + \sum_{i=2}^{n} \rho_i \prod_{k=1}^{i-1} f_k \right) \tag{3.35}$$

Therefore,

$$\alpha_{t1} = \rho_1 \alpha_{s1} = \rho_1 \frac{1}{\rho_1 + C_1} \tag{3.36}$$

where

$$C_1 = \sum_{i=2}^{n} \rho_i \prod_{k=1}^{i-1} f_k \tag{3.37}$$

From (3.34) and (3.36), the optimal values of α_t's, which is the fraction of informative data collected during measurement time can be obtained.

Knowing the optimal value of α_{s1}, the minimum round time can be calculated as

$$\begin{aligned} T_{r,n} = T_0 &= T_1 + \sum_{i=1}^{n} \alpha_{si} w_0 T_{cp} + 1 \cdot z_0 T_{cm} \\ &= t + \alpha_{s1}(y_1 T_{so} + z_1 T_{cm}) + \alpha_{s1} \left(1 + \sum_{i=2}^{n} \prod_{k=1}^{i-1} f_k \right) w_0 T_{cp} + z_0 T_{cm} \end{aligned} \tag{3.38}$$

From $\alpha_{s1} = \frac{1}{\rho_1 + C_1}$ (see (3.36)), $T_{r,n}$ can be rewritten as follow

$$T_{r,n} = t + \frac{1}{\rho_1 + C_1}(y_1 T_{so} + z_1 T_{cm}) + C_3 w_0 T_{cp} + z_0 T_{cm} \tag{3.39}$$

where

$$C_1' = \sum_{i=2}^{n} \prod_{k=1}^{i-1} f_k, \;\; C_3 = \frac{1 + C_1'}{\rho_1 + C_1} \tag{3.40}$$

As a special case, consider the situation of a homogeneous cluster where all sensor nodes have same inverse sensing speed and inverse communication speed (i.e., $y_i = y$, $z_i = z$ for

$i = 1, 2, \ldots, n$). Note here that the inverse communication speed of the clusterhead, z_0 can be different. Consequently, $f = 1$ (i.e., $C_1 = \sum_{i=2}^{n} \rho_i$ and $C_1' = n - 1$).

In this case, one can solve for α_{s1} as

$$\alpha_{s1} = \frac{1}{\sum_{i=1}^{n} \rho_i} \tag{3.41}$$

The minimum round time is then given by

$$T_{r,n} = t + \frac{1}{\sum_{i=1}^{n} \rho_i}(yT_{so} + zT_{cm} + nw_0T_{cp}) + z_0T_{cm} \tag{3.42}$$

For further simplification, consider the situation of a fully homogeneous network (i.e. $y_i = y$, $z_i = z$, and $\rho_i = \rho$ for $i = 1, 2, \ldots, n$). Consequently, $f = 1$ (i.e., $C_1 = \rho(n - 1)$ and $C_1' = n - 1$).

In this case, one can solve for α_{s1} as

$$\alpha_{s1} = \frac{1}{\rho n} \tag{3.43}$$

Intuitively, the equal amount of informative (sensed) data (i.e., $\frac{1}{n}$) is reasonable due to the simultaneous measurement start time and reporting finish time under the fully homogeneous condition.

The minimum round time is then given by

$$T_{r,n} = t + \frac{1}{\rho n}yT_{so} + \frac{1}{\rho n}zT_{cm} + \frac{1}{\rho}w_0T_{cp} + z_0T_{cm} \tag{3.44}$$

From the above equation, under the fully homogeneous condition, we can also see the minimum round time can be achieved when information utility constant equals to unity which is an ideal case. This also follows intuition as no redundant data is generated, time delays for sensing, reporting, and data aggregation can be reduced.

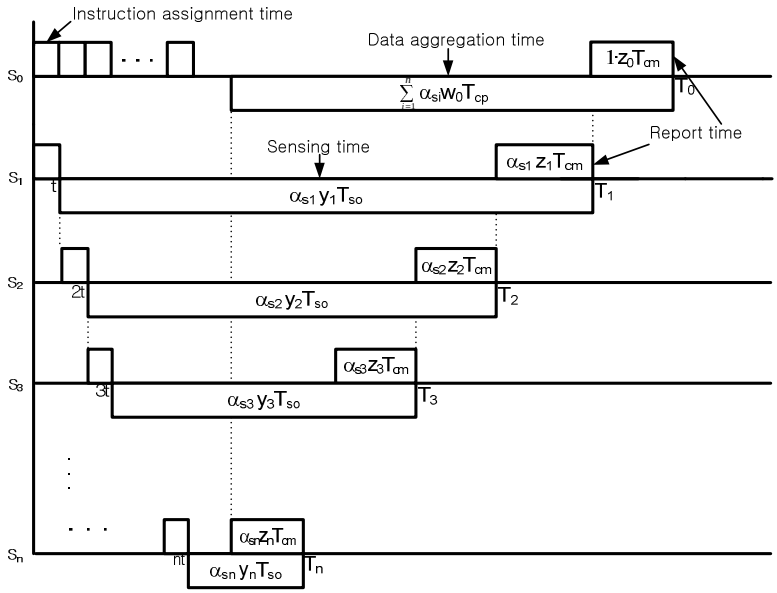


Figure 3.4: Timing diagram for single channel hierarchical wireless sensor network with front-end processors.

Since the minimum round time on a single sensor node under the homogeneous condition is

$$T_{r,1} = t + \frac{1}{\rho} y T_{so} + \frac{1}{\rho} z T_{cm} + \frac{1}{\rho} w_0 T_{cp} + z_0 T_{cm} \tag{3.45}$$

the speedup is then

$$Speedup = \frac{T_{r,1}}{T_{r,n}} = \frac{t + \frac{1}{\rho} y T_{so} + \frac{1}{\rho} z T_{cm} + \frac{1}{\rho} w_0 T_{cp} + z_0 T_{cm}}{t + \frac{1}{\rho n} y T_{so} + \frac{1}{\rho n} z T_{cm} + \frac{1}{\rho} w_0 T_{cp} + z_0 T_{cm}} \tag{3.46}$$

The asymptotic minimum round time, $T_{r,\infty}$ can be simply written as

$$T_{r,\infty} = t + \frac{1}{\rho} w_0 T_{cp} + z_0 T_{cm} \tag{3.47}$$

The asymptotic speedup can be obtain as follows

$$Speedup_{n\to\infty} = 1 + \frac{y T_{so} + z T_{cm}}{\rho t + w_0 T_{cp} + \rho z_0 T_{cm}} \tag{3.48}$$

3.2.3 Single Channel with front-end processor, SCP

The network model that is discussed in this subsection is similar to that discussed in subsection 3.2.1 except for the fact that each of n sensor nodes and clusterhead are equipped with a front-end processor for communicating off-loading. That is, the sensor nodes can communicate (report) and sense at the same time. The clusterhead can perform the data aggregation as it gathers sensory data from each sensor node. We assume here, the data aggregation is started immediately after the first sensory data is reported as shown in Fig. 3.4. Intuitively, the reporting time lasts at least when sensing operation terminates. In this subsection, we assume that $z_i T_{cm} < y_i T_{so}$, so that reporting of sensory data at each sensor node can end with sensing operation at the same instant even though the i^{th} reporting starts when the $(i+1)^{th}$ sensor nodes terminates its own reporting (Fig. 3.4). Surely, each report can be started only after the sensing operation is started. For the clusterhead, we can conjecture $\sum_{i=1}^{n} \alpha_{si} w_0 T_{cp} - z_0 T_{cm} \leq \sum_{i=1}^{n} \alpha_{si} z_i T_{cm}$ to meet the minimum round time. Here, we can see that the conjecture also implies $z_0 T_{cm} \leq \sum_{i=1}^{n} \alpha_{si} w_0 T_{cp}$ so that reporting to sink (BS) and data aggregation end at the same time.

According to Fig. 3.4, the fundamental recursive equation of the network can be obtained as follows:

$$\alpha_{si} y_i T_{so} = t + \alpha_{si+1} y_{i+1} T_{so} + \alpha_{si} z_i T_{cm} \qquad i = 1, 2, \ldots, n-1 \tag{3.49}$$

One can rewriting the above set of equations as

$$\alpha_{si+1} = f_i \alpha_{si} - g_i \qquad i = 1, 2, \ldots, n-1 \tag{3.50}$$

Here, the above equation has the same formation with (3.2), but

$$f_i = \frac{y_i T_{so} - z_i T_{cm}}{y_{i+1} T_{so}}, \quad g_i = \frac{t}{y_{i+1} T_{so}} \qquad i = 1, 2, \ldots, n-1 \tag{3.51}$$

As we mentioned previously, $z_i T_{cm} < y_i T_{so}$. That is, communication speed must be faster than sensing speed. The optimal values of α_ts can be obtained using eq (3.6) and (3.8). Referring the Fig. 3.4, the minimum round time, $T_{r,n}$ can be then obtained as follows

$$T_{r,n} = min(T_0) = T_1 + z_0 T_{cm} = t + \alpha_{s1} y_1 T_{so} + z_0 T_{cm} \tag{3.52}$$

Note that $min(T_0)$ is derived according to the condition, $\sum_{i=1}^{n} \alpha_{si} w_0 T_{cp} - z_0 T_{cm} = \sum_{i=1}^{n} \alpha_{si} z_i T_{cm}$ as we mentioned previously.

Using eq (3.8) and (3.9), $T_{r,n}$ can be rewritten as follow

$$T_{r,n} = t + \frac{1 + C_1}{\rho_1 + C_2} y_1 T_{so} + z_0 T_{cm} \tag{3.53}$$

As a special case, consider the situation of a fully homogeneous network (i.e. $y_i = y$, $z_i = z$, and $\rho_i = \rho$ for $i = 1, 2, \ldots, n$). Note here that the inverse communication speed of the clusterhead, z_0 can be different.

Consequently,

$$f = 1 - \frac{z T_{cm}}{y T_{so}}, \; g = \frac{t}{y T_{so}} \tag{3.54}$$

Here, $y T_{so} > z T_{cm}$. Thus $0 < f < 1$.

Also, under the fully homogeneous condition the initial two conjectures we mentioned previously,

$\sum_{i=1}^{n} \alpha_{si} w_0 T_{cp} - z_0 T_{cm} \leq \sum_{i=1}^{n} \alpha_{si} z_i T_{cm}$ and $z_0 T_{cm} \leq \sum_{i=1}^{n} \alpha_{si} w_0 T_{cp}$, can be combined as follows

$$\rho \leq \frac{w_0 T_{cp}}{z_0 T_{cm}} \leq \rho + \frac{z T_{cm}}{z_0 T_{cm}} \tag{3.55}$$

The above inequality gives an information of the feasible range of the ratio of $\frac{w_0 T_{cp}}{z_0 T_{cm}}$ to meet the minimum round time.

Following similar procedures showed previous subsections, α_{s1} can be obtained as a identical form with (3.18).

The minimum round time is then given by

$$T_{r,n} = t + \frac{\frac{1}{\rho} + C_1'}{1 + C_2'} y T_{so} + z_0 T_{cm} \tag{3.56}$$

From the above equation, under a fully homogeneous condition, we can also see the minimum round time can be achieved when information utility constant equals to unity which is an ideal case.

Under the fully homogeneous condition, a closed form of the condition for a feasible instruction assignment time, t_{feas} can be derived as the similar manner shown in subsection 3.2.1.

$$0 \leq t_{feas} < min(\frac{\eta_k}{\gamma_k}, \frac{1 - \eta_l}{\gamma_l}) \tag{3.57}$$

where,

$$\eta_i = \frac{\frac{1}{\rho}}{1 + C_2'} f^{i-1}, \ \gamma_i = \frac{C_1''}{1 + C_2'} f^{i-1} + \frac{1}{y T_{so} - z T_{cm}} \left(\frac{f - f^{i-1}}{1 - f} \right) \tag{3.58}$$

Here, k and l are the index of the sensor node number where $\gamma_i < 0$ and $\gamma_i > 0$, respectively. From the above equation, under fully homogeneous condition, we can also see the minimum round time can be achieved when information utility constant equals to unity which is an ideal case.

Since the minimum round time on a single sensor node under the fully homogeneous condition is

$$T_{r,1} = t + \frac{1}{\rho} y T_{so} + z_0 T_{cm} \tag{3.59}$$

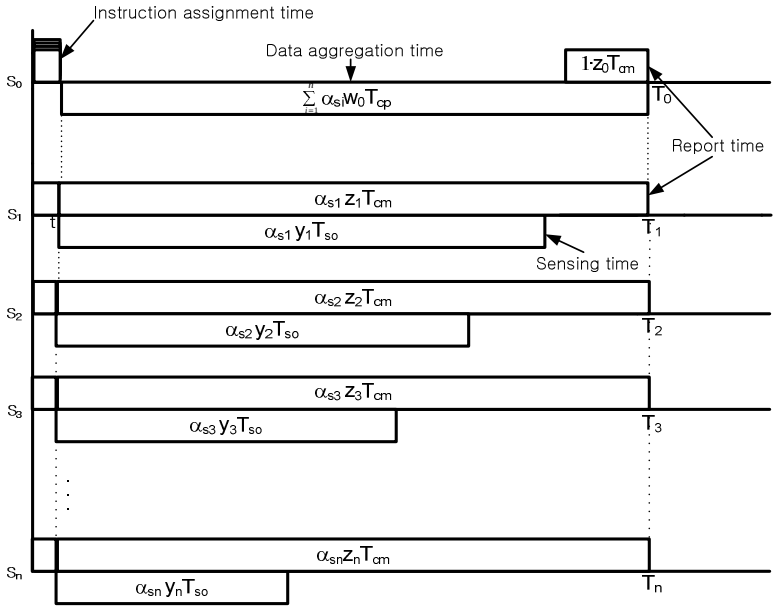


Figure 3.5: Timing diagram for multi channel hierarchical wireless sensor network with front-end processors.

the speedup is then

$$Speedup = \frac{T_{r,1}}{T_{r,n}} = \frac{t + \frac{1}{\rho} y T_{so} + z_0 T_{cm}}{t + \frac{\frac{1}{\rho} + C_1'}{1 + C_2'} y T_{so} + z_0 T_{cm}} \tag{3.60}$$

From the (3.56), using the fact, $C_1' \to 0$ (i.e., $t \to 0$) and $C_2' \to \frac{yT_{so}}{zT_{cm}} - 1$ as $n \to \infty$, the asymptotic minimum round time, $T_{r,\infty}$ can be simply written as

$$T_{r,\infty} = \frac{\sigma}{\rho} y T_{so} + z_0 T_{cm} = \frac{1}{\rho} z T_{cm} + z_0 T_{cm} \tag{3.61}$$

Here, $\sigma = \frac{zT_{cm}}{yT_{so}}$.

The asymptotic speedup can be then obtained as follows

$$Speedup_{n \to \infty} = \frac{yT_{so} + \rho z_0 T_{cm}}{zT_{cm} + \rho z_0 T_{cm}} \tag{3.62}$$

3.2.4 Multi Channel with front-end processor, MCP

This subsection is similar to the previous one except for the fact that now the communication between sink node and sensor nodes takes place under multiple independent channels as described in subsection 3.2.2, so that simultaneous reporting is possible. It was shown that the simultaneous sensing operation start and simultaneous reporting termination over all sensor nodes provided through multiple independent channels contribute improved performance in subsection 3.2.1. Furthermore, more improvement in performance would be achieved from the simultaneous reporting termination over all nodes include the clusterhead with front-end processor. The timing diagram of the network is plotted in Fig. 3.5. We assume here that $z_i T_{cm} > y_i T_{so}$, so that the speed of sensing operation is faster than the speed of reporting. In other words, by intuitive sense, the sensing operation is required to be ceased before or exactly when the reporting terminates. For the clusterhead, we can conjecture $\sum_{i=1}^{n} \alpha_{si} w_0 T_{cp} \leq \alpha_{si} z_i T_{cm}$ to meet the minimum round time from in Fig. 3.5. Also, we can conjecture that $z_0 T_{cm} \leq \sum_{i=1}^{n} \alpha_{si} w_0 T_{cp}$ so that reporting to sink (BS) and data aggregation end at the same time. From the conjecture, we can expect that the minimum T_0 can be obtained under a certain condition, $\sum_{i=1}^{n} \alpha_{si} w_0 T_{cp} = \alpha_{si} z_i T_{cm}$.
The fundamental recursive equation which is independent of the information utility, ρ can be formulated as follows:

$$\alpha_{si} z_i T_{cm} = \alpha_{si+1} z_{i+1} T_{cm} \qquad i = 1, 2, \ldots, n-1 \tag{3.63}$$

One can rewriting the above set of equations as

$$\alpha_{si+1} = f_i \alpha_{si} \qquad i = 1, 2, \ldots, n-1 \tag{3.64}$$

Here, the above equation has the same formation with (3.31), but

$$f_i = \frac{z_i T_{cm}}{z_{i+1} T_{cm}} \qquad i = 1, 2, \ldots, n-1 \tag{3.65}$$

Thus, the optimal values of α_is can be obtained as (3.34) using the identical equation, (3.36). Referring Fig. 3.5, the minimum round time, $T_{r,n}$ can be obtained as follows

$$T_{r,n} = min(T_0) = T_1 = t + \alpha_{s1} z_1 T_{cm} \tag{3.66}$$

Using the (3.36), $T_{r,n}$ can be rewritten as follow

$$T_{r,n} = t + \frac{1}{\rho_1 + C_1} z_1 T_{cm} \tag{3.67}$$

As a special case, consider the situation of a fully homogeneous network (i.e. $y_i = y$, $z_i = z$, and $\rho_i = \rho$ for $i = 1, 2, \ldots, n$). Note here that the inverse communication speed of the clusterhead, z_0 can be different.

Since $f = 1$ (i.e., $C_1 = \rho(n-1)$), α_{s1} can be obtained as the equal amount of sensed data shown as (3.43).

From the two initial conjecture, $\sum_{i=1}^{n} \alpha_{si} w_0 T_{cp} \leq \alpha_{si} z_i T_{cm}$ and $z_0 T_{cm} \leq \sum_{i=1}^{n} \alpha_{si} w_0 T_{cp}$, the following combined inequality can be obtained

$$\rho \leq \frac{w_0 T_{cp}}{z_0 T_{cm}} \leq \frac{z T_{cm}}{n z_0 T_{cp}} \tag{3.68}$$

The above inequality shows that as the number of sensor nodes increases the required upper bound of the ratio $\frac{w_0 T_{cp}}{z_0 T_{cm}}$ reduced. Also, from the condition, $\rho \leq \frac{z T_{cm}}{n z_0 T_{cp}}$, we can see that minimum required value of the ratio $\frac{z T_{cm}}{z_0 T_{cp}}$ is the number of sensor nodes in the cluster, n. This is expected as all the sensing operations of sensor nodes are performed simultaneously and are also the corresponding reporting performed simultaneously, for the effective data aggregation at clusterhead, the processing speed is needed increased. The minimum round time is then given by

$$T_{r,n} = t + \frac{1}{\rho n} z T_{cm} \tag{3.69}$$

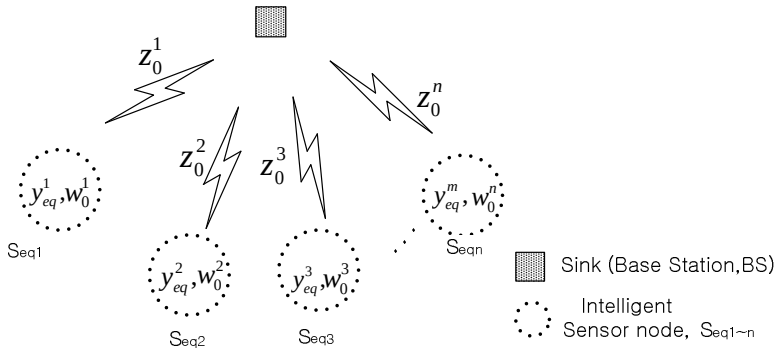


Figure 3.6: The equivalent flat wireless sensor network topology with intelligent sensor nodes

Since the minimum round time on a single sensor node under the fully homogeneous condition is

$$T_{r,1} = t + \frac{1}{\rho} z T_{cm} \tag{3.70}$$

the speedup is then

$$Speedup = \frac{T_{r,1}}{T_{r,n}} = \frac{t + \frac{1}{\rho} z T_{cm}}{t + \frac{1}{\rho n} z T_{cm}} \tag{3.71}$$

Note that if one consider the case that the instruction assignment time, t is negligible (i.e., $t \to 0$), speedup can be achieved as n, which is a linear function to the number of sensor nodes, that is speedup is scalable in this case.

From (3.69) and (3.71), the asymptotic minimum round time and the asymptotic speedup are given respectively as

$$T_{r,\infty} = t, \quad Speedup_{n \to \infty} = 1 + \frac{1}{\rho t} z T_{cm} \tag{3.72}$$

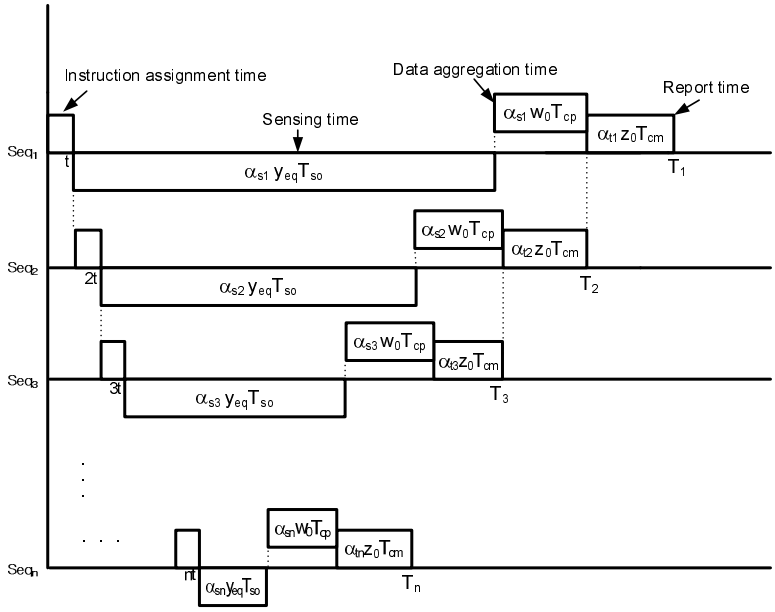


Figure 3.7: Timing diagram for single channel flat wireless sensor network with homogeneous intelligent sensors with no front-end processors.

3.3 Multi-Cluster based hierarchical wireless sensor network scheduling

In this section, we will discuss multi-cluster three tier hierarchical WSN model based on the four scenarios we analyzed in the previous section (see Fig. 3.1(b)). The homogeneous network (i.e.,$z_0^i = z_0$, $w_0^i = w_0$, and $\rho_i = \rho$ for $i = 1, 2, \ldots, n$) is assumed for finding a closed form equation. The methodology used to analyze the multi-cluster model is to collapse a group of nodes (a clusterhead + sensor nodes) composing a cluster into a single equivalent sensor node. This methodology is similar with the one applied in several previous studies [7],[41],[42]. Based on the equivalent node element, we can collapse a cluster into a single "intelligent" sensor node. The terminology, "intelligent" is used for denoting a sensor node which performs its own data aggregation, a significant role of clusterhead. Hence, the final equivalent WSN topology would be flat, not hierarchical as shown in Fig. 3.6. As we mentioned previously, the four scenarios we will discuss in this section is similar to that analyzed in the previous one except for the fact that all sensors equally participate in the sensing operation.

3.3.1 Single Channel with no front-end processor

On the way of collapsing, n clusters will be replaced by n equivalent intelligent sensor nodes. Referring (3.20), the round time on a single equivalent intelligent sensor node can be written as follows

$$t + \frac{1}{\rho} y_{eq} T_{so} + \frac{1}{\rho} z T_{cm} + \frac{1}{\rho} w_0 T_{cp} + z_0 T_{cm} \tag{3.73}$$

By equating the above equation and (3.19), we can obtain y_{eq} as

$$y_{eq} = \frac{1}{T_{so}} \left\{ \frac{1 + \rho C_1'}{1 + C_2'} y T_{so} + \left(\frac{1 + \rho C_1'}{1 + C_2'} - 1 \right) z T_{cm} \right\} \tag{3.74}$$

where, C_1' and C_2' are as described in (3.17).

The timing diagram of the flat WSN with the n homogeneous intelligent sensor nodes is illustrated in Fig. 3.7. From the timing diagram (Fig. 3.7), one can set up the following corresponding recursive load distribution equations

$$\alpha_{si}(y_{eq} T_{so} + w_0 T_{cp}) = t + \alpha_{si+1}(y_{eq} T_{so} + w_0 T_{cp}) + \alpha_{ti+1} z_0 T_{cm} \qquad i = 1, 2, \ldots, n-1 \tag{3.75}$$

The above set of equations can be further expressed using the information utility constant, ρ as

$$\begin{aligned} \alpha_{si}(y_{eq} T_{so} + w_0 T_{cp}) &= t + \alpha_{si+1}(y_{eq} T_{so} + w_0 T_{cp}) + \rho \alpha_{si+1} z_0 T_{cm} \\ &= t + \alpha_{si+1}(y_{eq} T_{so} + w_0 T_{cp} + \rho z_0 T_{cm}) \qquad i = 1, 2, \ldots, n-1 \end{aligned} \tag{3.76}$$

Rewriting the above set of equations as

$$\alpha_{si+1} = f \alpha_{si} - g \qquad i = 1, 2, \ldots, n-1 \tag{3.77}$$

where

$$f = \frac{y_{eq}T_{so} + w_0T_{cp}}{y_{eq}T_{so} + w_0T_{cp} + \rho z_0T_{cm}},\ g = \frac{t}{y_{eq}T_{so} + w_0T_{cp} + \rho z_0T_{cm}} \tag{3.78}$$

From the derivational similarity with subsection 3.2.1, we can solve for α_{s1} as (3.18). Hence, the minimum round time using n homogeneous intelligent sensor nodes, $T_{r,n}$ can be achieved as follows

$$\begin{aligned} T_{r,n} = T_1 &= t + \alpha_{s1}(y_{eq}T_{so} + w_0T_{cp} + \rho z_0T_{cm}) \\ &= t + \frac{\frac{1}{\rho} + C_1'}{1 + C_2'}(y_{eq}T_{so} + w_0T_{cp} + \rho z_0T_{cm}) \end{aligned} \tag{3.79}$$

Since the minimum round time on a single intelligent sensor node is

$$T_{r,1} = t + \frac{1}{\rho}(y_{eq}T_{so} + w_0T_{cp} + \rho z_0T_{cm}) \tag{3.80}$$

the speedup is then

$$Speedup = \frac{T_{r,1}}{T_{r,n}} = \frac{t + \frac{1}{\rho}(y_{eq}T_{so} + w_0T_{cp} + \rho z_0T_{cm})}{t + \frac{\frac{1}{\rho} + C_1'}{1 + C_2'}(y_{eq}T_{so} + w_0T_{cp} + \rho z_0T_{cm})} \tag{3.81}$$

3.3.2 Multi Channel with no front-end processor

On the way of collapsing, n clusters will be replaced by n equivalent intelligent sensor nodes as we showed in the previous subsection. Referring (3.45), the round time on a single equivalent intelligent sensor node can be written as follows

$$t + \frac{1}{\rho}y_{eq}T_{so} + \frac{1}{\rho}zT_{cm} + \frac{1}{\rho}w_0T_{cp} + z_0T_{cm} \tag{3.82}$$

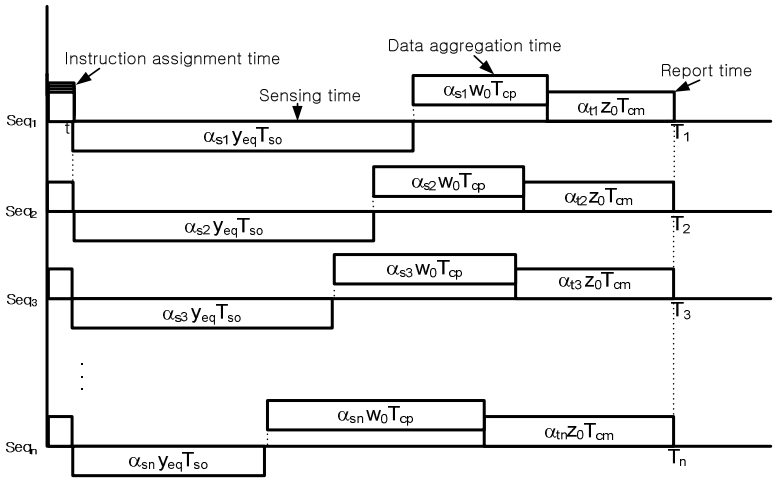


Figure 3.8: Timing diagram for multi channel flat wireless sensor network with homogeneous intelligent sensors with no front-end processor.

By equating the above equation and (3.44), we can obtain y_{eq} as

$$y_{eq} = \frac{1}{T_{so}} \left\{ \frac{1}{n} y T_{so} + \left(\frac{1}{n} - 1 \right) z T_{cm} \right\} \tag{3.83}$$

The timing diagram of the flat WSN with the n homogeneous intelligent sensor nodes is illustrated in Fig. 3.8. From the timing diagram (Fig. 3.8), one can set up the following corresponding recursive load distribution equations

$$t + \alpha_{si}(y_{eq} T_{so} + w_0 T_{cp}) + \alpha_{ti} z_0 T_{cm} = t + \alpha_{si+1}(y_{eq} T_{so} + w_0 T_{cp}) + \alpha_{ti+1} z_0 T_{cm}$$
$$i = 1, 2, \ldots, n-1 \tag{3.84}$$

The above set of equations can be further expressed using the information utility constant, ρ as

$$t + \alpha_{si}(y_{eq} T_{so} + w_0 T_{cp} + \rho z_0 T_{cm}) = t + \alpha_{si+1}(y_{eq} T_{so} + w_0 T_{cp} + \rho z_0 T_{cm})$$
$$i = 1, 2, \ldots, n-1 \tag{3.85}$$

Rewriting the above set of equations simply as

$$\alpha_{si+1} = \alpha_{si} \qquad i = 1, 2, \ldots, n-1 \tag{3.86}$$

From the intuitive similarity with $III - B$, we can solve for α_{s1} as (3.43).
Hence, the minimum round time using n homogeneous intelligent sensor nodes, $T_{r,n}$ can be achieved as follows

$$\begin{aligned} T_{r,n} = T_1 &= t + \alpha_{s1}(y_{eq}T_{so} + w_0T_{cp} + \rho z_0T_{cm}) \\ &= t + \frac{1}{\rho n}(y_{eq}T_{so} + w_0T_{cp} + \rho z_0T_{cm}) \end{aligned} \tag{3.87}$$

Since the minimum round time on a single intelligent sensor node is

$$T_{r,1} = t + \frac{1}{\rho}(y_{eq}T_{so} + w_0T_{cp} + \rho z_0T_{cm}) \tag{3.88}$$

the speedup is then

$$Speedup = \frac{T_{r,1}}{T_{r,n}} = \frac{t + \frac{1}{\rho}(y_{eq}T_{so} + w_0T_{cp} + \rho z_0T_{cm})}{t + \frac{1}{\rho n}(y_{eq}T_{so} + w_0T_{cp} + \rho z_0T_{cm})} \tag{3.89}$$

3.3.3 Single Channel with front-end processor

By following a similar step of collapsing showed in the previous subsections, referring (3.59), the round time on a single equivalent intelligent sensor node can be written as follows

$$t + \frac{1}{\rho}y_{eq}T_{so} + z_0T_{cm} \tag{3.90}$$

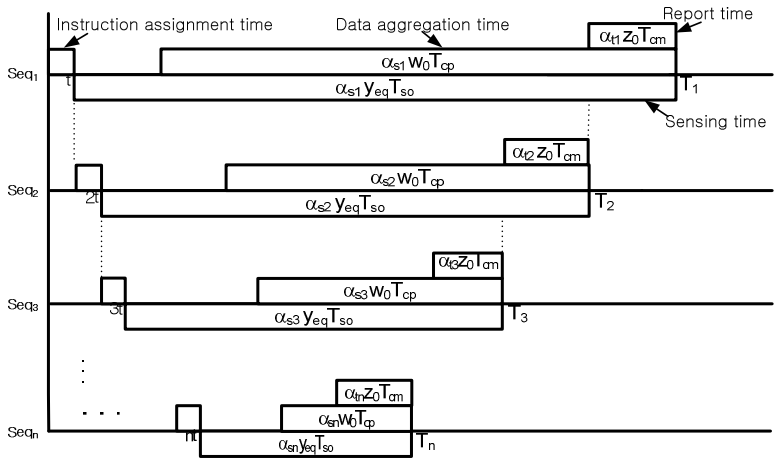


Figure 3.9: Timing diagram for single channel flat wireless sensor network with homogeneous intelligent sensors with front-end processor.

By equating the above equation and (3.56), we can obtain y_{eq} as

$$y_{eq} = \frac{1}{T_{so}} \left(\frac{1 + \rho C_1'}{1 + C_2'} \right) y T_{so} \tag{3.91}$$

The timing diagram of the flat WSN with the n homogeneous intelligent sensor nodes is illustrated in Fig. 3.9. Here, we assume that $z_0 T_{cm} < w_0 Tcp$ so that reporting of sensory data at each sensor node can end with own data aggregation at the same instant even though the i^{th} reporting starts when the $(i+1)^{th}$ equivalent intelligent sensor nodes terminates its own reporting (Fig. 3.9). From the timing diagram (Fig. 3.9), one can set up the following corresponding recursive load distribution equations

$$\alpha_{si} y_{eq} T_{so} = t + \alpha_{si+1} y_{eq} T_{so} + \alpha_{ti} z_0 T_{cm} \qquad i = 1, 2, \ldots, n-1 \tag{3.92}$$

The above set of equations can be further expressed using the information utility constant, ρ as

$$\alpha_{si} (y_{eq} T_{so} - \rho z_0 T_{cm}) = t + \alpha_{si+1} y_{eq} T_{so} \qquad i = 1, 2, \ldots, n-1 \tag{3.93}$$

Rewriting the above set of equations as

$$\alpha_{si+1} = f\alpha_{si} - g \qquad i = 1, 2, \ldots, n-1 \tag{3.94}$$

where

$$f = \frac{y_{eq}T_{so} - \rho z_0 T_{cm}}{y_{eq}T_{so}}, \quad g = \frac{t}{y_{eq}T_{so}} \tag{3.95}$$

Here, $y_{eq}T_{so} > \rho z_0 T_{cm}$.

From the derivational similarity with $III - A$, we can solve for α_{s1} as (3.18). Hence, the minimum round time using n homogeneous intelligent sensor nodes, $T_{r,n}$ can be achieved as follows

$$T_{r,n} = T_1 = t + \alpha_{s1} y_{eq} T_{so} = t + \frac{\frac{1}{\rho} + C_1'}{1 + C_2'} y_{eq} T_{so} \tag{3.96}$$

Since the minimum round time on a single intelligent sensor node is

$$T_{r,1} = t + \frac{1}{\rho} y_{eq} T_{so} \tag{3.97}$$

the speedup is then

$$Speedup = \frac{T_{r,1}}{T_{r,n}} = \frac{t + \frac{1}{\rho} y_{eq} T_{so}}{t + \frac{\frac{1}{\rho} + C_1'}{1 + C_2'} y_{eq} T_{so}} \tag{3.98}$$

3.3.4 Multi Channel with front-end processor

The timing diagram of the flat WSN with the n homogeneous intelligent sensor nodes is illustrated in Fig. 3.10. We assume here that $z_0 T_{cm} > w_0 T_{so}$, so that the speed data

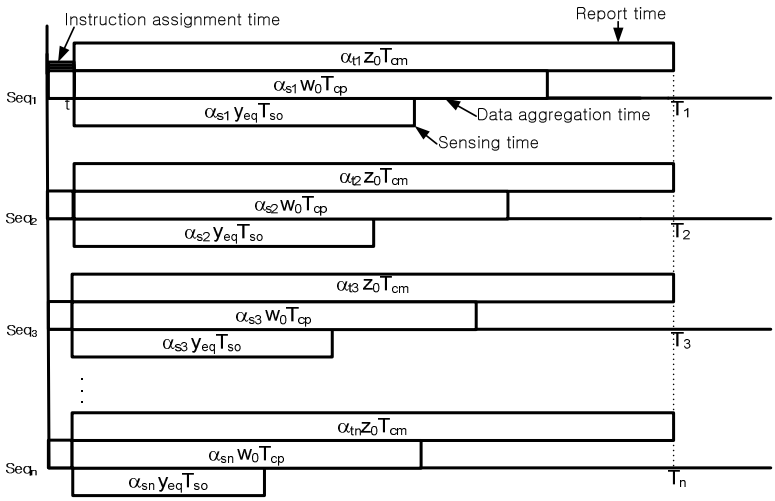


Figure 3.10: Timing diagram for multi channel flat wireless sensor network with homogeneous intelligent sensors with front-end processor.

aggregation is faster than the speed of reporting. In other words, by intuitive sense, the data aggregation is needed to be ceased before or exactly when the reporting terminates. From the timing diagram (Fig. 3.10), one can set up the following corresponding recursive load distribution equations without considering the equivalent sensing speed, y_{eq}.

$$\alpha_{ti} z_0 T_{cm} = \alpha_{ti+1} z_0 T_{cm} \qquad i = 1, 2, \ldots, n-1 \tag{3.99}$$

From the intuitive similarity with subsection 3.2.4, we can solve for α_{t1} as $\frac{1}{n}$.
Hence, the minimum round time using n homogeneous intelligent sensor nodes, $T_{r,n}$ can be achieved as follows

$$T_{r,n} = T_1 = t + \frac{1}{n} z_0 T_{cm} \tag{3.100}$$

Since the minimum round time on a single intelligent sensor node is

$$T_{r,1} = t + z_0 T_{cm} \tag{3.101}$$

Index	2	3	4	5	6	7	8	9	10
η_i	0.25024	0.12512	0.06256	0.03128	0.01564	0.00782	0.00391	0.00195	0.00098
γ_i	1.50240	0.25122	−0.37439	−0.68719	−0.84360	−0.92180	−0.96090	−0.98045	−0.99022
η_k / γ_k	·	·	0.16710	0.04552	0.01854	0.00848	0.00407	0.00199	0.00099
$(1-\eta_l)/\gamma_l$	0.49902	3.48250	·	·	·	·	·	·	·
$\alpha_{si(opt)}$	0.25160	0.12535	0.06222	0.03066	0.01488	0.00699	0.00305	0.00107	0.00009
$\alpha_{si(\text{inf eas})}$	0.25175	0.12537	0.06219	0.03059	0.01480	0.00690	0.00295	0.00098	−0.00001

Table 3.1: Example of the condition for the feasible measurement instruction assignment time.

the speedup is then

$$Speedup = \frac{t + z_0 T_{cm}}{t + \frac{1}{n} z_0 T_{cm}} \tag{3.102}$$

3.4 Performance evaluation

3.4.1 Feasible measurement instruction assignment time

Now we demonstrate the usage of the condition for the feasible measurement instruction assignment time constant by means of an illustrative example. We consider the scenario of SCnP with parameters, $n = 10$, $\rho = 1.0$, $T_{so} = 1.0$, $T_{cm} = 1.0$, and $T_{cp} = 1.0$. The speed parameters are set as $y = 1.0$, $z = 1.0$, $z_0 = 0.1$, and $w_0 = 0.1$. Based on the (3.24), η_i and γ_i for $i = 2, 3, \ldots, 10$ are given as Table 3.1. From the polarity of γ_i, $k = 4, 5, \ldots, 10$ and $l = 2, 3$. According to the condition (see (3.27)), the minimum value over the fourth and fifth rows of the Table 3.1 is given as 0.00099. Thus, the condition for the feasible measurement instruction assignment time constant is $0 \leq t < 0.00099$. For the check of feasibility of t, the sixth row of the Table 3.1 shows the optimal values of α_{si} for $i = 2, 3, \ldots, 10$ when $t = 0.0009$ sec, which is the feasible time value in the boundary. On the other hand, α_{s10} is given as a negative value which is obviously not a reasonable value for the α_s when $t = 0.001$ sec, which is an infeasible time value for the upper bound, 0.00099.

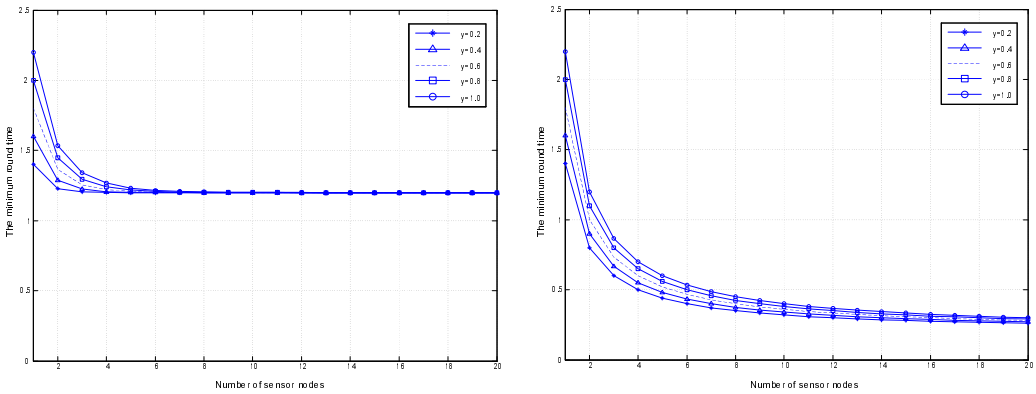
t=0, Tso=1.0, Tcm=1.0, Tcp=1.0					
	ρ	z0	w0	z	y
vs. y	1.0 (SCnP,MCnP, SCP,MCP)	0.1 (SCnP,MCnP, SCP,MCP)	0.1 (SCnP,MCnP,MCP)	1.0 (SCnP,MCnP)	Variable
			0.2 (SCP)	0.1 (SCP)	
				2.0 (MCP)	
vs. z	1.0 (SCnP,MCnP, SCP,MCP)	0.1 (SCnP,MCnP, SCP,MCP)	0.1 (SCnP,MCnP, SCP,MCP)	Variable	1.0 (SCnP,MCnP)
					1.2 (SCP)
					0.1 (MCP)
vs. ρ	Variable	0.1 (SCnP,MCnP, SCP,MCP)	0.1 (SCnP,MCnP, SCP,MCP)	1.0 (SCnP,MCnP,SCP)	1.0 (SCnP,MCnP,MCP)
				2.0 (MCP)	1.2 (SCP)

Table 3.2: Simulation speed parameters.

3.4.2 Minimum round time

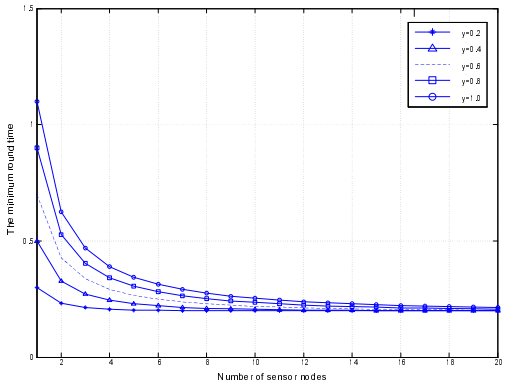
With common parameters $t = 0$, $T_{so} = 1.0$, $T_{cm} = 1.0$, and $T_{cp} = 1.0$, the minimum total round time of the four scenarios of SCnP, MCnP, SCP, and MCP are plotted against the number of sensor nodes in the fully homogeneous cluster for different sensing speeds, y, for different communication speeds, z, and for different information utility constants, ρ, using the speed parameters shown in the Table 3.2. Speed parameters for the scenarios of SCP and MCP are set according to the assumptions perviously mentioned for the minimum round time, $zT_{cm} < yT_{so}$, (3.55) and $zT_{cm} > yT_{so}$, (3.68) respectively.

In Fig. 11, the five performance curves are obtained with $y = 0.2, 0.4, 0.6, 0.8$, and 1.0, respectively for the 4 scenarios. As shown in the subfigures, the longer the sensing delay, the longer the total round time, and the round time for the five cases saturates and converges to the value, $T_{r,\infty}$ which is independent of the sensing speed parameter, y as the number of sensor nodes increases. But the reduction is not too significant after just a few sensor nodes. Several similar intuitive curves have been shown in [37],[38]. Specifically, for the MCP scenario, the round time curves for 5 different values of sensing speed are exactly identical since round time is given as a function independent to the sensing speed parameter as shown (3.69). Similarly, in Fig. 12, the minimum total round time of the each scenario with $z = 0.2, 0.4, 0.6, 0.8$, and 1.0, is plotted respectively except MCP scenario with $z = 2.2, 2.4, 2.6, 2.8$, and 3.0. The subfigures show better total round time is obtained as the communication speed increases. The subfigures show better total round time is obtained as the communication speed increases. For all of the scenarios, the saturation of the total

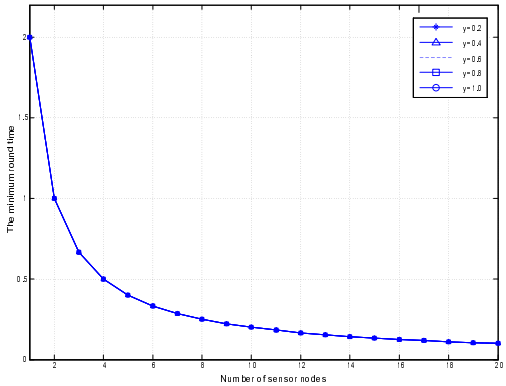


(a) SCnP

(b) MCnP

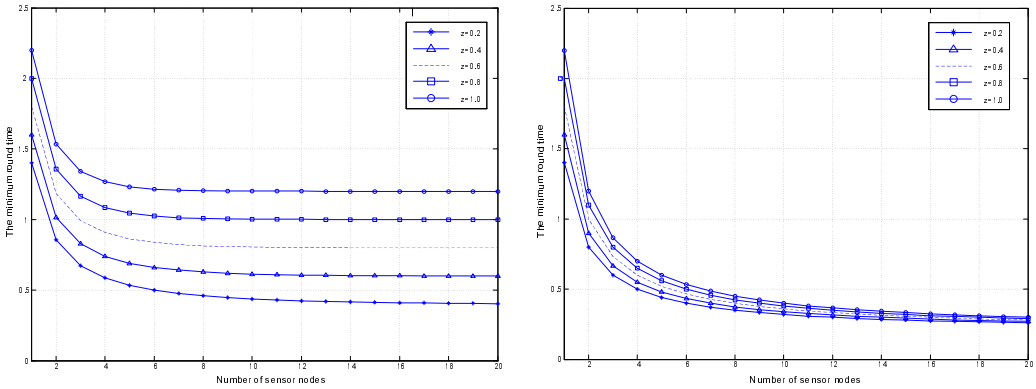


(c) SCP



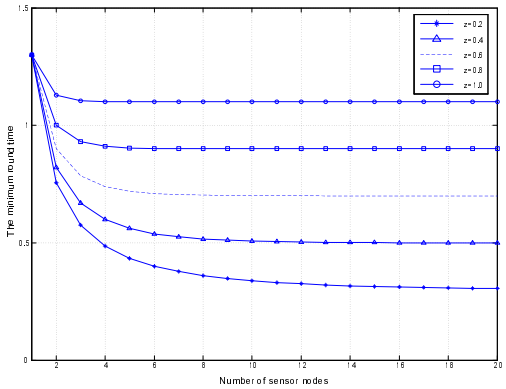
(d) MCP

Figure 3.11: Total round time versus the number of senor nodes for the fully homogeneous cluster with variable y.

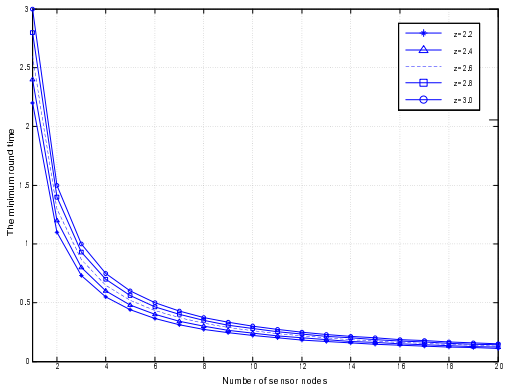


(a) SCnP

(b) MCnP



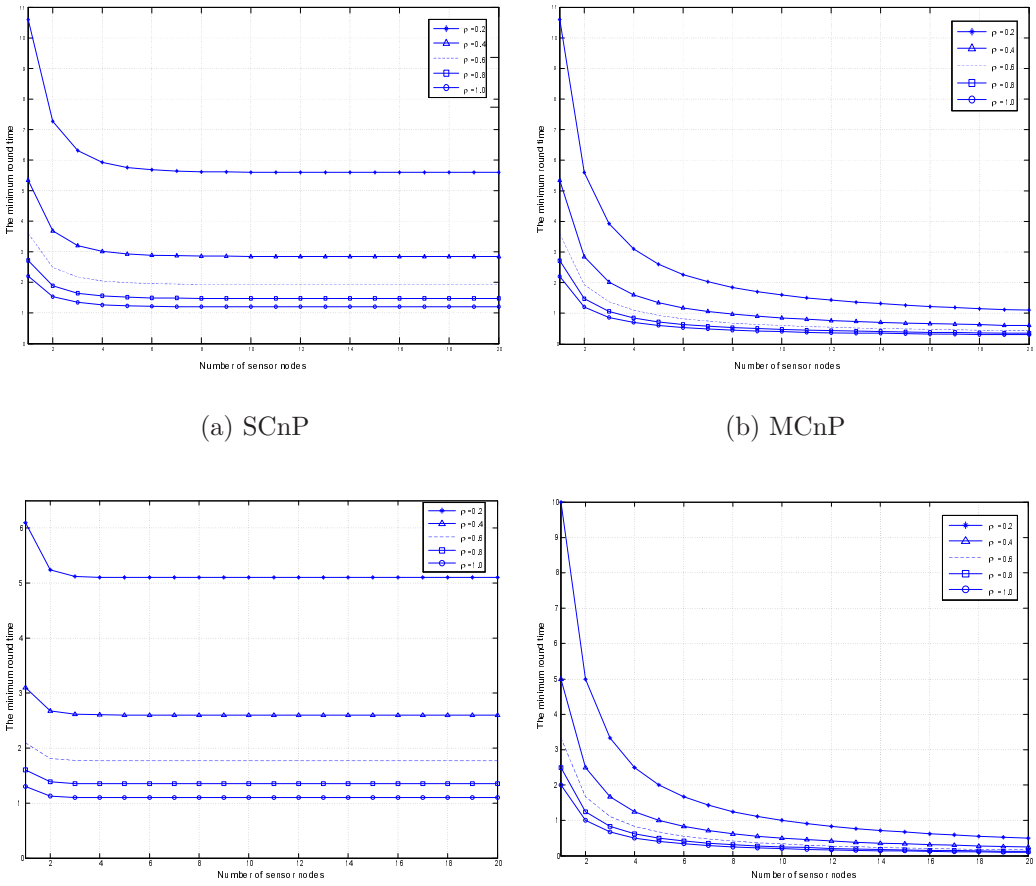
(c) SCP



(d) MCP

Figure 3.12: Total round time versus the number of sensor nodes for the fully homogeneous cluster with variable z.

round time with respect to different values of z is shown as the number of sensor nodes increases. The convergence of the total round time for five different communication speeds is shown in Fig. 3.12(b) and Fig. 3.12(d) since $T_{r,\infty}$ is independent to the communication speed parameter, z ((3.47) and (3.72)). In Fig. 13, the five performance curves are obtained with $\rho = 0.2, 0.4, 0.6, 0.8$, and 1.0, respectively for each scenario. In all subgraphs in Fig. 13, we can also see that the saturation of the total round time for five different information utility constants, ρ, as the number of sensor nodes increases. As we expect intuitively when the information utility gets higher (i.e., $\rho \to 1$), the total round time reduces. This is expected as the higher information utility constant decreases not only the sensing and



(a) SCnP (b) MCnP

(c) SCP (d) MCP

Figure 3.13: Total round time versus the number of sensor nodes for the fully homogeneous cluster with variable ρ.

reporting time at sensor nodes but also the data aggregation and reporting time at the clusterhead.

3.4.3 Speedup

In Fig. 14 and 15, the speedup of the the 4 scenarios (SCnP, MCnP, SCP,and MCP) are described against the number of sensor nodes and the information utility constant in the fully homogeneous cluster for different sensing speeds, y and communication speeds, z, respectively. The same speed parameters used in previous subsection are applied. A better speedup characteristic can be seen as a result of the multiple channel comparing Fig. 3.14(a)

to (b) (Fig. 3.15(a) to (b)) and Fig. 3.14(c) to (d) (Fig. 3.15(c) to (d)). It also can be seen that the front-end processors contribute to the better speedup comparing Fig. 3.14(a) to (c) (Fig. 3.15(a) to (c)) and Fig. 3.14(b) to (d) (Fig. 3.15(b) to (d)). All of the plots except for the MCP scenario show the speedup saturation as the number of sensor nodes increases. Especially, Fig. 3.14(d) and Fig. 3.15(d) show speedup given as a linearly increasing curve of first order n. As we previously mentioned, in the case that the instruction assignment time, t is negligible, speedup is achieved as a scalable function by n, which is independent of the speed parameter y and z (see (3.71)). As a reminder, the value of speedup saturation of the 4 scenarios are given as (3.29), (3.48), (3.62), and (3.72). Interestingly, a smaller increment in the speedup according to the variation of the information utility is shown relative to the case of the variation of the number of sensor nodes. The smaller sensitivity of the information utility constant in the speedup can be mathematically analyzed in that the information utility constant, ρ appearing in both numerator and denominator seems to largely cancel out. A modestly better speedup characteristic as information utility decreases implies that relatively more performance enhancement in round time can be achieved by additional sensor nodes when the accuracy of collected data is low.

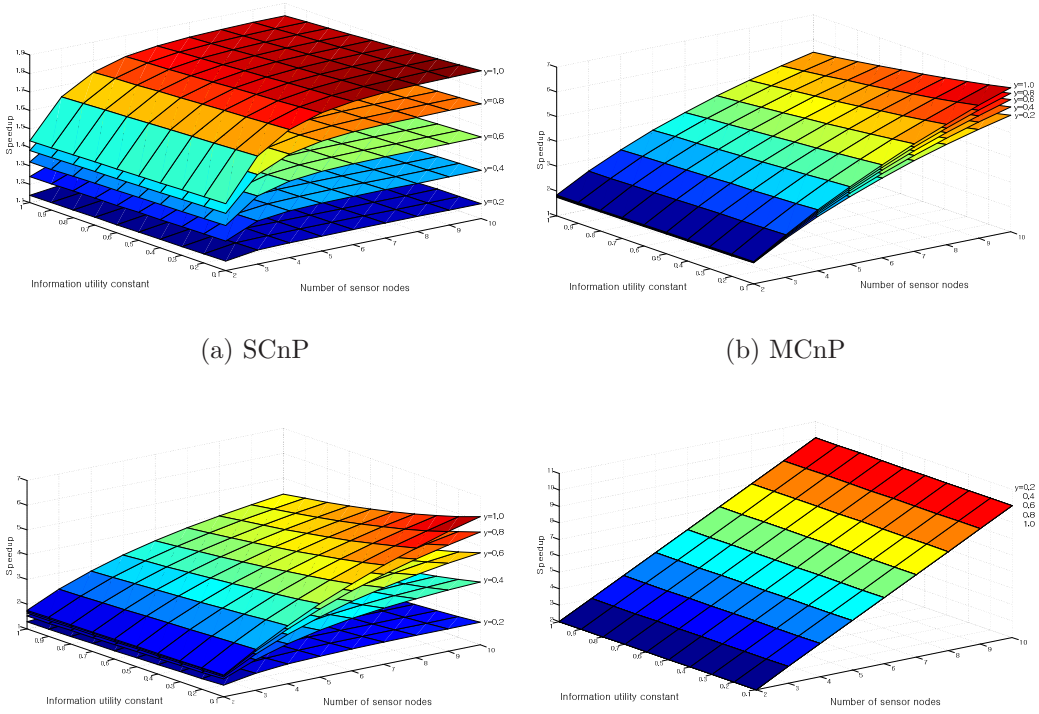
3.4.4 Energy Dissipation

Power usage in wireless sensor nodes has been studied for finite amounts of non renewable energy in sensor networks. A radio model has been developed to model the energy dissipated by a sensor node when transmitting and receiving data [43]. To transmit a k bit data a distance d, the energy dissipated is

$$E_{tx}(k,d) = E_{elec} \cdot k + \varepsilon_{amp} \cdot k \cdot d^2 \tag{3.103}$$

and to receive the k bit data, the radio expends

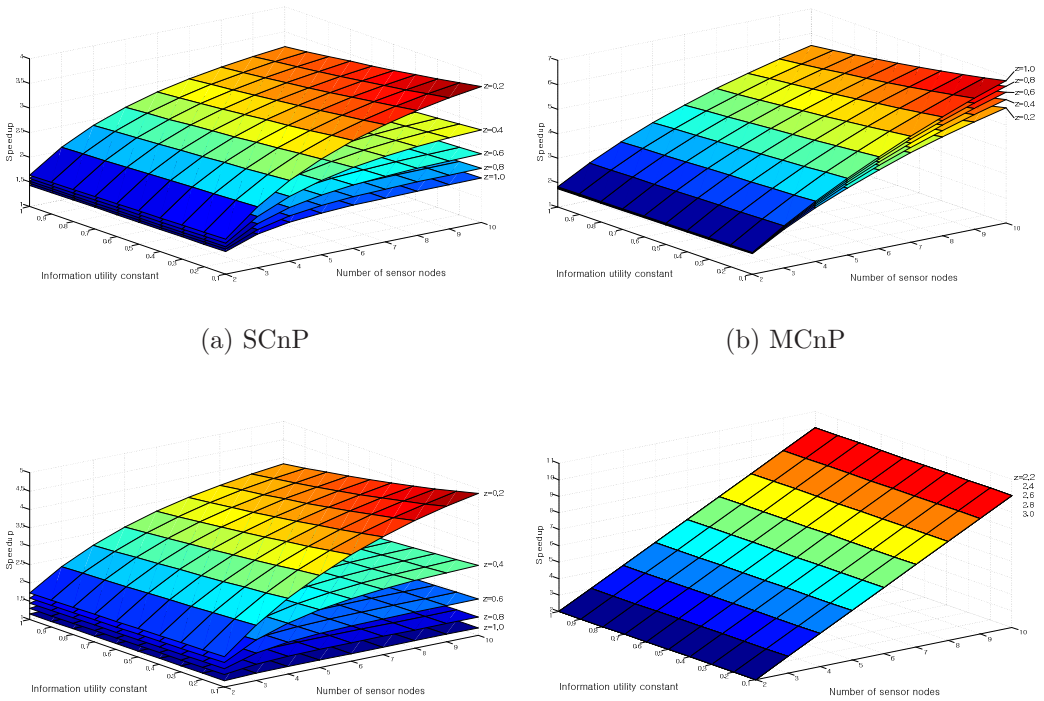
$$E_{rx}(k) = E_{elec} \cdot k \tag{3.104}$$



(a) SCnP (b) MCnP

(c) SCP (d) MCP

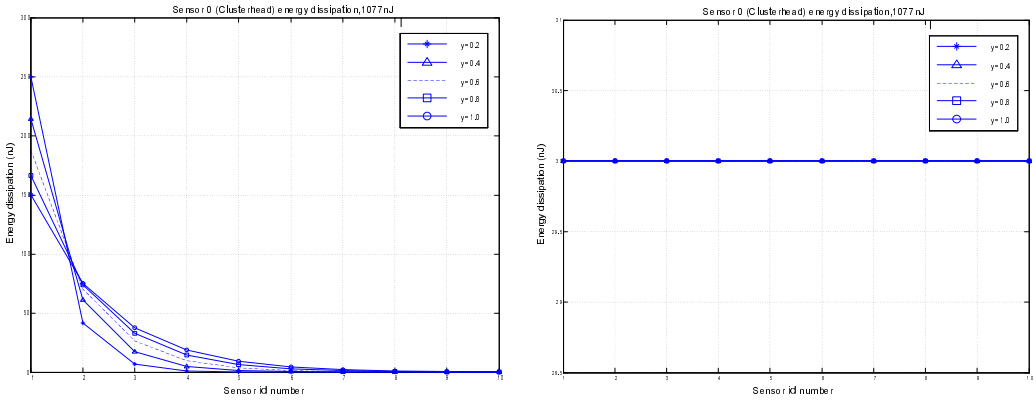
Figure 3.14: Speedup for the fully homogeneous cluster with variable ρ and y.



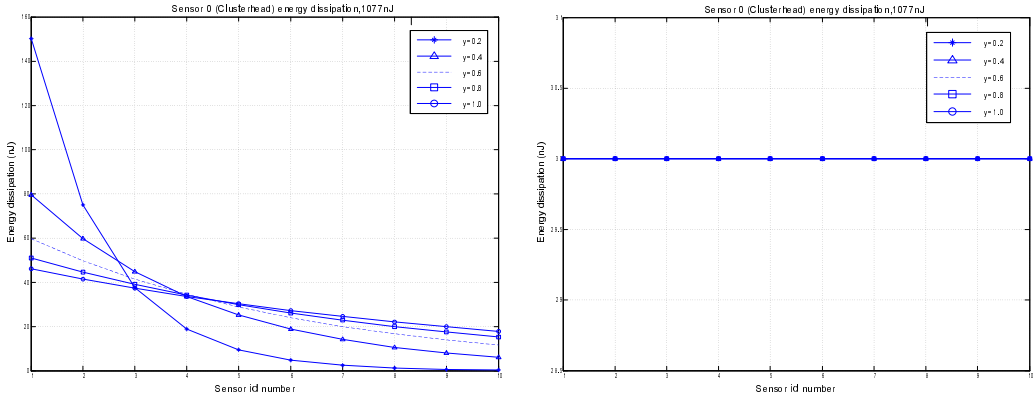
(a) SCnP (b) MCnP

(c) SCP (d) MCP

Figure 3.15: Speedup for the fully homogeneous cluster with variable ρ and z.

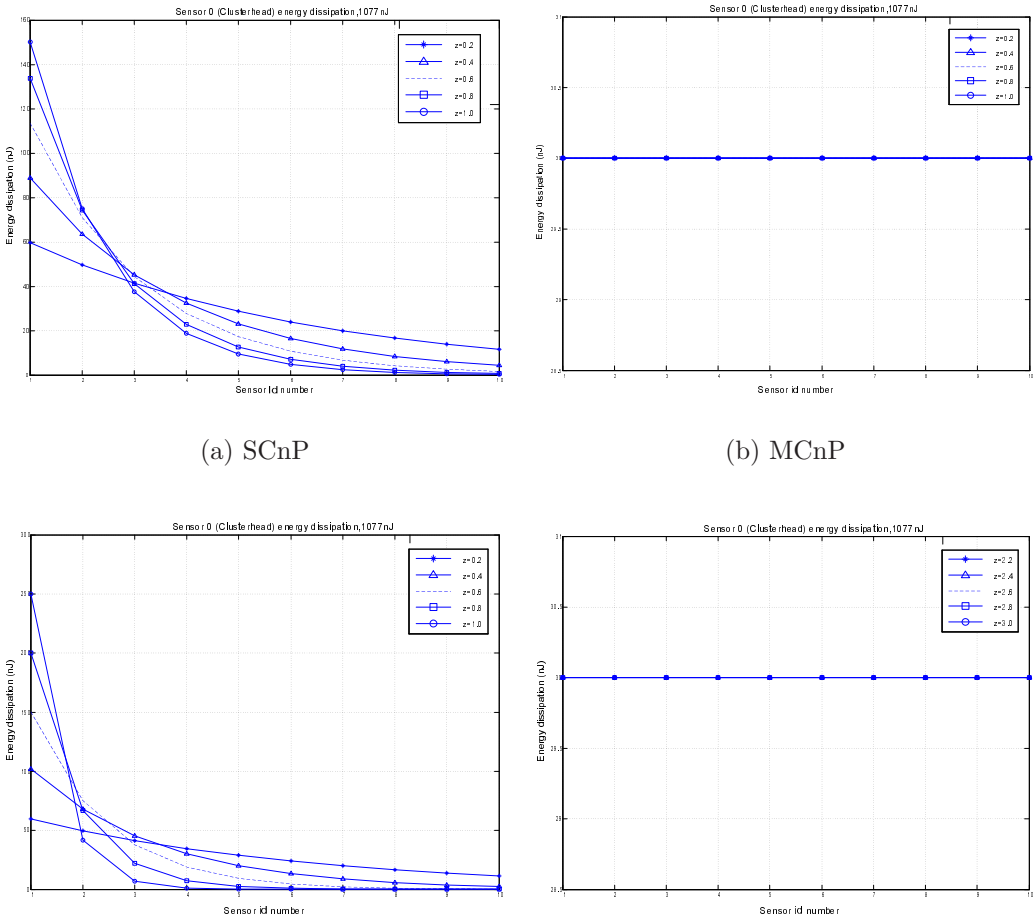


(a) SCnP (b) MCnP



(c) SCP (d) MCP

Figure 3.16: Energy dissipation versus sensor id number for the fully homogeneous cluster with variable y.

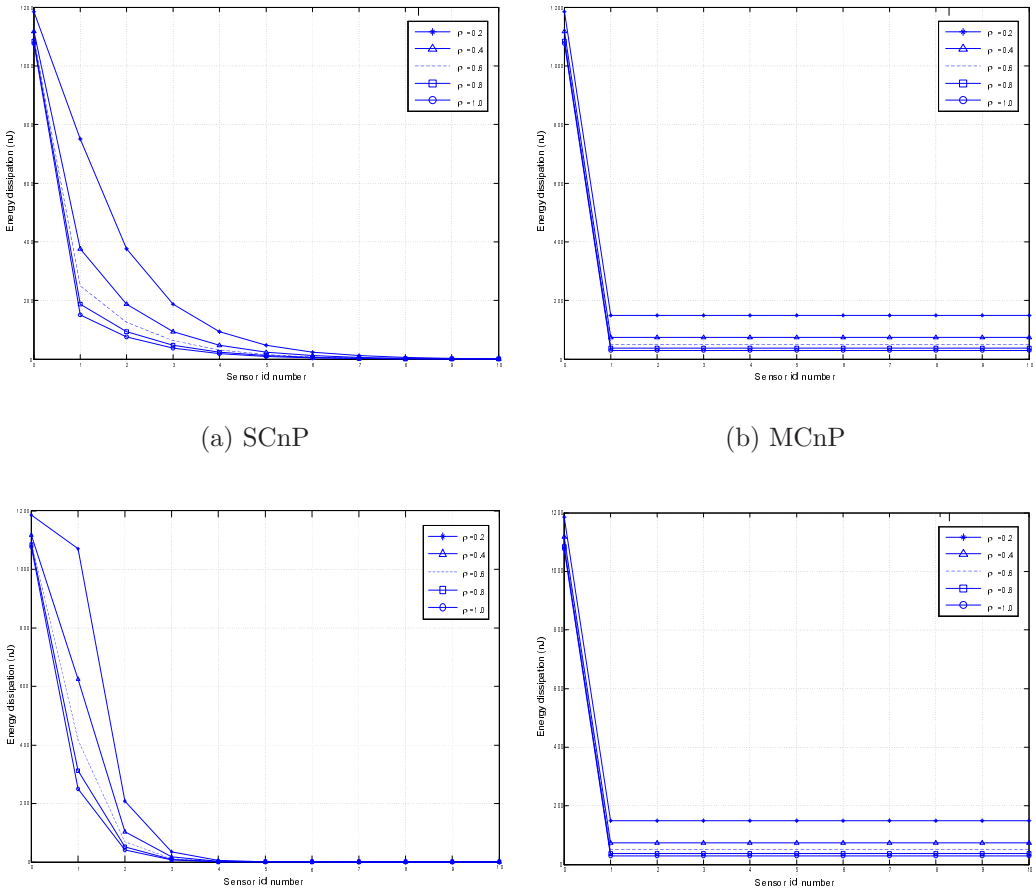


(a) SCnP (b) MCnP

(c) SCP (d) MCP

Figure 3.17: Energy dissipation versus sensor id number for the fully homogeneous cluster with variable z.

Here, the parameter, k can be substituted with the normalized fraction of data, αs, computed by DLT. The DLT based analysis of the wireless sensor energy dissipation has been studied using the first order radio model [37]. For this work, we assume that the radio model also follows the first order radio model. Also, we consider an energy dissipation other than the radio, mainly in the processors in the sensor nodes. It is well known that the radio energy dissipation overwhelms the other losses such as processing energy dissipation. However, energy dissipation in processing and data aggregation at the clusterhead seems important to be considered in the meaning of DLT since clusterhead deals with relatively large amount of collected data from each sensor node.



(a) SCnP (b) MCnP

(c) SCP (d) MCP

Figure 3.18: Energy dissipation versus sensor id number for the fully homogeneous cluster with variable ρ.

The following notation and the corresponding value are used for our simulation.

d : distance from each sensor node to clusterhead $50m$. (distance from clusterhead to sink, $100m$).

E_{elec} : 50 nJ/b.

ε_{amp} : 100 $pJ/b/m^2$.

For the analysis of the processing energy dissipation, we use the experimental parameters used in [44]. From [44] a Mica2 sensor mote is specified as a 38.4$Kbps$ radio that operates at $3V$ (2xAA Batteries), or $27nJ/b$ ($27W \cdot s/b$) processing cost.

Based on the first order radio model and the computed processing energy dissipation by using Mica2 specifications, the total energy dissipation of the 4 scenarios (SCnP, MCnP, SCP, and MCP) are plotted in Fig. 16 against 10 sensors including the clusterhead in the homogeneous cluster for different sensing speeds, y, with the same values of the parameters used in the previous simulation for the total round time for different sensing speeds. As shown in Fig. 4.5(a) and 4.5(c), the energy dissipation at each sensor node unevenly decreases as the sensor node number increase. It is because the energy dissipation is mainly related to the amount of collected and reported data. As the sensing speed is faster, the first few sensor nodes would have more data to be processed so that more energy dissipation is highly concentrated on the first few sensor nodes as shown in Fig. 4.5(a) and 4.5(c). The energy dissipation at a clusterhead is computed as an identical amount, $1077nJ$ for the four scenarios. This is expected as the clusterhead processes all the reported data which depends on only the information utility constant and reports the perfectly aggregated data (unit amount) to a sink. Specially, for the MCnP and MCP scenario, the energy dissipation curves for 5 different values of sensing speed are exactly identical to a constant value since the total data is equally distributed to each sensor node due to the simultaneous sensing and reporting enabled by muti-channels under the fully homogeneous cluster case as shown (3.43).

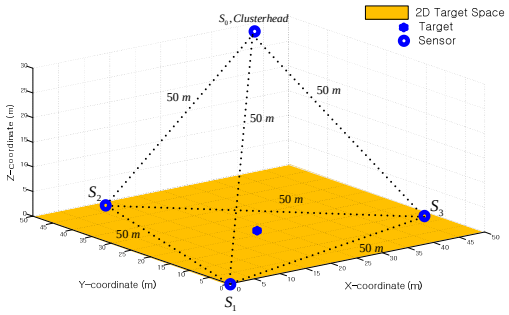
Similarly, in Fig. 17, the total energy dissipation of the 4 scenarios is plotted against the number of sensor nodes in the homogeneous cluster for different communication speeds, z,

with the values of the parameters used in the previous simulation for the total round time for different z. The plots show intuitively similar results with the result in the case of different sensing speed.

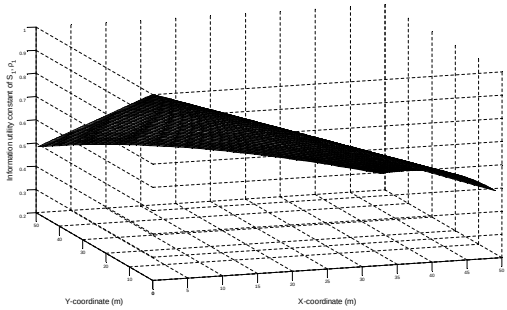
In Fig. 18, the total energy dissipation of the 4 scenarios are plotted against the number of sensor nodes in the homogeneous cluster for different information utility constant, ρ with the values of the parameters used in the previous simulation for the total round time for different ρ. Here, sensor number 0 denotes the clusterhead. Intuitively, the smaller the information utility constant (that is more redundant data), the more energy dissipation as shown in Fig. 18. As we mentioned before, the clusterhead processes all the reported data which depends on the information utility constant so that the energy dissipation at the clusterhead varies according to the information utility constant.

3.4.5 3D Cluster Model

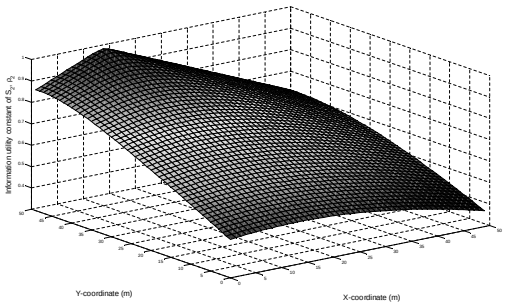
Simulation is carried out to illustrate the relationship between different information utility constants according to the variation of a target location model involving the minimum round time. We consider a three dimensional tetrahedron cluster with $50m$ edges, three sensor nodes (S_1,S_2, and S_3) positioned on the vertices of a base triangle and a clusterhead (S_0) positioned on the other vertice as illustrated in Fig. 3.19(a). The three sensor nodes could be ground based sensing stations and the clusterhead is airborne. The target location can be varied on the 2D square target space as shown in Fig. 3.19(a). In this simulation, we use the estimation technique introduce in [39] so that each sensor node has a value of information utility constant, 0.7, especially when the target lies at the center of the base triangle. That is the heterogeneous information utility constants are generated as the target location varies. Here, the multi channel with no front end processor (MCnP) scheduling scenario is applied for the simulation with the same values of the speed parameters used in the previous simulation for the total round time for different ρ. The information utility constant is illustrated when the target is moving in the square 2D target space as shown in Fig. 3.19(b), 3.19(c), and 3.19(d). As we expect, the figures show a convex peak at the



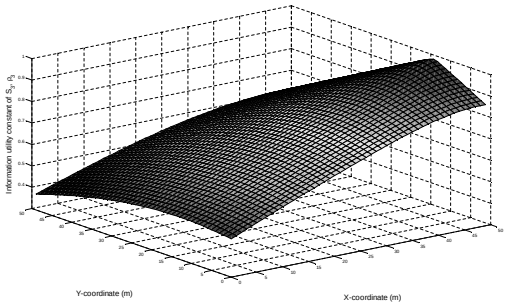
(a) 3D Cluster model with varying target location.



(b) Information utility constant, ρ_1 vs Target location

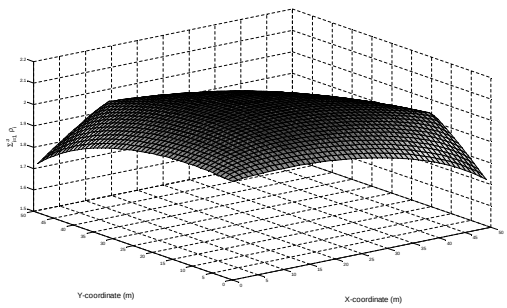


(c) Information utility constant, ρ_2 vs Target location

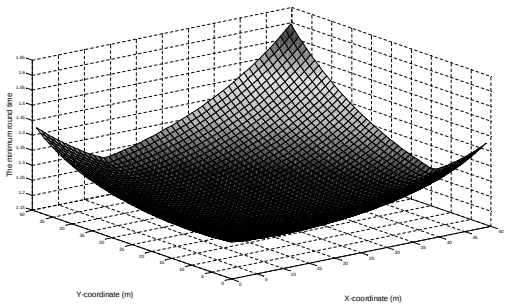


(d) Information utility constant, ρ_3 vs Target location

location of the each sensor. In other words, the corresponding information utility constant increases as the target moves toward each of the sensor nodes as expected. As we derived in (3.42), the minimum round time for MCnP scheduling scenario inversely depends on the value of the summation of the information utility constants of each sensor node. Fig. 3.19(e) shows the convex distribution of the value of $\sum_{i=1}^{3} \rho_i$ over the region of the target space. As we expect, a concave result for the minimum round time is shown in Fig. 3.19(f) as a vertical flipped version of Fig. 3.19(e). Fig. 20 shows the energy dissipation of each sensor (S_0, S_1, S_2, and S_3) over the region of the target space. From Fig. 20 we can recognize all plots are depicted as a similar shape by each other with Fig. 3.19(f). This is because the amount of reported data from each sensor node and total amount of the collected data at clusterhead also inversely depends on the sum of the information utility constants of each sensor node (see (3.41)). Similarly, in MCnP scenario, we can also expect identical result



(e) Total information utility constant vs Target location



(f) Minimum round time vs Target location.

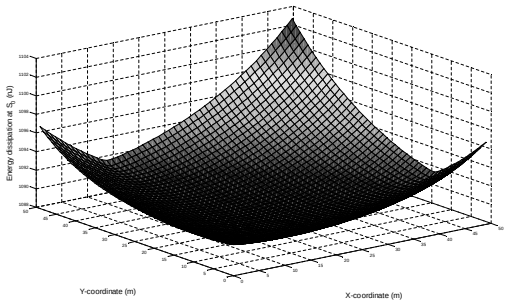
Figure 3.19: MCnP scheduling for 3D Cluster model

for the energy dissipation at the sensor nodes, S_1, S_2, and S_3 as shown in Fig. 19.

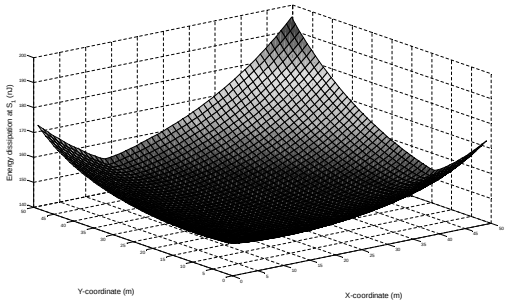
3.5 Concluding remarks

In this chapter, closed form solutions for the minimum round time for several scenarios of single cluster wireless sensor networks are derived. The performance of these scenarios are examined according to different sensing speeds, communication speeds, and information utility constants. The condition for feasible measurement instruction assignment time is derived and a numerical example is presented to describe the importance of DLT feasibility by demonstrating the operation of the condition. The special bounds for the ratio of speed parameters for the maintenance of the minimum round time are also derived. By using an equivalent speed parameter, it is shown that a multi-cluster based WSN can be analyzed as a flat wireless sensor network without clusters. The performance of clustered WSN networks is shown by using a deterministic analysis method, divisible load theory. By direct deterministic approaches, our work gives a general idea of the performance of WSN.

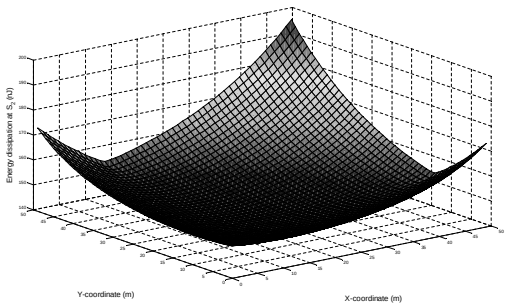
For extensions to our work, the analysis of a multi-cluster WSN topology would be interesting. A more comprehensive study concerning the relationship between the speed parameters and the information utility constant and the corresponding performance including speedup and asymptotic performance is worth addressing for future work. As for the more rigorous



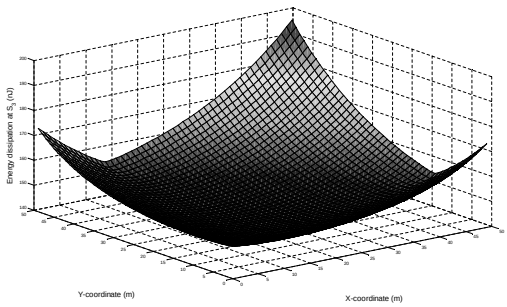
(a) Energy dissipation at S_0(clusterhead) vs Target location.



(b) Energy dissipation at S_1 vs Target location.



(c) Energy dissipation at S_2 vs Target location.



(d) Energy dissipation at S_3 vs Target location.

Figure 3.20: Energy dissipation versus Target location MCnP scheduling.

analytic results, the study of the following issues are also expected to extend our study:

• The analytical model for a WSN with direct communication between sensor nodes (Ad hoc WSN).

• The analysis of performance variation according to the quality of data aggregation.

• The effect of the heterogeneous speed parameters including the information utility.

• The analysis of special bounds for the related parameters (i.e., speed parameters, instruction assignment time, and information utility) under a heterogenous WSN.

Chapter 4

Resource Scheduling Heuristics for Data Intensive Networks

The unprecedented volume of data generated by modern physics experiments such as those taking place at the Relativistic Heavy Ion Collider (RHIC) at Brookhaven National Laboratory (BNL) and the Large Hadron Collider (LHC) at CERN, is demanding new strategies for data collection, sharing, management, and transfer over networks [45]. For example, the STAR experiment [46] is already collecting data at the rate of over one terabyte per day. The volume of data will increase by a factor of ten in the next five years. As another example, the BNL ATLAS Computing Facility needs to provide a Grid-based storage system capable of sustaining data rates of one gigabyte per second for incoming and outgoing data between BNL and ATLAS Tier 0, Tier 1 and Tier 2 sites, to support thousands of reconstruction and/or analysis jobs [47].

Experimental and analysis data need to transferred over the network within specific time frames to be shared among various institutions for processing and interpretation. Transferring such vast amounts of data and meeting arrival deadlines raises a significant scheduling problem. Networks are shared mediums with default behavior to treat all data flows equally, lacking the capability to dynamically reserve resources along the full path between sources and destinations. Such resource reservations, however, have been made possible through a

number of recent networking projects [48],[49],[50],[51],[52].

The TeraPaths project at BNL [53],[54],[55] is a DOE-funded project aimed at guaranteeing network resources to specific data flows. The TeraPaths software establishes end-to-end virtual network paths with guaranteed resources by acting both as an end site LAN controller and as a client to WAN-supported resource reservation service. TeraPaths utilizes DiffServ-based Quality of Service (QoS) and Policy-Based Routing (PBR) techniques to prioritize, protect, and regulate individual data flows. Furthermore, TeraPaths reserves and manages dedicated WAN paths indirectly, through OSCARS services. OSCARS [51] is a joint ESnet [56] and Internet2 [57] project. The OSCARS software establishes dedicated network paths within the ESnet backbone (layer 2 and layer 3) and the Internet2 backbone (layer 2 only). Layer 2 paths (dynamic circuits) can be established across both backbones, enabling this form of connectivity between universities and DOE laboratories. TeraPaths-controlled end-sites establish circuit connectivity between their border routers. Authorized flows are prioritized and regulated through DiffServ and directed into these circuits with PBR. The end result is that selected flows follow a resource-guaranteed path extending between their source and destination nodes.

The idea behind the work presented in this chapter is to utilize the new capabilities provided by TeraPaths not only to schedule the utilization of network resources for the benefit of data transfers, but also to minimize resource waste and increase the numbers of data transfers that can be accommodated simultaneously.

The establishment of virtual paths between source and destination node pairs simplifies the network topology into a set of virtual end-to-end connections between those nodes. In this chapter, we examine policies for assigning file transfers onto a set of source and destination node pairs interconnected by this simplified virtual topology. We consider two opposite policies. The first one minimizes the number of active source-destination node pairs by aggregating flows onto the same node pairs until the network connection capacity limit is reached. The second policy spreads the load (load balancing) amongst the node pairs. Many policy variations are possible, however, these two policies are at the two extremes of the

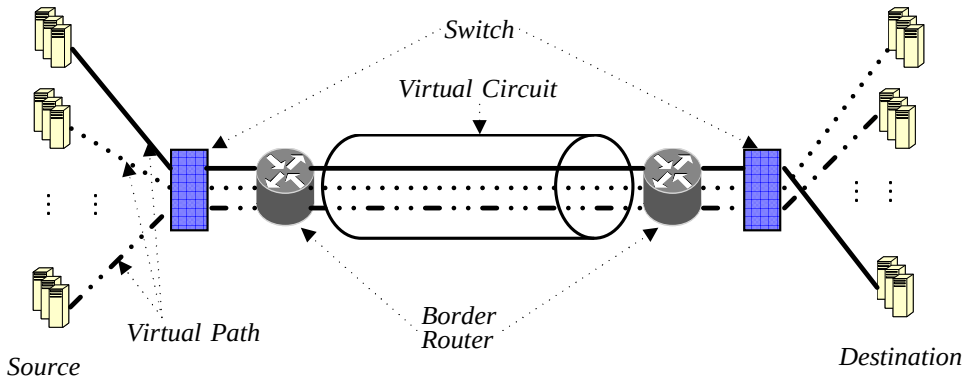


Figure 4.1: Virtual connections between multiple source nodes and destination nodes

problem space, and serve as a starting point for studying file transfer assignment problems when network resource scheduling is supported. As a data transfer model, we use a simple topology with four nodes at both source and destination sites so that the trend of capacity reservations of each node at both sites can be investigated tractably for both policies.

In the following section, we present the notation and analytic background used in this chapter. Section 4.2 describes the idea of time varying capacity analysis with an example. In Section 4.3 we introduce heuristic scheduling modules and present two resource reservation heuristics we developed, while we evaluate the performance of these heuristics in section 4.4. Finally, we present our conclusions in section 4.5.

4.1 Problem Formulation and Preliminary Remarks

Consider the case of a heterogeneous network model consisting of N source nodes (source cluster) and M destination nodes (destination cluster). Each cluster belongs to an end site. Virtual circuits (VCs) are established through the WAN between the border routers of the two end sites (layer 2 or 3). A virtual source to destination connection with specific, guaranteed capacity - called a virtual path (VP) - is established for each individual file transfer (see Fig. 4.1). Each VC can accommodate one or more VPs depending on its own capacity. There are three identifiable phases in a VP's lifetime:

1. VP setup: During the setup phase, the source node contacts (lookups) the database in

the switch which has the information of files and the current capacity state and specifies the destination node(s), and waits for the network to set up the VP.

2. File transfer: Once the VP has been established, files can begin to be forwarded along the VP.

3. VP teardown: This is initiated when the deadline of the file transfer times out. Either destination node may close the connection (VP).

The following assumptions are initially made:

1. Nodes can be considered to be hosts, routers, Ethernet switches, or any other device where the input and output links can have different characteristics.

2. File transfers are generally independent of each other except in terms of interacting through capacity constraints.

3. The file transmission is uni-directional. (e.q., a file can only be transferred from a source node to destination node(s)).

4. A node can establish and terminate multiple VPs concurrently and do this for the multiple nodes at the destination sites.

5. The connection requests arrivals are a stationary Poisson process. The assumption is used for this initial study. Other assumptions could be the subject of future work.

6. Each of the nodes has a capability to start a file transfer as soon as it receives the file transfer request.

7. Each of the datasets is composed of several files that can be distributed to multiple nodes, but each file is indivisible.

8. Compared to the size of the data, the time to setup a VP is negligible.

9. Once a file transfer request (FTR) is ready, all nodes at both source site and the destination site share the file profile information. The file profile is composed of three elements, $F($ file size, file transfer start time, file transfer deadline).

The following notation is used in this chapter:

N, (M) : Total number of nodes at the source, (destination) site in the network.

$VP_{(i,j)}$: Virtual Path (VP) that is established between i^{th} source node and j^{th} destination

node (where $i \in 1, 2, \ldots, N$, and $j \in 1, 2, \ldots, M$).

C_{Si}, (C_{Dj}) : Capacity upper bound of the i^{th}, (j^{th}) node at the source, (destination) site (NIC capacity).

$C_{VP(i,j)}$: Capacity upper bound of the established $VP_{(i,j)}$.

C_{VC} : Capacity upper bound of the VC over the WAN.

$C_{Si}(t)$, $(C_{Dj}(t))$: Reserved capacity temporal variation at the i^{th}, (j^{th}) node on the source, (destination) site.

$C_{(i,j)}(t)$: Reserved capacity temporal variation on $VP_{(i,j)}$.

f, (f_k) : Size of the file, (k^{th} file) involved in the file transfer.

T_S, (T_D) : The file transfer start (i.e., request arrival) time, (file transfer deadline).

t_s, (t_d) : The modified file transfer start (i.e., request arrival) time, (file transfer deadline), where $T_S \leq t_s < T_D$ and $T_S < t_d \leq T_D$.

T_S^k, (T_D^k) : The k^{th} file transfer start (i.e., request arrival) time, (the k^{th} file transfer deadline).

C_F, (C_F^k) : Least required capacity (i.e., data rate) of the VP to transfer the file, (k^{th} file) with size f, (f_k), where $C_F = \frac{f}{T_D - T_S}$, $(C_F^k = \frac{f_k}{T_D^k - T_S^k})$. Here, f, (f_k) is the maximum file size that can be transmitted from the source and correctly received by the destination over a VP during the interval $[T_S,\ T_D]$, $([T_S^k,\ T_D^k])$.

4.2 Time varying Capacity analysis

The network may have some *background* file transfers through established VPs. The background file transfers represent both the on-going file transfers utilizing the capacity of the VP and the reserved (i.e., capacity guaranteed) file transfer requests (FTRs) that have not yet begun transmission. Here, we call a VP node (pair) which has a background file transfer a "busy" node (pair). The background file transfers will have a varying effect on capacity. In other words, the capacity can be considered as a time-varying parameter, $C(t)$. An example diagram for the reserved capacity temporal variation of a VP, $C_{(i,j)}(t)$ is depicted in Fig. 4.2.

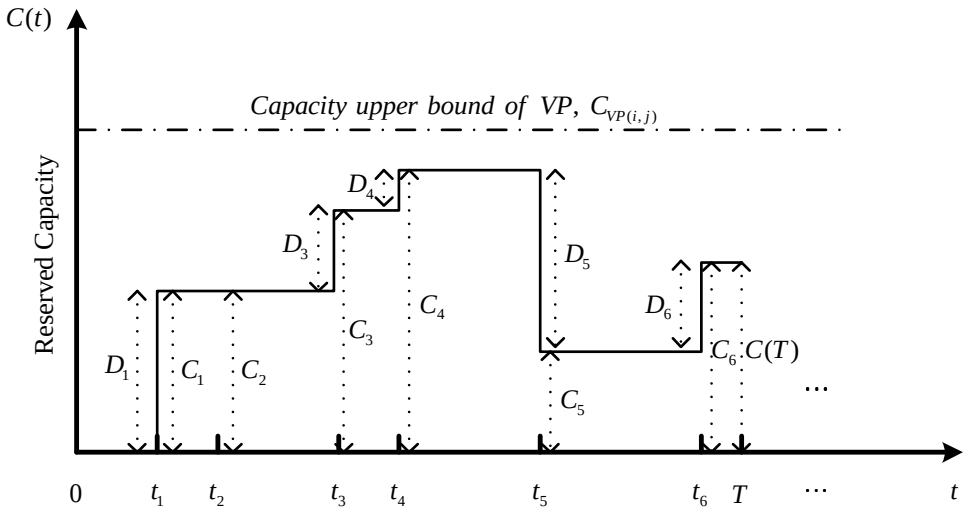


Figure 4.2: Example of reserved capacity temporal variation, $C(t)$ on $VP_{(i,j)}$. Here, T is current time

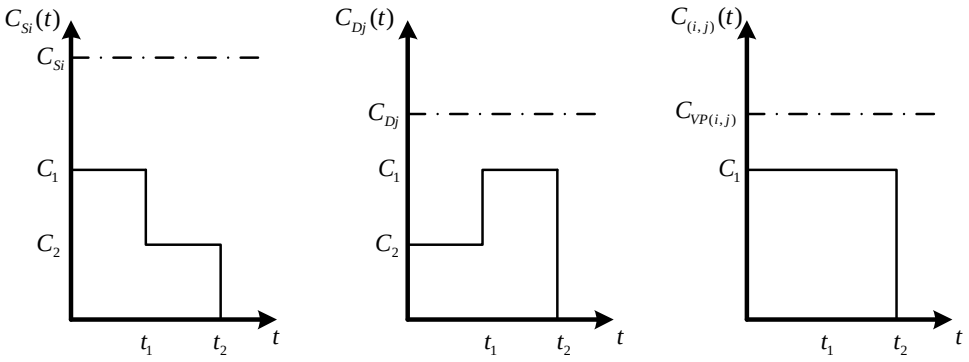


Figure 4.3: Example of capacity temporal variation of a VP connecting the i^{th} source node and the j^{th} destination node. Here, C_1 and C_2 are two different levels of reserved capacity.

The temporal variation of the aggregate reserved capacity of VPs established between the i^{th} source node and the j^{th} destination node can be investigated using the information of the reserved capacity temporal variation at both the source node and the destination node, $C_{Si}(t)$ and $C_{Dj}(t)$. Here, $C_{(i,j)}(t)$ is limited by the maximum reserved capacity at either source node or destination node value at each time instant.

$$C_{(i,j)}(t) = max[C_{Si}(t), C_{Dj}(t)] \tag{4.1}$$

With similar reasoning, the capacity upper bound of the established $VP_{(i,j)}$ between the i^{th} source node and the j^{th} destination node is limited by the minimum value of the capacity

upper bound among the capacity upper bound of i^{th} source node and j^{th} destination node, C_{Si} and C_{Dj} as a bottleneck.

$$C_{VP(i,j)} = min[C_{Si}, C_{Dj}] \tag{4.2}$$

For clarification, an example of capacity temporal variation of the i^{th} source node, the j^{th} destination node, and the VP is illustrated in that order in Fig. 4.3.

From the timing diagram, Fig. 4.2, the corresponding equation for reserved capacity temporal variation in a VP, $C(t)$ can be formulated as

$$C(t) = \sum_{i} C_i \Big\{ U(t - t_i) - U(t - t_{i+1}) \Big\} \tag{4.3}$$

where index i denotes every instance of T_S^k and T_D^k. Here, $U(t)$ is the unit step function. Constant C_i is the sum of the allocated capacities of each existing flow at time t_i. Constant C_i can be defined as

$$C_i = \sum_{x} C_F^x \tag{4.4}$$

where the integer value, x is the index k satisfying $T_S^k \leq t_i < T_D^k$. Now, we define a parameter, ρ $(0 \leq \rho \leq 1)$, capacity utilization, along a time slice $[T - W,\ T]$ (i.e., a time slice with size W) as

$$\rho = \frac{\overline{C(t)}}{C_{VP(i,j)}} \tag{4.5}$$

where

$$\overline{C(t)} = \frac{\int_{T-W}^{T} C(t)dt}{W}, \tag{4.6}$$

which is the time average of the reserved capacity of the VP in the time slice $[T - W,\ T]$. Generally, the utility constant, ρ implies the capacity utilization of a VP within a time slice with a certain size that one is interested in. We can use ρ as an indicator of the node information utility which describes the utilization trend of the capacity at a certain node. Here, W is set as the farthest deadline of the reserved FTRs (file transfer requests) or the background file transfer flow. Thus, the utilization constant, ρ, represents the fraction of the capacity that is being used and is a value between zero (meaning nothing is used) and one (meaning the VP is fully saturated). Multiplying the utilization by 100 yields the percent utilization of the VP.

To find the utilization along W, it is necessary to find, $\int_{T-W}^{T} C(t)dt$. The part of the integration can be modified as an appropriate arithmetic form for a computer simulation involving discrete time.

$$\int_{T-W}^{T} C(t)dt = T \cdot C(T) - (T - W) \cdot C(T - W) - \sum_{i=n+1}^{m} t_i \Big(C(T_i) - C(T_{i-1}) \Big) \tag{4.7}$$

See Appendix A for details. Finally, from eq (4.7) one can obtain the VP utilization, ρ with deterministic analysis. This is feasible due to the fact that VC network maintains state information for its ongoing file transfer connection. The connection-state information including $\overline{C(t)}$, T_S^k, and T_D^k is being stored by tracking and updating the switch's translation table, containing each VP's connection information.

4.3 Capacity scheduling heuristic

The heuristic algorithm is basically composed of three modules as follows

1. VP Selection module: The FTRs (file transfer requests) are mapped to a busy node pair which is eligible to accommodate the FTR in a least instantaneous capacity demand manner to meet the deadline. In case that there exists no busy node pair, the initial VP node pair is selected randomly. The VP selection module chooses an efficient VP, which can

accommodate the least capacity demand, C_F based on the following conditions

$$C_F \leq C_{VP(i,j)} - max(C_{(i,j)}(t)) \tag{4.8}$$

Here, the value of $max(C_{(i,j)}(t))$ is the maximum reserved capacity over the time slice $[T_S, T_D]$ on the temporal capacity domain of $VP_{(i,j)}$. In a similar manner, the candidacy of the VP is also required to be taken into account under the consideration of the capacity upper bound of the VC.

$$C_F \leq C_{VC} - max(C_{(i,j)}(t)) \tag{4.9}$$

To search for the maximum reserved capacity and reduce the algorithmic computational complexity, the searching scheme is performed only at the capacity stepping times (for instance, t_k from Fig. 4.2), not at all unit time granularity. It can be expected that the algorithm running time is significantly reduced by using this searching scheme. Note that the maximum possible number of VPs is $M \cdot N$, that is the maximum number of VPs taken into account in the module.

2. Time-slice Search module: If it is found that there is no node pair candidate having enough capacity that can accommodate C_F at the VP Selection module, feasible modified time slices, $[t_s, t_d]$, which can accommodate a modified (increased) capacity demand, $C'_F (= \frac{f}{t_d - t_s})$ within the time slice, $[T_S, T_D]$, are investigated in this module. The range of the the modified time slices, $[t_s, t_d]$, is formed within the original time slice, $[T_S, T_D]$, so that the modified capacity demand is larger than the original capacity demand over the modified time slice. A similar searching scheme utilized in the VP selection module is also applied for searching for the modified time slice candidates. Here, if there is no feasible modified time slice, the initial file transfer request is rejected in the end. Here, we call the heuristic that does not consider the time slice modification as "*best effort, BE*". In other words, the BE heuristic rejects the FTR if there is no eligible VP candidate.

3. Time-slice Selection module: Once modified time slice candidates are obtained, an efficient modified time slice is chosen by the module. According to the criteria for choosing an efficient modified time slice, two heuristic algorithms are introduced in the following sections.

It is worth mentioning here that the processes in all three modules are performed based on the VP capacity temporal domain, $C_{(i,j)}(t)$, not on the source capacity temporal domain nor on the destination capacity temporal domain.

4.3.1 Most Conservative (MC) heuristic algorithm

The objective of the most conservative (MC) heuristic algorithm is to provide capacity-guaranteed file transfers which maximizes the utilization of busy VP node pairs as much as possible. Since the MC heuristic algorithm runs in the context of utilizing the busiest node pairs (i.e., node pair having max ρ), each file transfer is likely to highly utilize a relatively small number of node pairs. We can intuitively expect that the MC heuristic provides a better reservation performance under high capacity demand circumstances. It should be emphasized here that we know information about the temporal capacity state of the network based on the analysis shown in the previous section.

The MC heuristic algorithm basically follows the three modules we briefly introduced previously. Once multiple node pair candidates satisfy the condition, eq (4.8) and eq (4.9), the algorithm gives the busiest node pair (i.e., having max ρ) a preference. Therefore, the reserved capacity tends to be concentrated in "busy" time periods where capacity utilization, ρ, is maximized. This makes available more conserved capacity during "non-busy" time periods. The details of the modules consisting of the MC heuristic are described in Fig. 4.4. In Fig. 4.4(b), C_{PEAK} denotes the value of the maximum reserved capacity along the time slice $[t_s, t_d]$. An insufficient capacity region (i.e, time region demanding capacity beyond the upper bound of VP (or VC)) truncation in advance of the grid-like searching of feasible time slices contributes to reducing unnecessary time consumption for the modified time-slice search by reducing the possible modification time region. In Fig. 4.4(c), $min\ C'_F$

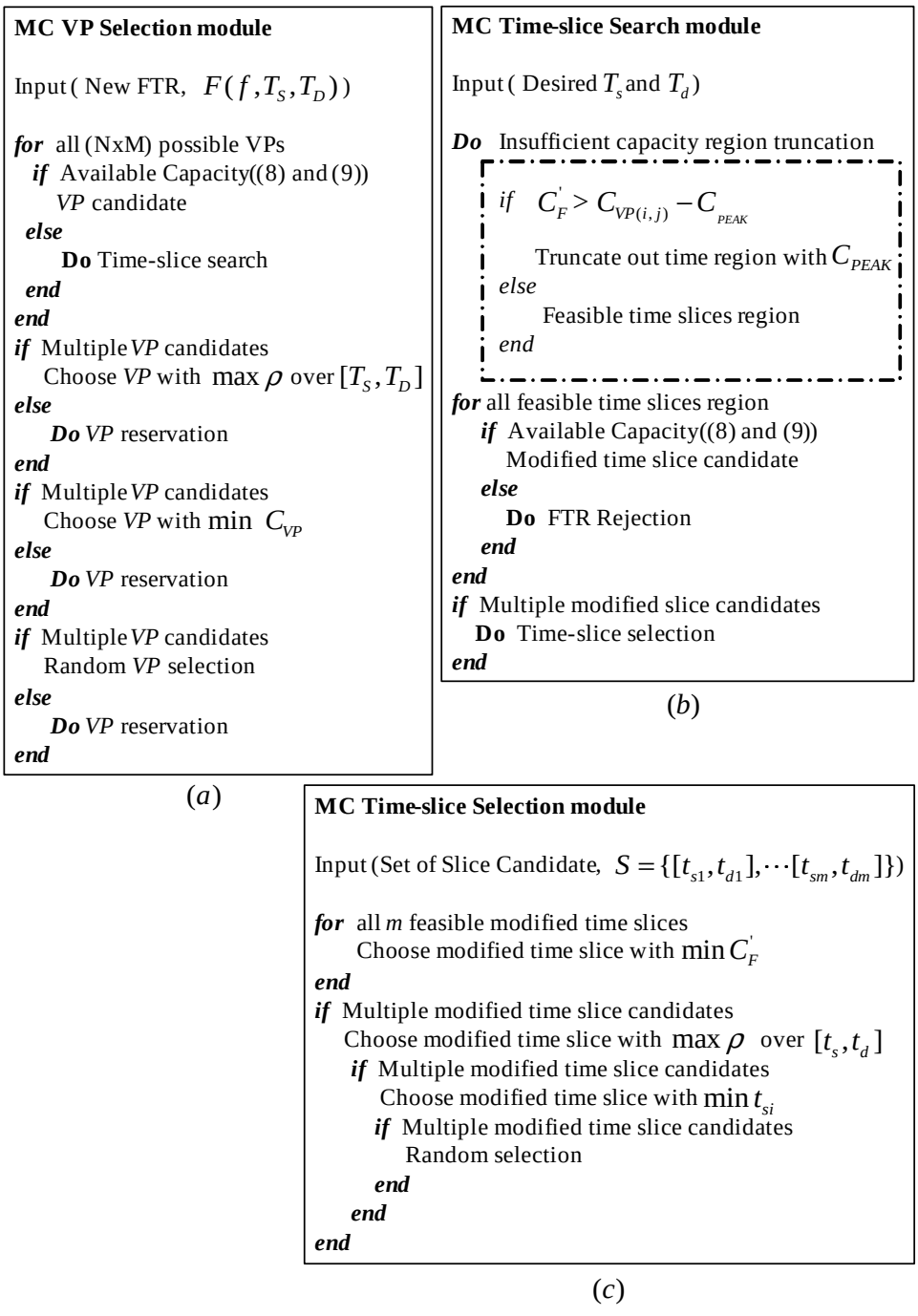


Figure 4.4: MC heuristic modules. (a) VP selection module, (b) Time-slice search module, (c) Time-slice selection module.

100 FTRs, 100 Iterations		
M, N = 4, C_{VP} = 10 GB/m, C_{VC} = 40 GB/m		
File size, f	**Start time, Ts**	**Transfer time, TD-Ts**
Ave file size 2 ~ 202 GB	~ U(1,20) min Ave Ts ≈ 10 min	~ U(1,120) min Ave TD-Ts ≈ 60 min
~ P(20,2) Ave f ≈ 40 GB	~ U(1,20) min Ave Ts ≈ 10 min	Ave transfer time 1 ~ 145 min
~ P(20,2) Ave f ≈ 40 GB	Ave start time 1 ~ 49 min	~ U (1,120) min Ave TD-Ts ≈ 60 min

Table 4.1: Simulation parameters for the comparison of the heuristics

denotes for the least required capacity to transfer a file within a modified time slice.

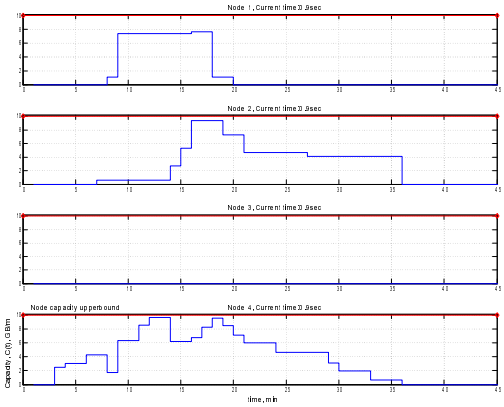
4.3.2 Load Balancing (LB) heuristic algorithm

In contrast to the MC heuristic, the load balancing (LB) heuristic algorithm provides capacity-guaranteed file transfer while minimizing the utilization of busy node pairs as much as possible. In other words, the only difference with respect to the algorithm for the MC heuristic is the substitution *min* ρ for *max* ρ. The LB scenario is advantageous under lower capacity demand conditions. This is because each file transfer is likely to utilize various VP node pairs satisfying min ρ. The LB heuristic contributes to allowing balanced capacity reservation by using multiple VP node pairs as compared with the MC heuristic (see Fig. 4.5).

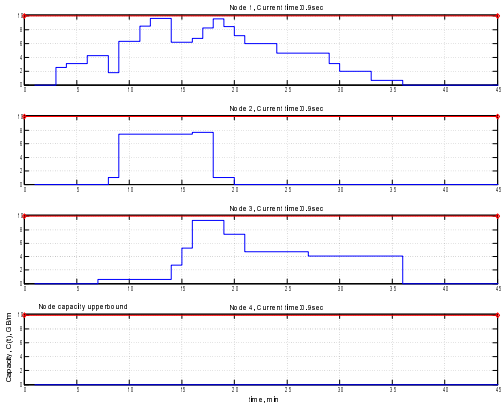
4.4 Performance comparison of the heuristics

4.4.1 Nodal capacity comparison

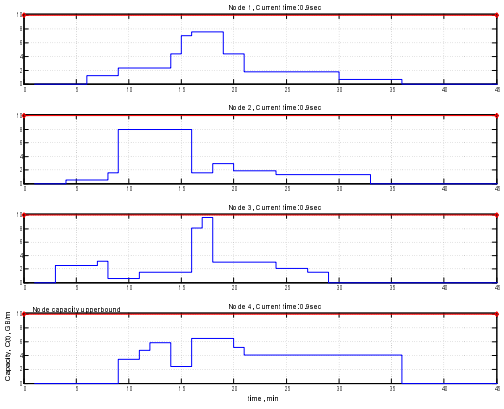
In Fig. 4.5, the capacity temporal reservation state over 4 nodes on the source and destination site respectively with C_{Si} and C_{Dj}=10 GB/m, and C_{VC}=40 GB/m is plotted. The file size is generated by one of the heavy tailed distributions [58], the Pareto distribution, $P(k, \alpha)$ with shape parameter, k=10 and scale parameter, α=2. The file transfer start time, T_S and the file transfer time, $T_D - T_S$ is generated by a uniform distribution on the interval $[1, 20]$, $U(1, 20)$ (i.e., T_D= T_S+$U(1, 20)$). For the simulation, 20 sample FTRs are gener-



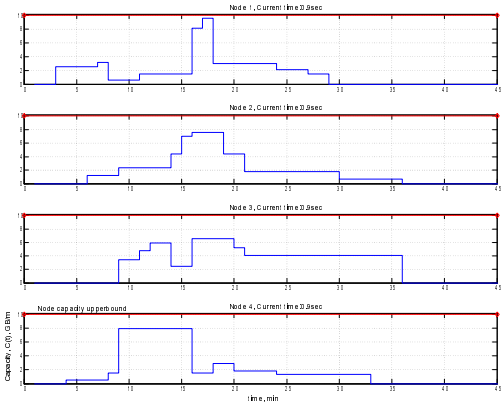
(a) Capacity reservation state of 4 source nodes, MC



(b) Capacity reservation state of 4 destination nodes, MC



(c) Capacity reservation state of 4 source nodes, LB



(d) Capacity reservation state of 4 destination nodes, LB

Figure 4.5: State of the capacity reservation over 20 file transfer requests with 1 rejection

ated, and one rejected FTR is seen. From Fig. 4.5(a) and 4.5(b), it can be seen that the $VP_{(4,1)}$ is relatively highly utilized than other possible node pairs so that $VP_{(3,4)}$ becomes a strong potential candidate especially for the future FTRs demanding high capacity. This corresponds with the intent of the MC heuristic. Compared with the MC heuristic, capacity reservation using the LB heuristic tends to be made over all possible VPs as shown Fig. 4.5(c) and 4.5(d). It can be seen that the capacity reservations are made more evenly through all VPs through all possible node pairs relative to the MC heuristic case. As a result, biased capacity reservation on a certain node pair illustrated in the case of the MC heuristic is not generated by using the LB heuristic, and a balanced capacity reservation is accomplished. To see clearly the different characteristic in terms of the capacity reservation, we introduce the node utilization trend, which is the frequency that each node is involved in the established VPs. The average node utilization trend is obtained as

$$\frac{\sum_{run} \frac{Total\ \ frequency\ \ of\ \ node\ \ particapation\ \ of\ \ VP\ \ setup}{Total\ \ number\ \ of\ \ the\ \ established\ \ VP}}{Total\ number\ of\ simulation\ runs} \tag{4.10}$$

Fig. 4.6(a) and 4.6(b) describe the average node utilization trend for BE(*best effort*)-MC and BE(*best effort*)-LB, respectively. Fig. 4.7(a) and 4.7(b) describe the average node utilization trend for MC heuristic and LB heuristic, respectively. The simulation parameters including the file profile are given in Table 4.1. The four bars at each grid shows the value of the highest node utilization rate to the lowest one among the 4 nodes on the source site. As comparing Fig. 4.6(a) to 6(b), and Fig. 4.7(a) to 4.7(b), we can observe that the average node utilization rate is similar with each other for the LB heuristic. It shows the balanced capacity reservation trend. On the other hand, the average node utilization rate for the MC heuristic shows a distinct gap between the node with the highest node utilization rate and the node with the lowest one. It clearly shows the different reservation trends of the MC heuristic and the LB heuristic as we described before. We can see that the different trends are more distinguishable especially at relatively small size file reservations. Intuitively, this is because the probability of the random VP selection increases as the file size increases

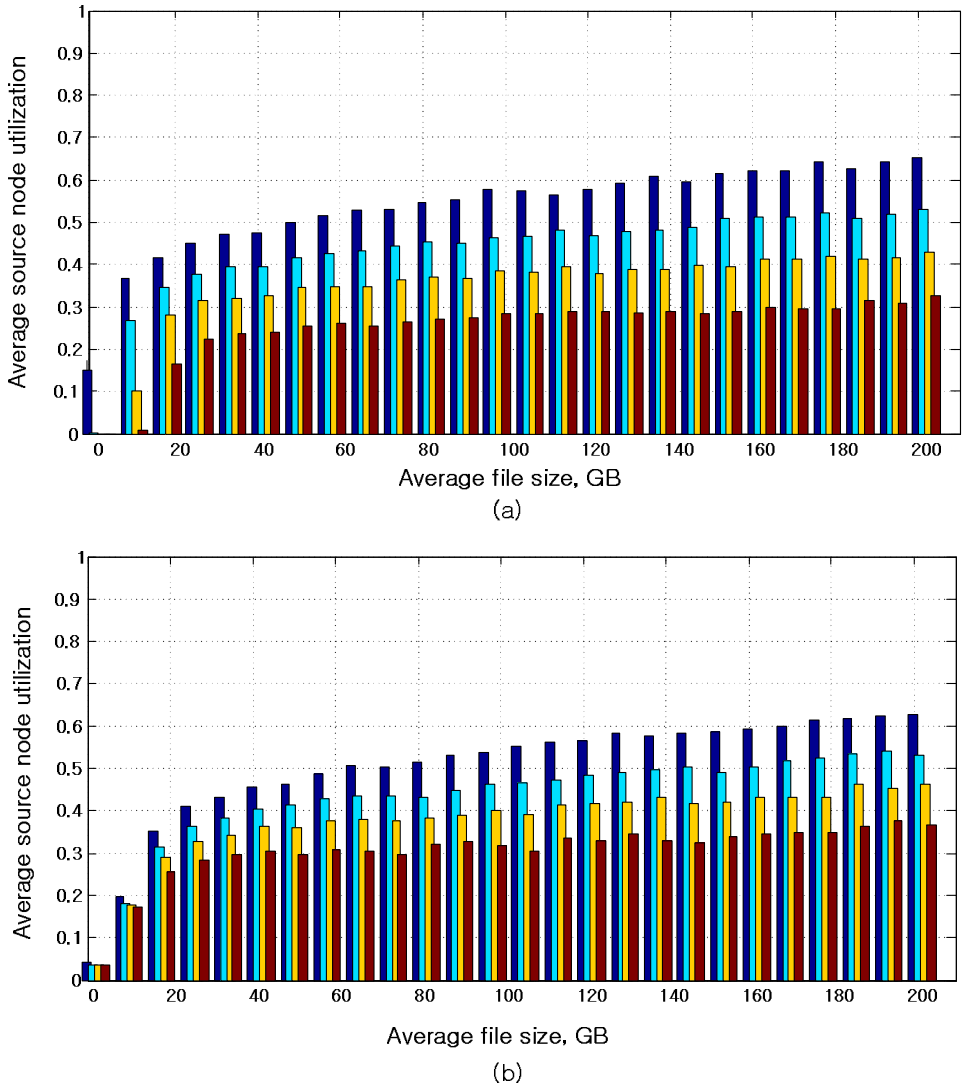


Figure 4.6: Average node utilization vs. file size, (a) BE-MC. (b) BE-LB.

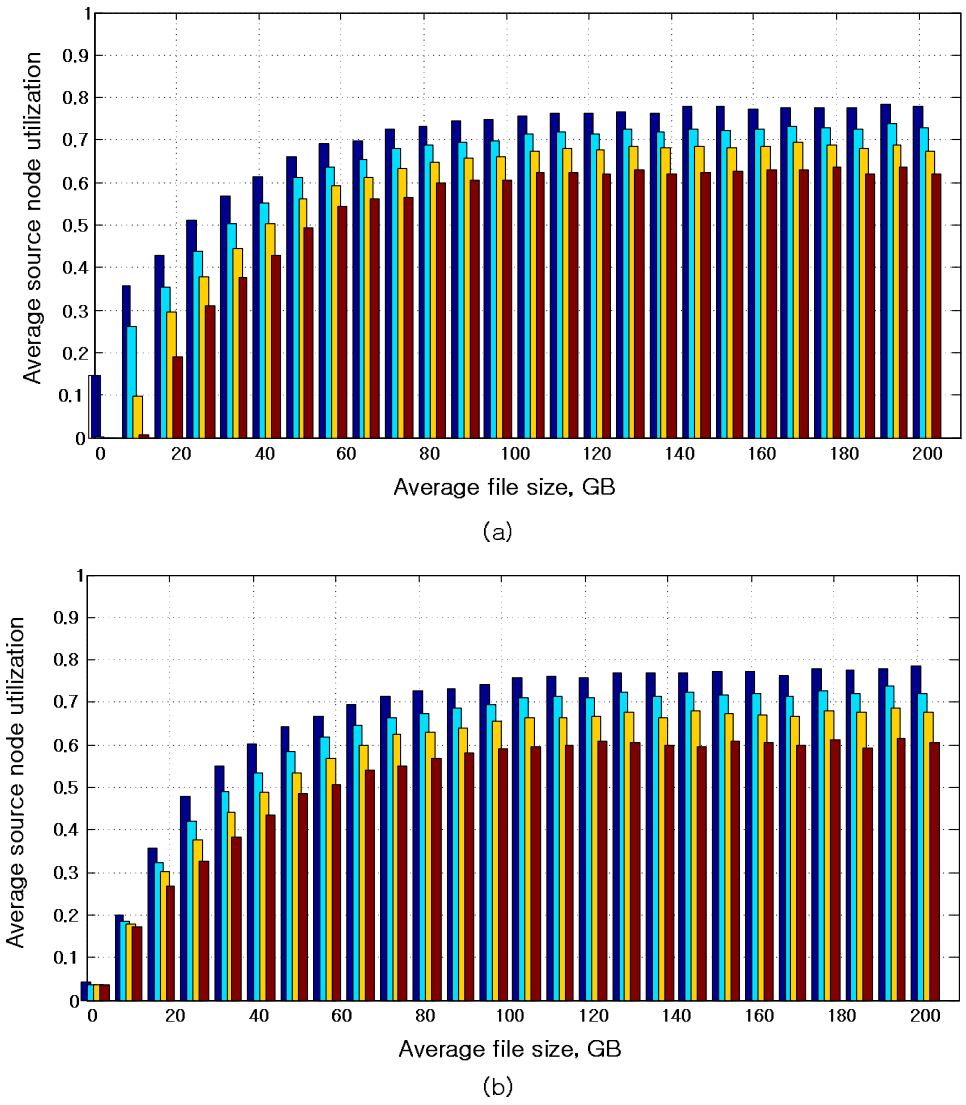


Figure 4.7: Average node utilization vs. file size, (a) MC. (b) LB.

according to the heuristic modules we introduced. As compared to Fig. 4.6, the overall higher node utilization rate in Fig. 4.7 shows that the time slice modification contributes to more VP establishment.

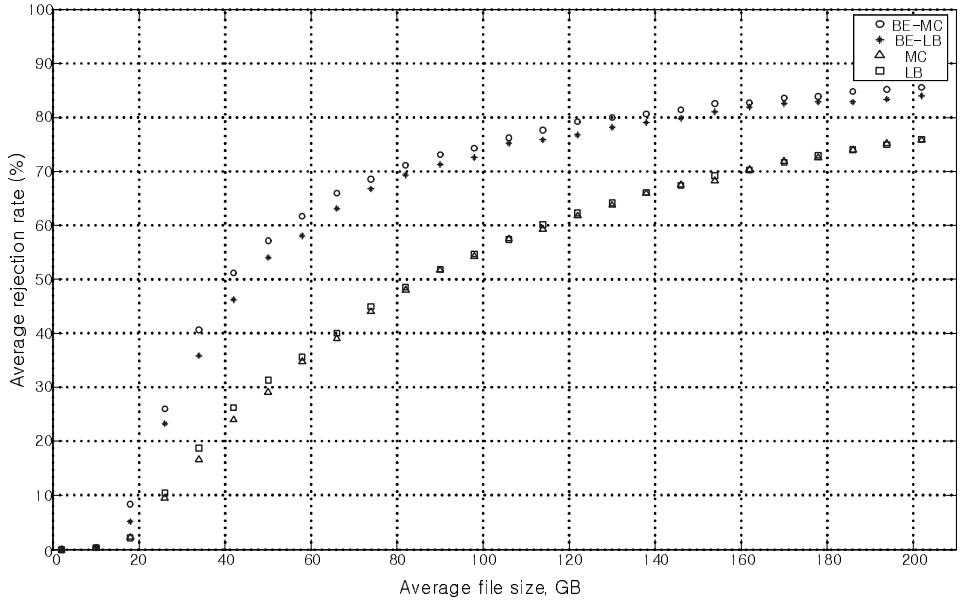
4.4.2 File size variation

The simulation parameters including the file profile are given in Table 4.1. The average rejection rate and average time slice modification rate against the file size varying from 2 GB to 202 GB for the four heuristic, BE-MC, BE-LB, MC, and LB are plotted in Fig. 4.8(a) and 4.8(b) respectively. The average rejection rate is obtained as

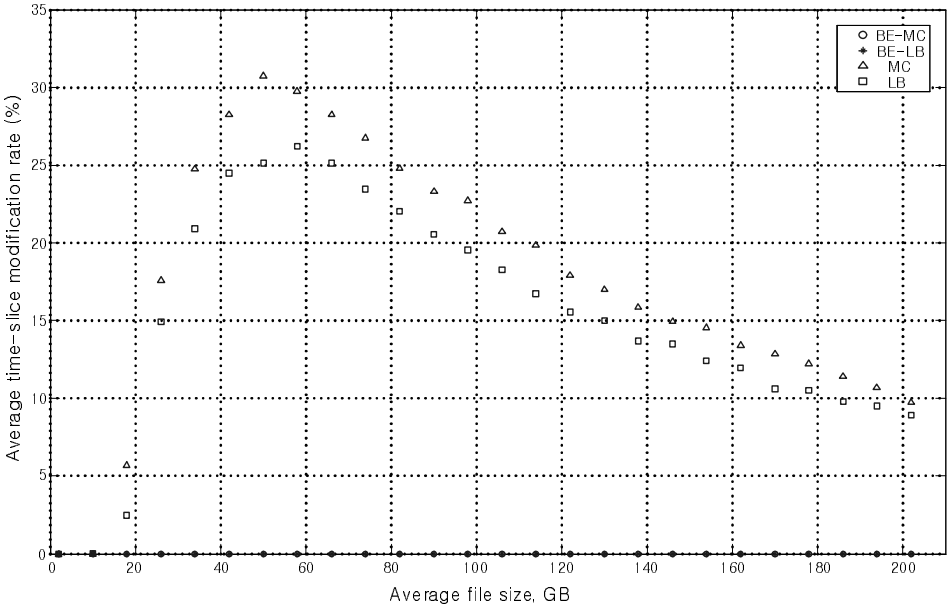
$$\frac{\sum_{run} \frac{Total\ number\ of\ the\ rejected\ FTR}{Total\ number\ of\ the\ investigated\ FTR}}{Total\ number\ of\ simulation\ runs} \tag{4.11}$$

From Fig. 4.8(a), we can observe that LB heuristic has a slightly smaller rejection rate than the MC heuristic in the case that there is no modification, best effort scenario. With the process of the time slice modification, the overall rejection rate decreases as compared with the best effort scenarios. Interestingly, the rejection rate of the LB heuristic is about the same as the one of the MC heuristic. Intuitively, the reservations based on the MC heuristic without the time-slice modification tend to stack the reservations on the busiest VP in which the reserved capacity stack reaches the highest point. Hence, the available capacity will run out faster relative to the case of the LB heuristic during busy time periods. On the other hand, with the time-slice modification process, the reserved capacity tends to be squeezed more densely in time rather than stacked in an appropriate capacity available area. As we expect, the rejection rate becomes higher as the file size increases. The average time-slice modification rate shown in Fig. 4.8(b) is obtained as

$$\frac{\sum_{run} \frac{Total\ number\ of\ the\ time-slice\ modifications}{Total\ number\ of\ the\ investigated\ FTR}}{Total\ number\ of\ simulation\ runs} \tag{4.12}$$



(*a*)



(*b*)

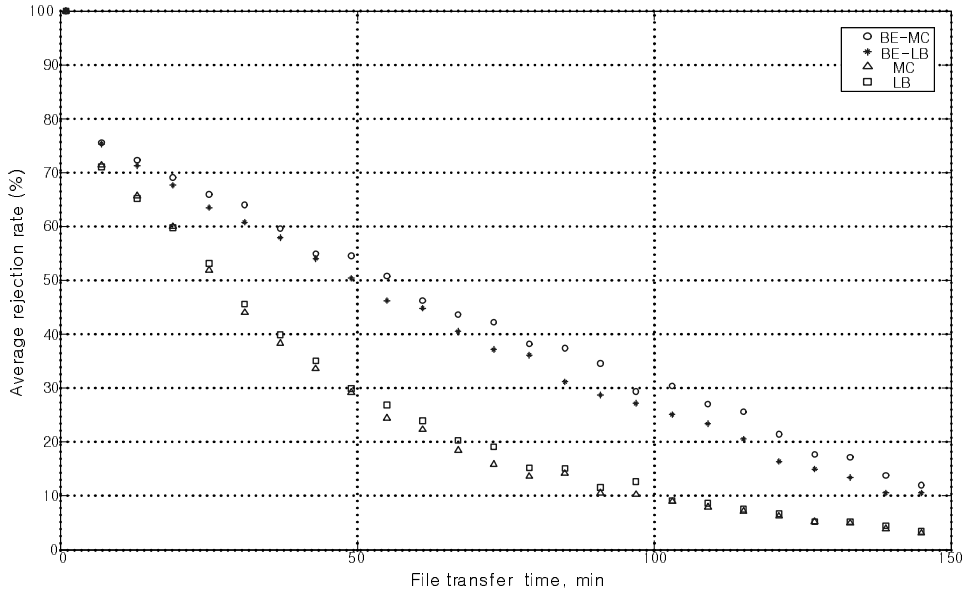
Figure 4.8: (a) Average rejection rate vs. file size, f. (b) Average time-slice modification rate vs. file size, f.

As we expect, the time-slice modification rate of the two BE heuristics is zero because the BE heuristic has no modification. For the MC and the LB heuristics the modification rate curves are in a convex shape. Intuitively, the initial increase is due to the increasing need for time slice modification as the file size increases, and the trend of the decreasing rejection rate can be interpreted as the case that the FTR rejection rate becomes dominant. The reason that MC heuristic has a somewhat higher time-slice modification rate than the LB heuristic can be obtained from the reasoning on the rejection rate. Based on the same probability of the time-slice modification demand, the MC heuristic tends to have a higher probability to make a reservation via the time-slice modification. This corresponds the less rejection rate as compared with the LB heuristic. From Fig. 4.8(b), it can be reasoned that FTRs with approximately 0.75 GB/m ($\approx 45GB$ / $60min$) capacity demand are most likely to be reserved after the time-slice modification than FTRs with other amounts of capacity demand.

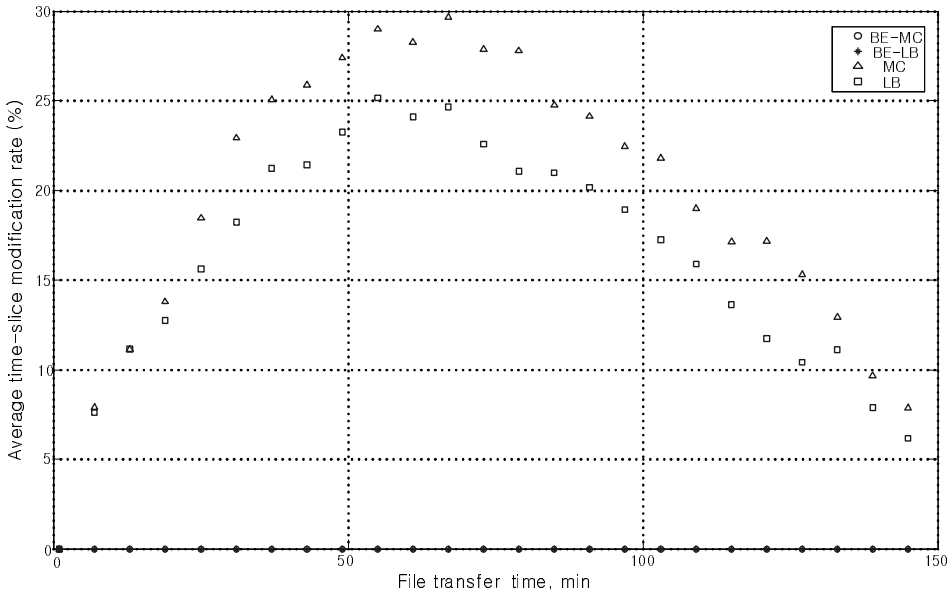
4.4.3 File transfer time variation

Based on the parameters in Table 4.1, the average rejection rate and average time slice modification rate against the file transfer time varying from 1 min to 145 min for the four heuristic are plotted in Fig. 4.9(a) and Fig. 4.9(b) respectively. As we can expect, the rejection rate decreases due to the fact that the capacity demand decreases as the file transfer time duration increases with a given file size are illustrated in Fig. 4.9(a). The idea that the process of the time-slice modification enhances the performance in terms of the rejection rate also can be seen as in the previous experiment. The explanation for the details about the curves between the MC heuristic and the LB heuristic is based on similar reasoning as mentioned in the previous section.

The curves of average time-slice modification rate shown in Fig. 4.9(b) are also convex in shape. In this case, the initial increase is due to the increasing demand for time-slice modifications caused by decreasing FTR capacity demand as file transfer time increases for a given file size. The trend of decreasing rejection rate appears because a capacity reserva-



(*a*)



(*b*)

Figure 4.9: (a) Average rejection rate vs. file transfer time, $T_D - T_S$. (b) Average time-slice modification rate vs. file transfer time, $T_D - T_S$.

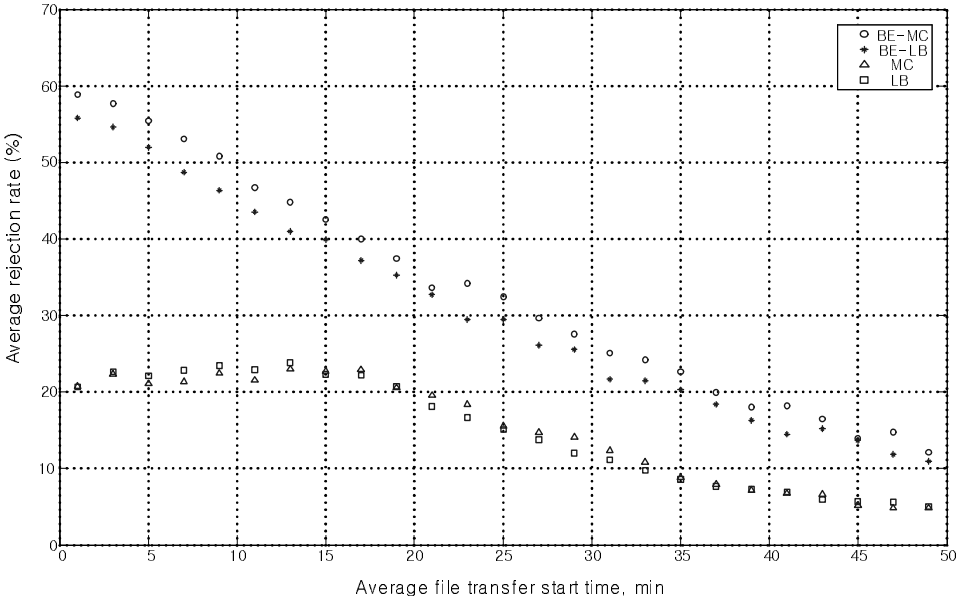
tion is likely to be accomplished without time-slice modification as demand for capacity is dramatically reduced with longer file transfer times. From Fig. 4.9(b), it is also seen that FTRs with approximately 0.75 GB/m ($\approx 40GB$ / $55min$) capacity demand are most likely to be reserved after the time-slice modification than FTRs with other amounts of capacity demand for the parameters of the simulation.

4.4.4 File transfer start time variation

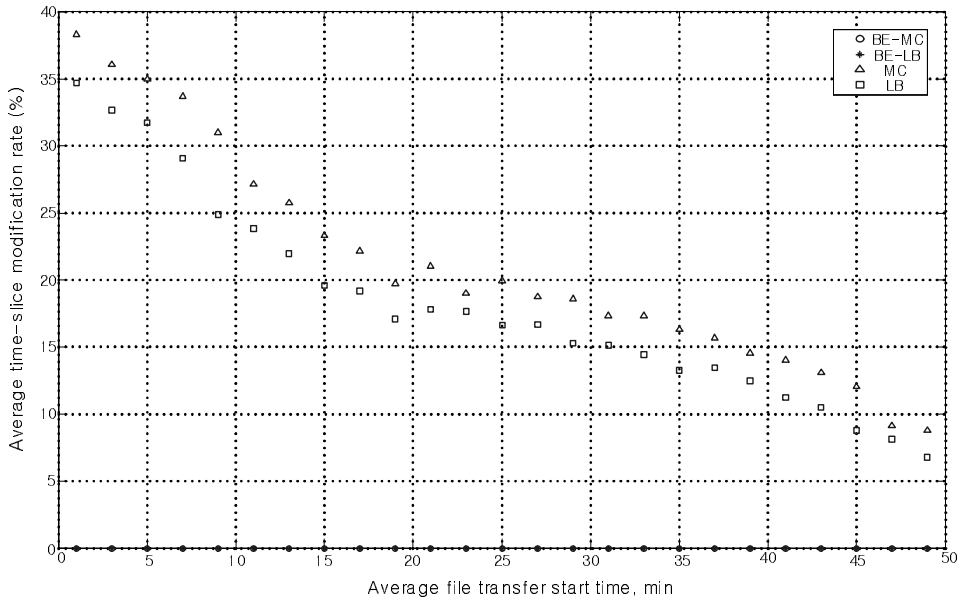
Based on the parameters in Table 4.1, the average rejection rate and average time slice modification rate is obtained against the file transfer time varying from 1 min to 49 min for the four heuristic in Fig. 4.10(a) and Fig. 4.10(b) respectively. It can be seen that the rejection rate for all heuristics tend to decreases as the file transfer start time, T_s increases. This is because the later the value of T_s, the higher the chance to reserve capacity on the less busy time slice. The reasoning about the rejection rate curves between the BE-MC heuristic and the BE-LB heuristic is based on the similar reasoning mentioned in the previous section. However, the information of T_s is insufficient for the performance comparison in terms of the rejection rate of the MC and LB heuristics as shown in Fig. 4.10(a). Fig. 4.10(b) gives us an information that the MC heuristic tends to accomplish the capacity reservation by utilizing the time-slice modification module as compared with the LB heuristic. The decreasing trend of the time-slice modification rate as the file transfer start time increases reflects the fact that the capacity reservation of the FTRs with the later T_S is likely made without the time-slice modification.

4.5 Concluding remarks

An initial study of two heuristic methods for optimizing network resource reservations for virtual paths carrying file transfers between end-site node clusters has been presented. The MC heuristic, which maximizes the utilization of network resources of the cluster interconnecting network, and the LB heuristic, which spreads transfer load evenly among source and



(*a*)



(*b*)

Figure 4.10: (a) Average rejection rate vs. file transfer start time, T_S. (b) Average time-slice modification rate vs. file transfer start time, T_S.

destination cluster nodes, were shown to be effective to reduce file transfer request rejection rate when used with a time slice modification algorithm. This study indicates that the use of time slice modification has a bigger impact on performance than the choice of either the MC or LB heuristic. The proposed heuristic algorithms are promising as a starting point for studying the scheduling of file transfers through modern hybrid high-performance networks.

Potential extensions to this starting point include:

- Improving the computational efficiency of the scheduling algorithm.
- Simulating larger networks.
- Simulating networks with different statistical assumptions on file transfer request arrival processes and file size.
- The MC and LB heuristics are at two extremes of possible algorithms. Blended algorithms or algorithms with a different basis would be of interest for further study.

Collaborations of researchers sharing the results of large scale experiments are leading the way in massive file sharing (in file quantity and file size) through networks. This research is a step in engineering such large file transfer capable networks.

Chapter 5

Grid Scheduling Divisible Load with Load Adaptive Computing Power

A special type of parallel computing system, "Grid" computing systems, has appeared as a promising trend for distributed parallel processing systems. Grid computing is applying the resources of many computers in a network to a single task at the same time. For collaborative grid computing across many computers, each computer is multiprogrammed. Multiprogramming is the technique of running several programs at a time using timesharing. It allows a computer to do several tasks at the same time. Thus, multiprogramming creates logical parallelism.

In recent years, there have been several interesting studies in divisible load grid scheduling. Environments of multiple job submissions are investigated through stochastic queuing models in [59]. Grid load scheduling algorithms under buffer resource space constraint are examined in [60],[61]. In [62], divisible load scheduling strategies in grid networks is studied via linear programming. Under a certain constraint, optimal load solution is analyzed in [63]. Flow cost analysis is performed for the case of multi-source load distribution in linear daisy chain topologies [64]. Divisible load scheduling is introduced to design grid systems [65].

To this end, in this chapter optimal computing power allocation solution adapted to divisible load the parallel computing grid is developed in a fundamental grid with two sources

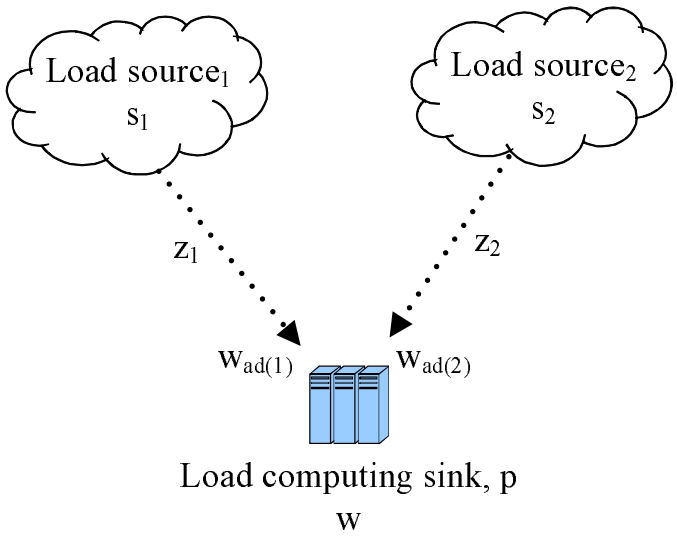


Figure 5.1: Grid network with two load sources and a load computing sink.

and a single sink computing processor. We note that the analysis of the optimal computing power allocation is provided for the first time.

The organization of this chapter is follows. Section 5.1 presents the types of notations and analytic backgrounds, Section 5.2 analyzes the optimal adaptive computing power based on the divisible load theory. Performance simulation results appear are discussed in Section 5.3. Finally, this chapter concludes with Section 5.4.

5.1 Problem Formulation and Preliminary Remarks

The grid network discussed in this chapter is a grid network consisting of two workload sources and a single sink (Fig. 5.1). The sink p is connected to the sources through 2 communication links. Each of the sources has role as a workload distributor with no computing itself. The two sources distribute optimally calculated load fractions to the sink. The sink is involved in computing its own load assigned from the sources via the direct links. The sink is a processor which does workload processing. The sink is a multiprogrammed (multitasking) processor. Multiprogrammed processor allows simultaneous computing of multiple loads distributed from sources (i.e., parallel computing). Simultaneous multiple load computing has *background* loads being computed (i.e., sharing the computing power (speed) at the

same time. Thus, the entire computing power (speed) of the sink is differentially consumed for the each of the parallel computing loads. The objective of this study is the optimal load scheduling through sharing adaptive computing power (speed) of the sink.

The following notation is used in this chapter.

i : The index of source, where $i = 1, 2$.

s_i : The ith source (load distributor).

p : The sink (load computing processor).

w : The inverse maximum computing speed (power) of the sink.

z_i : The inverse communication speed of the link connecting the ith source and the sink.

T_{cp} : Computing intensity constant. The entire load can be computed on the sink in time wT_{cp}.

T_{cm} : Communication intensity constant. The entire load can be transmitted over the i^{th} link in time zT_{cm}.

α_i : The load fraction assigned to the sink from the ith source.

$\Delta\alpha_i$: The fraction of α_i which is processed with the inverse computing speed $w_{ad(i)}$. Here, $0 \leq \Delta\alpha_i \leq \alpha_i$.

$w_{ad(i)}$: The adaptive inverse computing speed (power) for the α_i. Here, $w_{ad(i)} > w$

t : Parallel computing start time at which the computation of the load fractions assigned from the both sources starts simultaneously at the sink.

C : Parallel computing time region at which the adaptive computing speed (power) is applied.

T_f : The total time that elapses between the beginning of the process at $t = 0$ and the time when the sink completes its entire computation.

T_i : The total computing finish time of the α_i. Here, $T_i \leq T_f$.

T_{ave} : The average of load weighted computing finish time at the sink

$$T_{ave} = \sum_{i=1}^{2} \alpha_i T_i \tag{5.1}$$

In this chapter, the following assumptions are initially made:

1. The sources are capable to distribute their own loads to the sink simultaneously (i.e., simultaneous distribution, $t = 0$). This is possible as long as a source is fast enough to continually load buffers for each of its output links. The load allocation time instant of each load can differ.

2. Without loss of generality, the load distribution from the two sources initiates by s_1 at $t = 0$. The computing process of the loads from s_2, α_2 is undertaken basically under the circumstance that the α_1 is currently being computed.

3. The second load, α_2 allocation must be undertaken before the computing of the first load, α_1 ends (i.e., $t \leq \alpha_1 w T_{cp}$).

4. The sink begins computing as soon as it begins to receive load from the either of two sources.

5. The speed of communication in a link is faster than the speed of computation of the sink, which is connected to the link (i.e., $z_i \ll w$).

6. At the sink, the computing of the first assigned load, α_1 can not extend beyond the time at which computing of the subsequent load, α_2 terminates at least.

5.2 Analysis of Adaptive Computing Speed

Consider a simple scenario of a grid networking with two sources (s_1 and s_2) and a single sink, p. Initially the load α_1 is distributed to the sink p from the source s_1. As the s_1 begins to distribute the load α_1, the sink p starts computing α_1. At time t, sink p begins to compute the second load α_2 as it receives the second load from s_2 while the first load is still being computed (parallel computing). The parallel computing is possible in that p is a multiprogrammed processor. Now, the background computing process of the load α_1 affects the computing power allocation for simultaneous parallel computing of the α_2. The corresponding timing diagram of the scenario is shown in Fig. 5.2. The shaded time area in

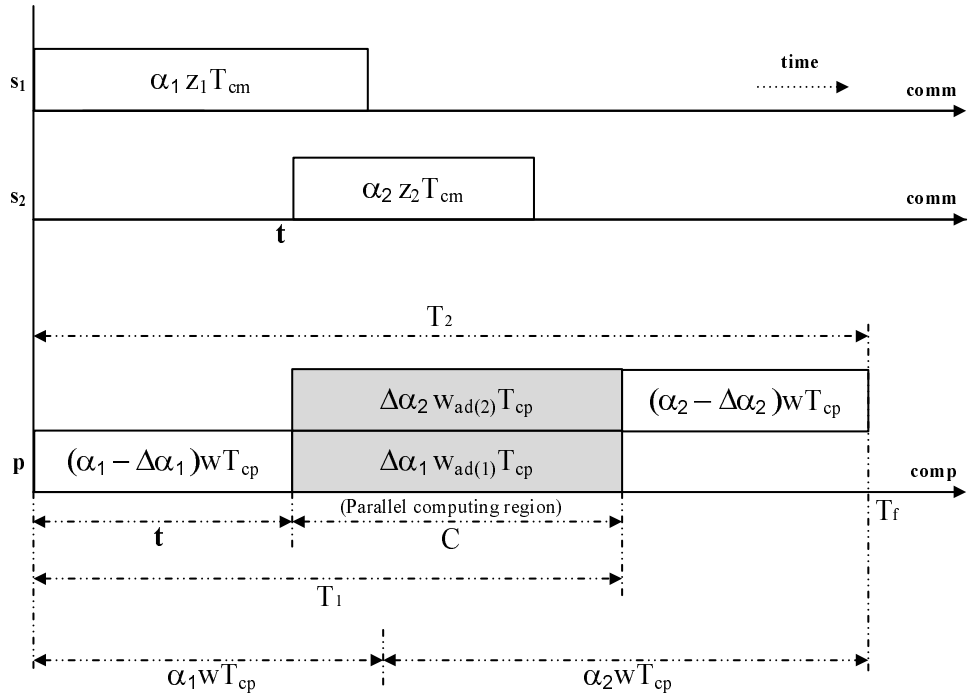


Figure 5.2: Generalized load distribution with two sources and a multiprogrammed sink.

Fig. 5.2 demonstrates the time at which the simultaneous parallel computing is performed through the adaptive computing speed (power) allocation.

From the timing diagram, we can see that,

$$(\alpha_1 - \Delta\alpha_1)wT_{cp} = t \tag{5.2}$$

By rewriting eq (5.2), $\Delta\alpha_1$ can be expressed as

$$\Delta\alpha_1 = \frac{\alpha_1 wT_{cp} - t}{wT_{cp}} \tag{5.3}$$

Since $0 \leq \Delta\alpha_1 \leq \alpha_1$, the bound of feasible t is given from eq (5.3) as

$$0 \leq t \leq \alpha_1 wT_{cp} \tag{5.4}$$

which implies that idle time during which the sink waits for being fed with a subsequent load is not allowed (assumption 3). The simultaneous start of load computing (i.e., $t = 0$)

involves the entire load α_1 being computed with $w_{ad(1)}$ (i.e., $\Delta\alpha_1 = \alpha_1$) . The case $t = \alpha_1 w T_{cp}$ involves that the second load α_2 being assigned when the computing of the load α_1 with the full computing power ends (i.e., $\Delta\alpha_1 = \Delta\alpha_2 = 0$) so that the full computing power is guaranteed for the second load α_2 (i.e., sequential load computing).

The next step is to find an expression for the adaptive inverse computing speed (power) for the α_1, $w_{ad(1)}$. As it can be seen from Fig. 5.2, the shaded time area in which the adaptive computing speed is applied has a constant time value, C. Thus,

$$\Delta\alpha_1 w_{ad(1)} T_{cp} = \Delta\alpha_2 w_{ad(2)} T_{cp} = C \tag{5.5}$$

Here, $C = 0$ if and only if $t = \alpha_1 w T_{cp}$.

By substituting eq (5.3) into eq (5.5), $w_{ad(1)}$ can be expressed as w proportional to a time ratio (see Fig. 5.2)

$$w_{ad(1)} = \frac{C}{\Delta\alpha_1 T_{cp}} = \left(\frac{C}{\alpha_1 w T_{cp} - t}\right) w \tag{5.6}$$

From the reasoning that the sum of the adaptive computing power dedicated to the α_1 and α_2 is the maximum computing power of the sink, we obtain

$$\frac{1}{w_{ad(1)}} + \frac{1}{w_{ad(2)}} = \frac{1}{w} \tag{5.7}$$

By substituting eq (5.6) into eq (5.7), $w_{ad(2)}$ can be obtained as the w proportional to a time ratio (see Fig. 5.2) as similar to the $w_{ad(1)}$

$$w_{ad(2)} = \left(\frac{C}{C - \Delta\alpha_1 w T_{cp}}\right) w = \left(\frac{C}{C - (\alpha_1 w T_{cp} - t)}\right) w \tag{5.8}$$

Since $\Delta\alpha_2 w_{ad(2)} T_{cp} = C$ (the shaded time area in Fig. 5.2), from eq (5.8), $\Delta\alpha_2$ can be achieved as

$$\Delta\alpha_2 = \frac{C - (\alpha_1 w T_{cp} - t)}{w T_{cp}} \tag{5.9}$$

Further, from $0 \leq \Delta\alpha_2 \leq \alpha_2$, the boundary condition for the feasible C can be achieved as

$$\alpha_1 w T_{cp} - t \leq C \leq (\alpha_1 + \alpha_2) w T_{cp} - t \tag{5.10}$$

Intuitively, the boundary information of the C seems reasonable due to the assumption that the computing time of the α_1 can not extend beyond the time at which computing of the subsequent load, α_2 terminates as was mentioned previously.

From eq (5.3) and eq (5.9), the ratio of the load fractions sharing the computing power (speed) can be expressed as a time ratio

$$\frac{\Delta\alpha_2}{\Delta\alpha_1} = \frac{C - (\alpha_1 w T_{cp} - t)}{\alpha_1 w T_{cp} - t} \tag{5.11}$$

Also, from eq (5.5), we have

$$\frac{w_{ad(1)}}{w_{ad(2)}} = \frac{\Delta\alpha_2}{\Delta\alpha_1} = \frac{C - (\alpha_1 w T_{cp} - t)}{\alpha_1 w T_{cp} - t} \tag{5.12}$$

By intuition, it can be reasoned that the adaptive computing speed is commensurate with the amount of the load fraction to be computed. Here, interestingly it can be seen that the adaptive inverse computing speed, $w_{ad(1)}$ and $w_{ad(2)}$ can be written as a function of w that linearly depends on the ratio of $\Delta\alpha_1$ and $\Delta\alpha_2$. From eq (5.3), eq (5.6), eq (5.8), and eq (5.9)

$$w_{ad(1)} = \left(\frac{\Delta\alpha_1 + \Delta\alpha_2}{\Delta\alpha_1}\right) w, \quad w_{ad(2)} = \left(\frac{\Delta\alpha_1 + \Delta\alpha_2}{\Delta\alpha_2}\right) w \tag{5.13}$$

The eq (5.13) shows that the adaptive computing speed (power) linearly depends on the fractions of load to be parallel computed.

From Fig. 5.2, the entire load processing (computing) finish time, T_f is given by

$$T_f = t + C + (\alpha_2 - \Delta\alpha_2) w T_{cp} \tag{5.14}$$

From eq (5.9), eq (5.14) can be rewritten as

$$T_f = (\alpha_1 + \alpha_2) w T_{cp} \tag{5.15}$$

The above eq (5.15) gives the intuition that the entire load processing time, T_f equals to that of the case of the full computing power, wT_{cp} dedicated to the entire load $\alpha_1 + \alpha_2$, which is the optimal case. In other words, the eq (5.6) and eq (5.8) yield the adaptive inverse computing speed for the loads $\Delta\alpha_1$ and $\Delta\alpha_2$.

Now, from Fig. 5.2, the average load weighted computing finish time, T_{ave} defined as eq (5.1) can be expressed as

$$T_{ave} = \sum_{i=1}^{2} \alpha_i T_i = \alpha_1 (t + C) + \alpha_2 T_f \tag{5.16}$$

Here, our objective is to determine the optimal fractions α_i satisfying the minimum average load weighted computing finish time. An equality constraint, $\alpha_1 + \alpha_2 = L$, where L is the total amount of the load at the sources, is essential for a unique solution. Using the equality constraint and eq (5.15), eq (5.16) can be rewritten as

$$T_{ave} = \alpha_1 (t + C) + (L - \alpha_1) L w T_{cp} \tag{5.17}$$

From eq (5.17), it is clear that the lower bound of C strictly contributes to the minimization of T_{ave}. The condition $C = \alpha_1 w T_{cp} - t$ (see eq (5.10)) implies that the sink devote its full inverse computing speed (power), w to process the entire α_1 (i.e, $w_{ad(1)} = w$ from eq (5.6)). In other words, the computing of the α_2 is delayed until the computing of α_1 ends even though the α_2 arrives at time instant t (i.e, $w_{ad(2)} = \infty$ from eq (5.8)). In a nutshell, the minimum average load weighted computing finish time can be achieved through sequential computing scenario regardless of t.

Substituting C with $\alpha_1 w T_{cp} - t$, the T_{ave} can be reformulated as a downward concave function

of a variable α_1

$$T_{ave} = \alpha_1^2 w T_{cp} - \alpha_1 L w T_{cp} + L^2 w T_{cp} \tag{5.18}$$

As special cases, consider the case in which the computing power is evenly allocated for the each of the loads that is $w_{ad(1)} = w_{ad(2)} = 2w$ (leading $\Delta\alpha_1 = \Delta\alpha_2$). From the condition eq (5.12)

$$C = 2(\alpha_1 w T_{cp} - t) \tag{5.19}$$

By substituting eq (5.19) into eq (5.17), T_{ave} can be written as

$$T_{ave} = 2\alpha_1^2 w T_{cp} - \alpha_1(t + L w T_{cp}) + L^2 w T_{cp} \tag{5.20}$$

So far, the analysis of the minimum average load weighted computing finish time has implied, t is a known priori. To generalize the analysis without the assumption, a special type of mathematical optimization model, Quadratic programming (QP) can be adopted. In our case, it is the problem of optimizing (minimizing) a quadratic objective function, eq (5.17) of four variables, $[\alpha_1, \alpha_2, t, C]$ subject to linear constraints on these variables :
Equality constraint :

$$\alpha_1 + \alpha_2 = L \tag{5.21}$$

Linear Inequality Constraints :

$$\begin{aligned} 0 &< \alpha_1 < L \\ 0 &< \alpha_2 < L \\ 0 &\leq t \leq \alpha_1 w T_{cp} \\ \alpha_1 w T_{cp} - t &\leq C \leq (\alpha_1 + \alpha_2) w T_{cp} - t \end{aligned} \tag{5.22}$$

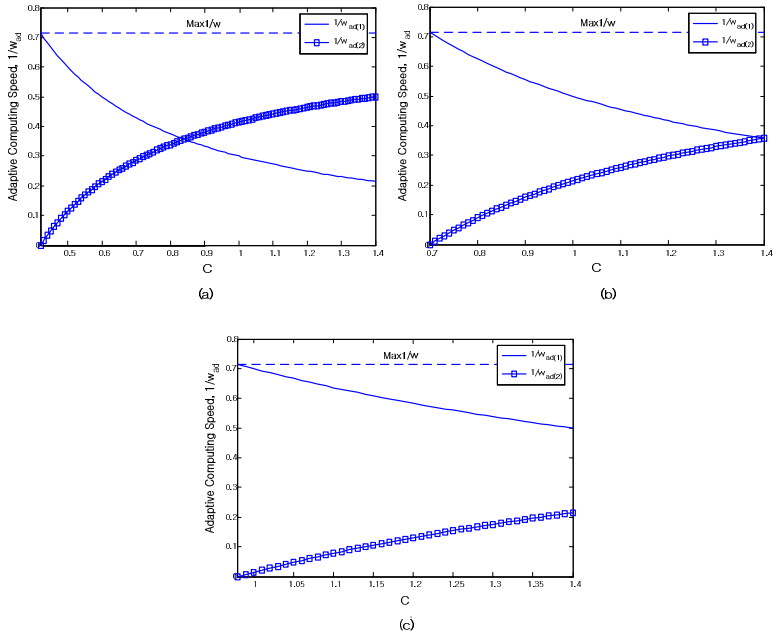


Figure 5.3: Adaptive computing speed against C, $t = 0$, (a) $\alpha_1 < \alpha_2$, (b) $\alpha_1 = \alpha_2$, (c) $\alpha_1 > \alpha_2$.

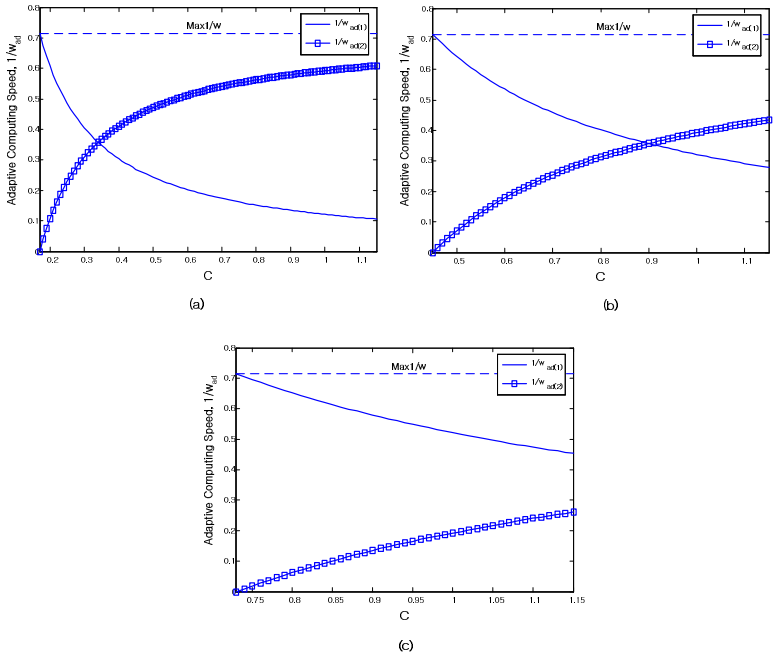


Figure 5.4: Adaptive computing speed against C, $t = 0.25$, (a) $\alpha_1 < \alpha_2$, (b) $\alpha_1 = \alpha_2$, (c) $\alpha_1 > \alpha_2$.

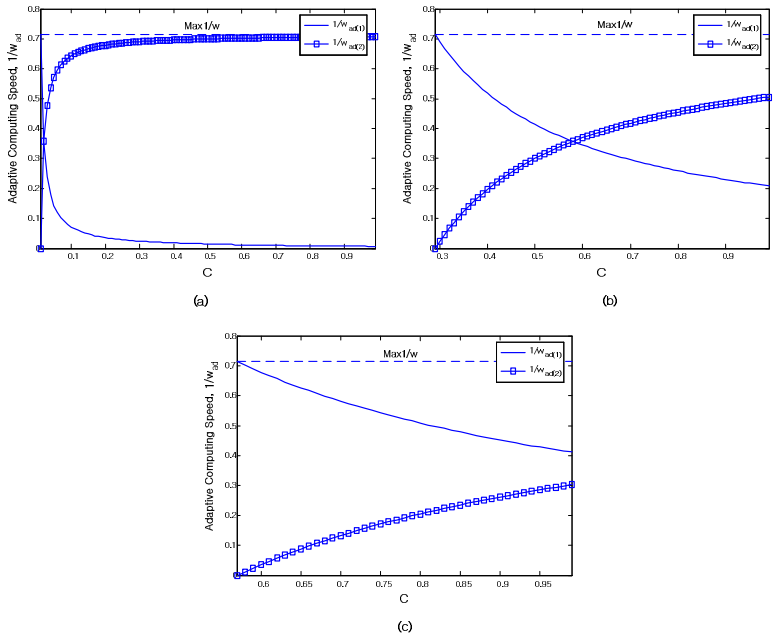


Figure 5.5: Adaptive computing speed against C, $t = 0.41$, (a) $\alpha_1 < \alpha_2$, (b) $\alpha_1 = \alpha_2$, (c) $\alpha_1 > \alpha_2$.

5.3 Performance evaluation

A simulation is performed in the case where there are two sources and a single sink with parameters $w = 1.4$, $T_{cp} = 1$, $a_1 = 0.3, 0.5$, and 0.7, the corresponding $a_2 = 0.7, 0.5$, and 0.3, and the $t = 0, 0.25$, and 0.41, respectively. Thus, the minimum computing finish time, $T_f = 1.4$. Fig. 5.3, Fig. 5.4, and Fig. 5.5 show the dependence of the adaptive computing speed (power) on the parallel computing time region, C against different parallel computing start times, t. Note that the range of C is different with each subplot from eq (5.10). The subfigures are presented against different load conditions (i,e., $a_1 < a_2$, $a_1 = a_2$, and $a_1 > a_2$). The initial superior adaptive computing speed (power) is due to the fact that the computing of the α_1 load already exists in the background while the application of full computing power $1/w$ is being assumed. In other words, the α_1 load has a priority in utilizing the full computing power of the sink, p. The increasing trend of the adaptive computing power for α_2 (or decreasing trend of adaptive computing power for α_1) against C is reasonable in that the relatively more adaptive computing speed (power) for the α_2 than the α_1 is required to meet the T_f within the bigger parallel computing time region C. As

Type	Boundary	α_1	$T_{ave}^{\min}$	α_2
Adaptive (Parallel Comp)	$t < LwT_{cp}/2$	$L/2$	$3L^2wT_{cp}/4$	$L-\alpha_1$
Adaptive (Sequential Comp)	$t \geq LwT_{cp}/2$	t/wT_{cp}	$t^2/wT_{cp} - Lt + L^2wT_{cp}$	
Even (Parallel Comp)	$t < LwT_{cp}/3$	$t + LwT_{cp}/4wT_{cp}$	$-t^2/8wT_{cp} - Lt/4 + 7L^2wT_{cp}/8$	
Even (Sequential Comp)	$t \geq LwT_{cp}/3$	t/wT_{cp}	$t^2/wT_{cp} - Lt + L^2wT_{cp}$	

Table 5.1: Analytic optimal solutions.

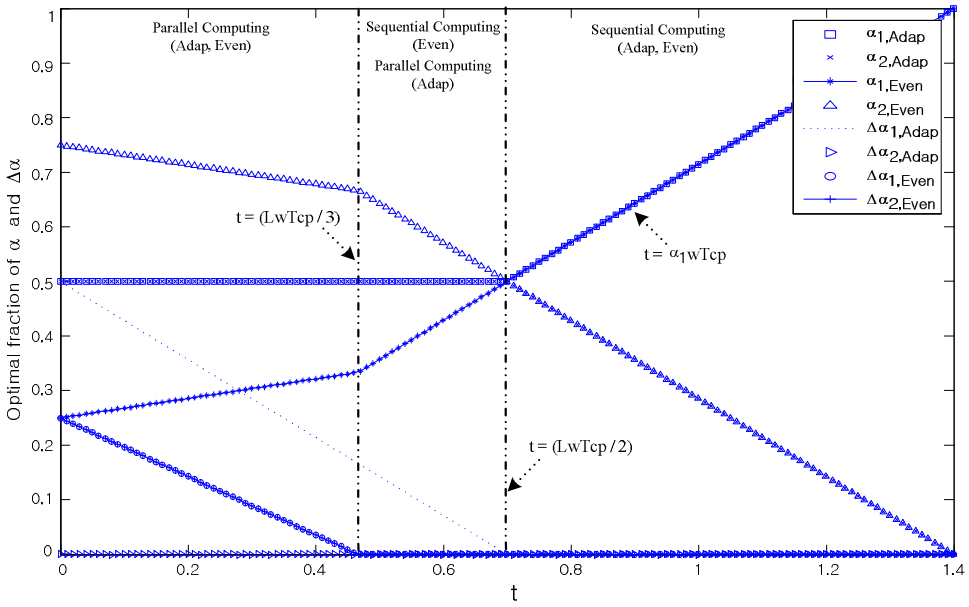


Figure 5.6: QP optimal load solution against t.

it was explained, the sum of the adaptive computing powers is $1/w$. It can be seen in the subplots with the same load conditions in Fig. 5.3, Fig. 5.4, and Fig. 5.5 against different t that the latter parallel computing starts, the more adaptive computing power α_2 needs. Intuitively, this is because the larger faction of the α_1 is likely to be computed utilizing the full computing power, as the α_2 requests the computing power allocation later. In Fig. 5.3, Fig. 5.4, and Fig. 5.5, given t and C, it can be seen that the amount of the adaptive power dedicated to either load is bigger when the load size of one is bigger than or equal to one of the other.

Table 5.1 contains the analytical solutions of optimal fraction of load to attain the min-

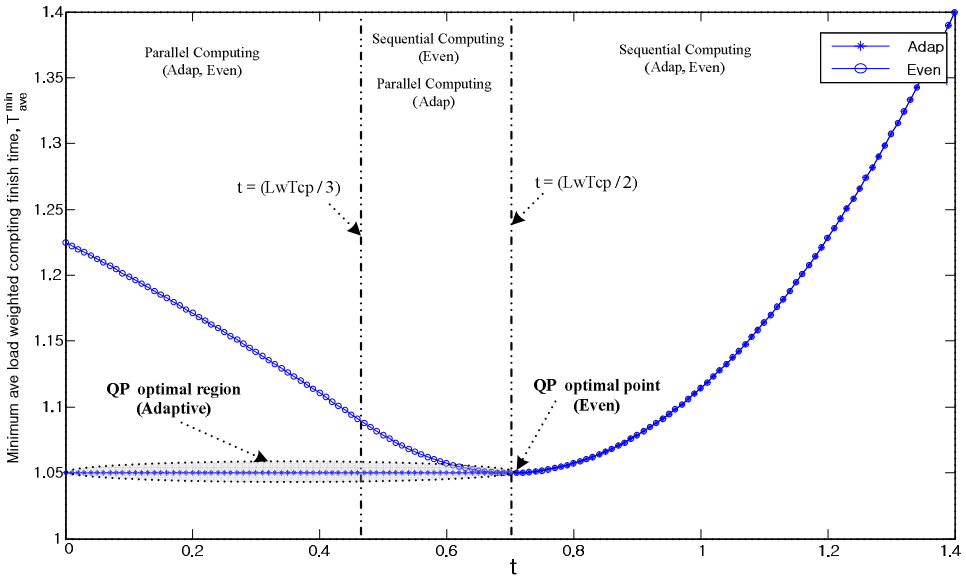


Figure 5.7: Minimum average load weighted computing finish time against t.

imum average load weighted computing finish time, T_{ave}^{min} for the both cases (i.e, adaptive computing power allocation, eq (5.18) and even computing power allocation, eq (5.20)). The optimal solution, α_1 satisfying the T_{ave}^{min}, can be achieved by the first derivative test. By substituting the optimal solutions for the both cases into eq (5.18) and eq (5.20), the T_{ave}^{min} can be obtained as shown in Table 5.1. The analytical point here is to take into account the boundary condition formulated from the constraint, $t \leq \alpha_1 w T_{cp}$ based on the initial assumption 3. The value t out of the constraint leads to the sequential load computing scenario. Thus, the solution of α_1 has upper bound, t/wT_{cp}. By equalizing the optimal load solution to the upper bound of α_1, the condition of the boundary time can be obtained as the second column of Table 5.1.

Fig. 5.6 shows the optimal load fraction for the case that the adaptive computing speed is utilized with parameters $L = 1$, $w = 1.4$, $T_{cp} = 1$, and known priori t varying 0 to 1.4. The optimal load solutions are obtained via QP with the objective functions eq (5.18) and eq (5.20) with two variables $[\alpha_1, \alpha_2]$. Clearly, the optimal solutions are distributed as corresponding to the analytical solutions described in Table 5.1. The solutions exist beyond the bound satisfying the constraint $t \leq \alpha_1 w T_{cp}$ (i.e., sequential computing region), the optimal solutions are restricted to the value t/wT_{cp} (see Fig. 5.6).

Fig. 5.7 depicts the minimum average load weighted computing finish time of the both cases (i.e, adaptive computing power allocation and even computing power allocation). Interestingly, we can observe that the T_{ave}^{min} is achieved until the boundary (i.e., $LwT_{cp}/2$) in the case of adaptive computing power. This implies multiple solutions for T_{ave}^{min} exist in QP. Based on Fig. 5.6, the fact that the same optimal solution, $\alpha_{1,Adap} = 0.5$ regardless t (i.e, $t \leq 0.7$, the boundary, $LwT_{cp}/2$) implicitly shows that the full adaptive computing speed (power) is dedicated to compute α_1 until the entire α_1 is being computed (i.e., $1/w_{ad(1)} = 1/w$) in spite of the fact that α_2 arrives to be computed at t (i.e., sequential computing). This corresponds to the QP solution region over $t = [0, LwT_{cp}/2]$ as shown in Fig. 5.7. In the case using even computing power, we can see that a unique solution, $\alpha_{1,Even} = 0.5$ exists at the boundary $LwT_{cp}/2$ (i.e., $t = 0.7$) which is involved in the region of the sequential computing scenario of the even computing power case. The outperformed performance of the adaptive computing power case is intuitive because the adaptive computing speed (power) contributes to utilizing full computing power. The identical performance in $t \geq LwT_{cp}/2$ is due to the fact that the sequential computing scenario without idle time between two computing processes is applied. As shown in Fig. 5.6, the sequential computing scenarios confines the solutions, $\alpha_{1,Adap}$ and $\alpha_{1,Even}$ to t/wT_{cp}. The identical performance can be also clarified from Table 5.1.

5.4 Concluding remarks

In this chapter, a mathematical analysis to obtain the adaptive computing power scheduling of a sink in a grid with two sources is performed in first time. An optimization problem to minimize the average load weighted computing finish time is developed via Quadratic Programming (QP). The performance evaluation of the adaptive computing power and the even computing power are provided. The theoretical findings demonstrate that the optimal solutions is obtained via QP.

Chapter 6

Cost Performance Analysis in Parallel Computing Networks with Divisible Load Scheduling

With the emergence of massive parallel processing networks, commercial network service providers must maintain a balance between the requirements for parallel computing efficiency and the deployment of new differentiated services. To achieve this balance between an existing network service and deploying new network services, monetary network cost inevitably needs to be considered. Because the demand for maintaining efficient parallel computing network service, one much have information about monetary network cost trends corresponding to network environmental changes.

There have been several attempts to give insight into monetary network cost using divisible load models [66],[67],[68]. In a bus oriented networks, optimal load distribution sequences and associate numerical algorithms in minimizing total computing cost are investigated in [66]. As extended works, by considering communication cost besides of the computing cost, heuristics for the optimal load sharing sequence is examined and studied in [67],[68]. As an extended approach of parallel network performance evaluation using DLT, the *efficiency* of parallel systems and the concept of isomaps is studied [70], [71] using the concept

of an isoefficiency function [72].

In this chapter, by applying DLT, we consider the problem of monetary network cost on a homogeneous single-level tree network. Our objective is to analyze trends of the monetary network cost against ratio of network speed parameters and to determine relationships between the network cost and the network environment. To be specific, two strategies regarding load distribution (sequential load distribution and simultaneous load distribution) are discussed. The cost trends for various network conditions are studied for the both cases of load distribution. It is apparent that quantitative monetary cost is important to scale cost demand to perform parallel processing under a limited monetary budget. Further, the quality of monetary usage is worth taking into account for the both cases of load distribution. In other words, how efficiently the monetary cost is spent for the parallel processing indicates the value of the unit cost. In accordance, the cost efficiency contributes as a good indicator of the quality of cost consumption in the parallel processing.

This chapter is organized as follows. Section 6.1 presents the type of notation and analytic background that are used throughout the chapter. The analysis of monetary cost and cost efficiency for different load distribution strategies is constructed based on the DLT in section 6.2. Trends of network cost and cost efficiency for both load distribution strategies are investigated and compared through simulation results in Section 6.3. Finally, this chapter concludes with some possible extensions in Section 6.4.

6.1 Problem Formulation and Preliminary Remarks

The distributed computing network model to be considered in this chapter is a single-level tree network (bus network) with homogeneous speed parameters and cost coefficients. The single-level tree network consists of children processors connected to a root processor via direct communication links. The root processor has a role as a load distributor and also for computing itself. The children processors are involved in computing their own load assigned from the root processor via the direct links. Monetary "cost" involving processors'

computing and communicating linearly depends on the speed of processors and links and the amount of work load.

The following notation is used in this chapter.

α_i : The load fraction assigned to the ith children processor (where $i = 1, 2, \ldots, N$). Index $i = 0$ is for root processor.

w : The inverse computing speed of the processors. [sec/load].

z : The inverse communication speed of the links. [sec/load].

T_{cp} : Computing intensity constant. [Dimensionless]. The entire load (amount of unity) can be processed on a processor in time wT_{cp} sec.

T_{cm} : Communication intensity constant. [Dimensionless]. The entire load (amount of unity) can be transmitted over a link in time zT_{cm} sec.

c_p : Static computing cost coefficient. [cost/sec]. The computing cost for the entire load (amount of unity) is $c_p w T_{cp}$ cost.

c_l : Static communication cost coefficient. [cost/sec]. The communication cost for the entire load (amount of unity) is $c_l z T_{cm}$ cost.

$T_{f,N}$: The total processing finish time. Time at which a single root processor and N children processors complete their computation.

T_i : The total time that elapses between the beginning of the process at $t = 0$ and the time when i^{th} processor completes its own computation (where $i = 0, 1, \ldots, N$).

6.2 Cost analysis of homogeneous single level tree networks

Consider a single level tree network architecture with N children processors, $P_1, P_2, \ldots, P_N$ interconnected through N links to a root processor, P_0. The root processor distributes loads among the N children processors through the N direct links.

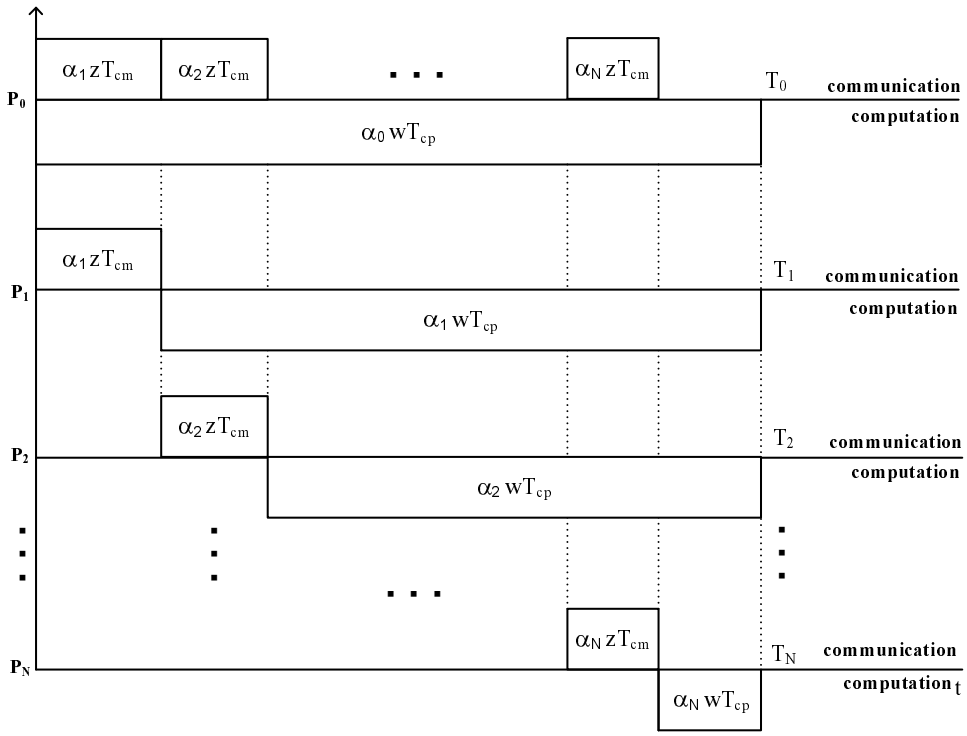


Figure 6.1: Timing diagram of N children processors with a root processor with sequential load distribution.

6.2.1 Sequential load distribution

In this subsection, we consider the case where the load distribution to the N children processors is performed in sequential fashion (i.e., *sequential distribution*). In other words, the sequence of load distribution by the root processor follows the order and let this order be $P_1, P_2, \ldots, P_N$. It is assumed that the root processor is equipped with a front-end processor, so that the root processor can compute and communicate at the same time. On the other hands, the children processors are not equipped with front-end processors, so that the children processors start computing only after they receive the whole of the processing load assigned to them (i.e., *staggered start*). The timing diagram for this distributed computing system is depicted in Fig. 6.1. In advance to the analysis of the network cost, the optimal amount of load fraction traversing the network needs to be achieved in a sense of linear dependency to the network cost. The time optimality of the load fraction is guaranteed by sustaining a minimum processing finish time. Intuitively, the minimum processing finish time can be achieved when all processors finish their own processing at the same instant

(i.e., $T_0 = T_1 = \ldots = T_N$). This is because otherwise some processors would be idle while others were still busy [1]. Regarding this condition for the optimality, it has been known on a intuitive basis that network elements should be kept constantly busy for good performance. Considering the instant of processing termination at the each processor from the Fig. 6.1, the N equations can be set as

$$\begin{aligned} T_0 &= \alpha_0 w T_{cp} \\ T_1 &= \alpha_1 z T_{cm} + \alpha_1 w T_{cp} \\ T_2 &= (\alpha_1 + \alpha_2) z T_{cm} + \alpha_2 w T_{cp} \\ &\vdots \\ T_N &= \sum_{i=1}^{N} \alpha_i z T_{cm} + \alpha_N w T_{cp} \end{aligned} \tag{6.1}$$

By the optimality condition (i.e., $T_0 = T_1 = \ldots = T_N$), the N corresponding recursive time equations can be formulated as

$$\alpha_i w T_{cp} = \alpha_{i+1}(z T_{cm} + w T_{cp}) \qquad i = 0, 1, \ldots, N-1 \tag{6.2}$$

The N recursive equations can be reformulated as

$$\alpha_i = \left(\frac{1}{\sigma + 1}\right)^i \alpha_0 \qquad i = 1, 2, \ldots, N \tag{6.3}$$

where, $\sigma = \frac{z T_{cm}}{w T_{cp}}$. Here, $\sigma < 1$ because the communication speed is generally faster than the computing speed.

To obtain $N + 1$ unknown α's, a normalization equation which states that the fractions of the total load should sum to one is applied :

$$\sum_{i=0}^{N} \alpha_i = 1 \tag{6.4}$$

By applying $N+1$ equations ((6.3) and (6.4)), one can obtain the optimal amount of load fraction, α_0 as

$$\alpha_0 = \frac{\sigma(\sigma+1)^N}{(\sigma+1)^{N+1}-1} \tag{6.5}$$

By substituting α_0 into (6.3), the other N solutions for the optimal load fraction, $\alpha_1, \alpha_2, \ldots,$ and α_N can be obtained.

Once the amount of optimal load fractions is obtained from deterministic recursive equations, the monetary cost consumed in the load communication process and the load computing process can be analyzed by adopting the static cost coefficients, c_l and c_p, communication cost coefficient and computing cost coefficient, respectively. As was mentioned in the previous section, the cost coefficients, c_l and c_p are not time varying. Thus, the monetary cost consumption can be formulated as a linear multiplication of the constant cost coefficients and the amount of load fractions. The total monetary cost demand, C_{total} for both communication via the individual links and computing in each of the processors can be expressed as a summation of individual processing costs :

$$C_{total} = c_l z T_{cm} \sum_{i=1}^{N} \alpha_i + c_p w T_{cp} \sum_{i=0}^{N} \alpha_i \tag{6.6}$$

By substituting the optimal load fraction achieved through (6.3) and (6.4) into αs in (6.6), (6.6) can be rewritten as

$$C_{total} = c_l z T_{cm} \left(\frac{(\sigma+1)^N - 1}{(\sigma+1)^{N+1} - 1} \right) + c_p w T_{cp} \tag{6.7a}$$

$$= w T_{cp} \left(c_l \sigma \left(\frac{(\sigma+1)^N - 1}{(\sigma+1)^{N+1} - 1} \right) + c_p \right) \tag{6.7b}$$

If $N \rightarrow \infty$, then the C_{total} asymptotically approaches to

$$C_{total,\infty} = wT_{cp}\left(c_l\sigma\left(\frac{1}{\sigma+1}\right)+c_p\right) = c_l z T_{cm}\left(\frac{1}{\sigma+1}\right)+c_p wT_{cp} \tag{6.8}$$

From (6.7b), by expanding the fraction having N^{th} power of sum in the numerator and $(N+1)^{th}$ power of sum in the denominator by the binomial theorem, (6.7b) can be reformulated as

$$C_{total} = wT_{cp}\left(c_l\sigma\left(\frac{\sum_{k=0}^{N-1}\binom{N}{k}\sigma^{N-k}}{\sum_{k=0}^{N}\binom{N+1}{k}\sigma^{N+1-k}}\right)+c_p\right) \tag{6.9}$$

As a special case, where $\sigma \ll 1$, the ratio of the communication speed to the computing speed is very small (i.e., the communication speed is much faster than the computing speed.), the polynomial found from the binomial expansion can be approximated by leaving out all but the largest term. Thus, (6.9) can be rewritten as a closed form :

$$C_{total} \approx wT_{cp}\left(c_l\sigma\left(\frac{N\sigma}{(N+1)\sigma}\right)+c_p\right) \approx c_l z T_{cm}\left(\frac{N}{N+1}\right)+c_p wT_{cp} \tag{6.10}$$

With an extreme condition that $\sigma \rightarrow 0$, $C_{total} = c_p wT_{cp}$ from (6.9). The cost for computing per unit load is relatively much higher than the communication cost especially when the communication speed is much faster than the computing speed. In this sense, the cost for the computing represents the larger part of the total cost, C_{total}.

By using the approximated closed form solution (6.10), the condition of the speed ratio, σ under the limitation of the monetary cost budget, K can be derived as

$$C_{total} \leq K \tag{6.11a}$$

$$wT_{cp}\left(c_l\sigma\left(\frac{N\sigma}{(N+1)\sigma}\right)+c_p\right) \leq K \tag{6.11b}$$

$$c_l \sigma \left(\frac{N}{N+1} \right) \leq \frac{K}{wT_{cp}} - c_p \tag{6.11c}$$

$$\sigma \leq \gamma \tag{6.11d}$$

where, the constant, $\gamma = \left(\frac{K}{wT_{cp}} - c_p \right) \frac{N+1}{c_l N}$. Here, the fact that right-hand side of (6.11c) is positive is guaranteed by the condition of (6.11a).

So far, the quantitative analysis with regard to total monetary cost has been performed. Now, it is an ideal point to ask the quality of the consumed monetary cost. In other words, how much efficiency in parallel processing can be achieved by consuming available monetary funds is worth being studied.

As a metric for evaluating performance of parallel computing systems, speedup, S, indicating the degree of improvement in total processing finish time as increasing the number of processors participating in the parallel computing is used. The speedup is the ratio of the total processing finish time on a single processor, $T_{f,0}$ to the total processing finish time on N+1 processors (i.e., a single root processor and N children processors). From the Fig. 6.1, the processing finish time on a single (root) processor, $T_{f,0}$ is given as

$$T_{f,0} = wT_{cp} \tag{6.12}$$

The total processing finish time on $N+1$ processors can be obtain by employing the optimality condition for the minimum processing finish time (i.e., $T_0 = T_1 = \ldots = T_N$) as

$$T_{f,N} = T_0 = \alpha_0 wT_{cp} = \left(\frac{\sigma(\sigma+1)^N}{(\sigma+1)^{N+1} - 1} \right) wT_{cp} \tag{6.13}$$

Hence, the speedup, S is given as

$$S = \frac{T_{f,0}}{T_{f,N}} = \frac{1}{\alpha_0} = \frac{(\sigma+1)^{N+1} - 1}{\sigma(\sigma+1)^N} = 1 + \frac{1}{\sigma} \left(1 - \left(\frac{1}{\sigma+1} \right)^N \right) \tag{6.14}$$

It can be seen that the speedup for $N \to \infty$ is asymptotically saturated to $S_\infty = 1 + \frac{1}{\sigma}$. As the communication speed becomes relatively faster than the computing speed (i.e., as σ decreases), the asymptotic speedup, S_∞ increases. Intuitively, this shows faster load distribution contributes to give better speedup performance in the sense of increasing the parallelism on the computing.

Now, we propose a new metric, cost efficiency, E_C. The efficiency of the parallel processing analyzed previously can be further analyzed in terms of the cost efficiency, E_C as an indicator showing how cost effectively the parallel processors can be used to process particular tasks. The cost efficiency, E_C is defined as

$$E_C = \frac{S}{C_{total}} \tag{6.15}$$

The E_C indicates the amount of improvement in the parallel processing finish time by a unit cost.

From (6.7a) and (6.14), the cost efficiency, E_C can be written as

$$\begin{aligned} E_C &= \frac{\left(\frac{1}{\sigma(\sigma+1)^N}\right)\Big((\sigma+1)^{N+1}-1\Big)^2}{c_l z T_{cm}\Big((\sigma+1)^N-1\Big) + c_p w T_{cp}\Big((\sigma+1)^{N+1}-1\Big)} \\ &= \frac{\Big((\sigma+1)-\frac{1}{(\sigma+1)^N}\Big)^2}{\sigma\left\{c_l z T_{cm}\Big(1-\frac{1}{(\sigma+1)^N}\Big) + c_p w T_{cp}\Big((\sigma+1)-\frac{1}{(\sigma+1)^N}\Big)\right\}} \end{aligned} \tag{6.16}$$

From (6.16), if $N \to \infty$, then the E_C is asymptotically saturated to

$$E_{C,\infty} = \frac{(\sigma+1)^2}{\sigma\Big(c_l z T_{cm} + c_p w T_{cp}(\sigma+1)\Big)} = \frac{S_\infty}{C_{total,\infty}} \tag{6.17}$$

Especially, for the case where $\sigma \ll 1$, (6.16) can be rewritten by using approximation of the binomial series (i.e., $(\sigma+1)^N \approx 1+N\sigma$, $(\sigma+1)^{N+1} \approx 1+(N+1)\sigma$) shown in the previous

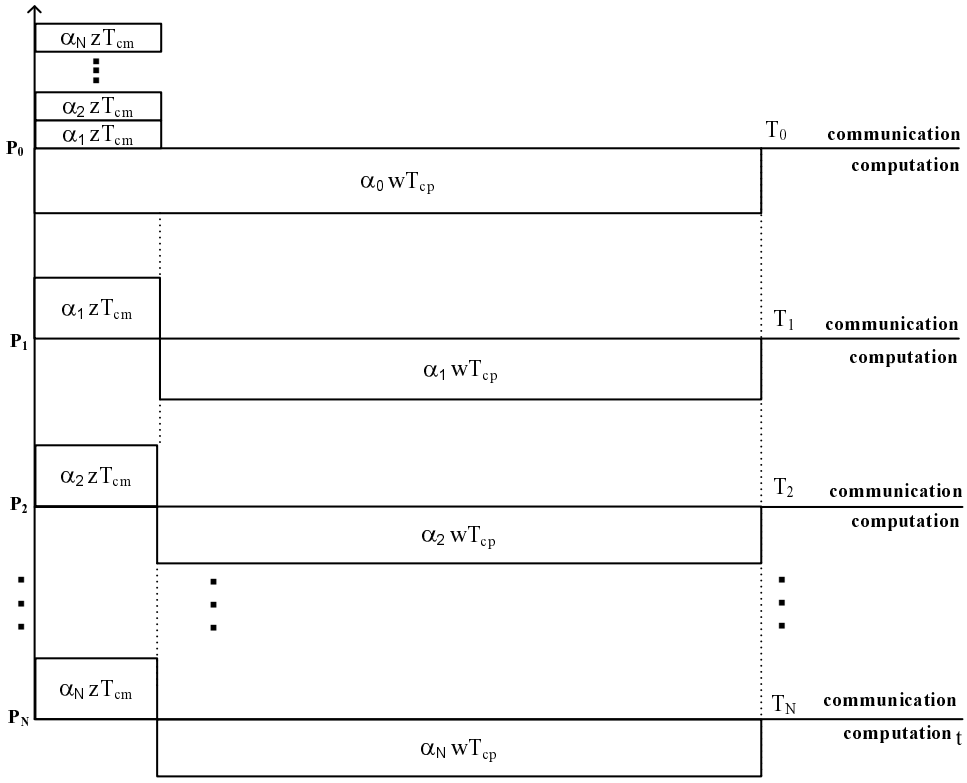


Figure 6.2: Timing diagram of N children processors with a root processor with simultaneous load distribution.

subsection :

$$E_C \approx \frac{(N+1)^2}{(N\sigma+1)\Big(c_l z T_{cm} N + c_p w T_{cp}(N+1)\Big)} \tag{6.18}$$

6.2.2 Simultaneous load distribution

Consider now the case where the root processor is able to communicate concurrently (simultaneously) with all the children processors. The simultaneous communication strategy can be implemented by utilizing multiple output buffers of the root processors for each communication link. Networking via wireless media or multi-carrier wireless networking using multiple wireless channels also makes the simultaneous communication strategy feasible [69]. The timing diagram shown in Fig. 6.2, shows the simultaneous load distribution strategy.

From the Fig. 6.2, the corresponding N equations can be set as

$$\begin{aligned} T_0 &= \alpha_0 w T_{cp} \\ T_1 &= \alpha_1 z T_{cm} + \alpha_1 w T_{cp} \\ T_2 &= \alpha_2 z T_{cm} + \alpha_2 w T_{cp} \\ &\vdots \\ T_N &= \alpha_N z T_{cm} + \alpha_N w T_{cp} \end{aligned} \tag{6.19}$$

By the optimality condition (i.e., $T_0 = T_1 = \ldots = T_N$), the N corresponding recursive time equations can be formulated as

$$\begin{aligned} \alpha_0 w T_{cp} &= \alpha_1 (z T_{cm} + w T_{cp}) \\ \alpha_i (z T_{cm} + w T_{cp}) &= \alpha_{i+1} (z T_{cm} + w T_{cp}) \qquad i = 1, 2, \ldots, N-1 \end{aligned} \tag{6.20}$$

The N recursive equations, (6.20) can be reformulated as

$$\alpha_i = \left(\frac{1}{\sigma + 1} \right) \alpha_0 \qquad i = 1, 2, \ldots, N \tag{6.21}$$

where, $\sigma = \frac{z T_{cm}}{w T_{cp}}$. Here, it can be seen that the load fractions assigned to each of the children processor are identical (i.e., $\alpha_1 = \alpha_2 = \ldots = \alpha_N$). This can be intuitively expected from the identical load distribution start time and identical computing finish time.

By using the normalization equation, (6.4) and (6.21), one can obtain the optimal amount of load fraction, α_0 as

$$\alpha_0 = \frac{\sigma + 1}{\sigma + 1 + N} \tag{6.22}$$

By substituting α_0 into (6.21), other N solutions for the optimal load fraction, $\alpha_1, \alpha_2, \ldots,$ and α_N can be obtained.

By employing (6.3), the total monetary cost can be obtained as

$$C_{total} = c_l z T_{cm} \sum_{i=1}^{N} \alpha_i + c_p w T_{cp} \sum_{i=0}^{N} \alpha_i \tag{6.23a}$$

$$= c_l z T_{cm} \left(\frac{N}{\sigma + 1 + N} \right) + c_p w T_{cp} \tag{6.23b}$$

$$= w T_{cp} \left(c_l \sigma \left(\frac{N}{\sigma + 1 + N} \right) + c_p \right) \tag{6.23c}$$

If $N \to \infty$, then the C_{total} asymptotically approaches to

$$C_{total,\infty} = w T_{cp}(c_l \sigma + c_p) = c_l z T_{cm} + c_p w T_{cp} \tag{6.24}$$

From (6.24), as compared to (6.23a), it can be seen that the factor $\sum_{i=1}^{N} \alpha_i$ in (6.23a) goes to 1. Intuitively, this phenomena can be explained in that the amount of load fractions to be transferred through links increases as the communication speed increases according to the DLT so that the amount of load fraction assigned to the root processor only for computing relatively decreases. Accordingly, the amount of sum of the load fractions (i.e., $\sum_{i=1}^{N} \alpha_i$) transmitting through the N links becomes to close to the unity. Also, the increment of the number of processors participating the parallel processing contributes to decreasing the overall amount of load fractions for each processors.

As a special case, where $\sigma \ll 1$ (when the communication speed is much faster than the computing speed), (6.23c) can be rewritten as follows

$$C_{total} = w T_{cp} \left(c_l \sigma \left(\frac{N}{1 + N} \right) + c_p \right) = c_l z T_{cm} \left(\frac{N}{1 + N} \right) + c_p w T_{cp} \tag{6.25}$$

Interestingly, (6.25) is an identical form with (6.10) in the sequential distribution case. The significantly smaller time consumption for the communication than for the computing

renders a small difference in the load communication time so that the difference in the strategies of load distribution becomes negligible.

With an extreme condition that $\sigma \rightarrow 0$, $C_{total} = c_p w T_{cp}$ from (6.23c). This is also an identical result with the sequential distribution case. In a similar intuitive sense in the case of sequential load distribution, the computing cost per unit load is relatively higher so that the computing cost is the majority part of the total cost.

From (6.23c), the condition of the speed ratio, σ under limitation of the monetary cost budget, K can be derived as

$$C_{total} \leq K \tag{6.26a}$$

$$wT_{cp}\left(c_l\sigma\left(\frac{N}{\sigma+1+N}\right)+c_p\right) \leq K \tag{6.26b}$$

$$c_l\sigma\left(\frac{N}{\sigma+1+N}\right) \leq \frac{K}{wT_{cp}} - c_p \tag{6.26c}$$

$$\frac{\sigma N}{\sigma+1+N} \leq \beta \tag{6.26d}$$

$$\sigma(N-\beta) \leq \beta(1+N) \tag{6.26e}$$

where, $\beta = \left(\frac{K}{wT_{cp}} - c_p\right)\frac{1}{c_l}$. Here, the fact that right-hand side of (6.26c) is positive is guaranteed by the condition of (6.26a). From (6.26e), according to the polarity of the factor $N-\beta$, two possible inequalities arise as below:

Case I : $N > \beta$

$$\sigma \leq \gamma \tag{6.27}$$

Case II : $N < \beta$

$$\sigma \geq \gamma \tag{6.28}$$

where, $\gamma = \frac{\beta(1+N)}{N-\beta}$. For the *Case II*, there is no restriction on the σ due to the fact that the right-hand side of (6.28), γ, is negative. Hence, the restriction on the σ exists only for the *Case I* as (6.27). Here, $N \neq \beta$. This can be clarified by using (6.23b).

To evaluate the cost efficiency, E_C, speedup, S is needed to be obtained as shown in the previous subsection. From the Fig. 6.2, the processing finish time on a single (root) processor, $T_{f,0}$ is given as

$$T_{f,0} = wT_{cp} \tag{6.29}$$

From the optimality condition, $T_0 = T_1 = \ldots = T_N$, the total processing finish time on $N+1$ processors can be obtain as

$$T_{f,N} = T_0 = \alpha_0 wT_{cp} = \left(\frac{\sigma+1}{\sigma+1+N}\right) wT_{cp} \tag{6.30}$$

Hence, the speedup, S is written as

$$S = \frac{T_{f,0}}{T_{f,N}} = 1 + \frac{N}{\sigma+1} \tag{6.31}$$

It can be seen that the speedup is scalable against N (i.e., $S_\infty = \infty$) as long as the root processor can simultaneously distribute load to the children processors. Hence, it seems reasonable to expect a better speedup performance than the case of the sequential distribution

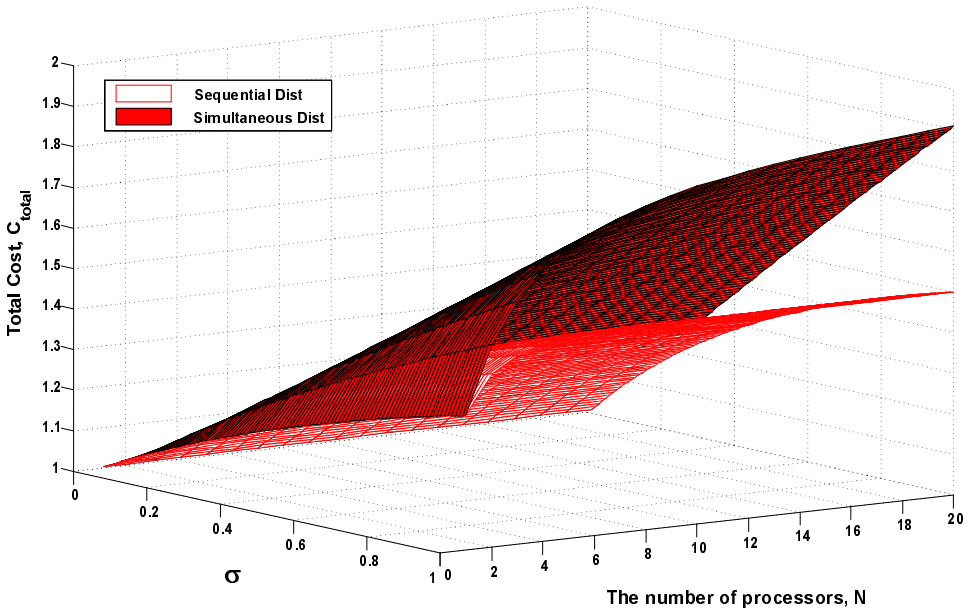


Figure 6.3: Total cost, C_{total} against the number of processors, N and speed ratio, σ.

whose asymptotic speedup is not scalable but saturated.

Now, from (6.23b) and (6.31) the cost efficiency, E_C is obtained as

$$E_C = \frac{S}{C_{total}} = \frac{1 + \frac{N}{\sigma+1}}{c_l z T_{cm}\left(\frac{N}{\sigma+1+N}\right) + c_p w T_{cp}} \tag{6.32}$$

From (6.32), it can be observed that the E_C is scalable against N which means $E_{C,\infty} \to \infty$ as $N \to \infty$:

$$E_{C,\infty} = \frac{\infty}{c_l z T_{cm} + c_p w T_{cp}} = \frac{S_\infty}{C_{total,\infty}} \tag{6.33}$$

Especially, for the case where $\sigma \ll 1$, (6.32) can be rewritten as

$$E_C \approx \frac{(N+1)^2}{c_l z T_{cm} N + c_p w T_{cp}(N+1)} \tag{6.34}$$

Comparing to (6.18) in the case of the sequential distribution, the E_C in (6.34) is improved by a factor of $N\sigma + 1$.

6.3 Cost Performance Evaluation

Based on the previous analytical result, cost performance comparisons between two load distribution strategies (sequential distribution and simultaneous distribution) are performed. All simulations are performed with specified network parameters, $w = 1$, $T_{cp} = 1$, $c_p = 1$, $c_l = 1$. According to the variable, speed ratio, σ, the value of zT_{cm} is decided. For the generality, it is assumed that the communication speed is faster than the computing speed (i.e., $wT_{cp} > zT_{cm}$). Thus, the upper bound of the variable, σ is 1.

In Fig. 6.3, total cost, C_{total} of parallel processing adopting the sequential load distribution and the simultaneous load distribution is shown. The 3D plot illustrates C_{total} as a function of two variables, the number of processors, N and speed ratio, σ. As we analyzed in the previous section, it can be seen that the C_{total} for an extremely small σ is identical in the both load distribution strategies as $c_p w T_{cp}$. Here, C_{total} changes as σ increases. As can be clearly seen, the C_{total} for the both load distribution strategies tends to increase as σ increases. The degree of the increment C_{total} in case of the simultaneous strategy is higher than the sequential distribution case. This trend can be explained by an intuitive sense from the DLT. Comparing (6.7a) and (6.23b), it can be seen that difference between the strategies in the C_{total} is caused by total amount of load to be transmitted to the children processors. When relatively slow communication speed ($\sigma \rightarrow 1$) is available in parallel processing, the root processor employing the simultaneous load distribution strategy tends to consume more communication cost in that a relatively larger amount of load is assigned to the children processors than in the sequential load distribution case. In other words, the root processor employing simultaneous load distribution keeps a relatively smaller amount of load for its own computing in order to maximizing the advantage of parallelism. On the other hand, a root processor employing sequential load distribution strategy tends keep as much load as possible to compensate for possible delay in sequential distribution. Regarding the variable N, the saturation in C_{total} can be seen as we analyzed in the previous section. The increment in C_{total} of the simultaneous distribution against the N is higher than the

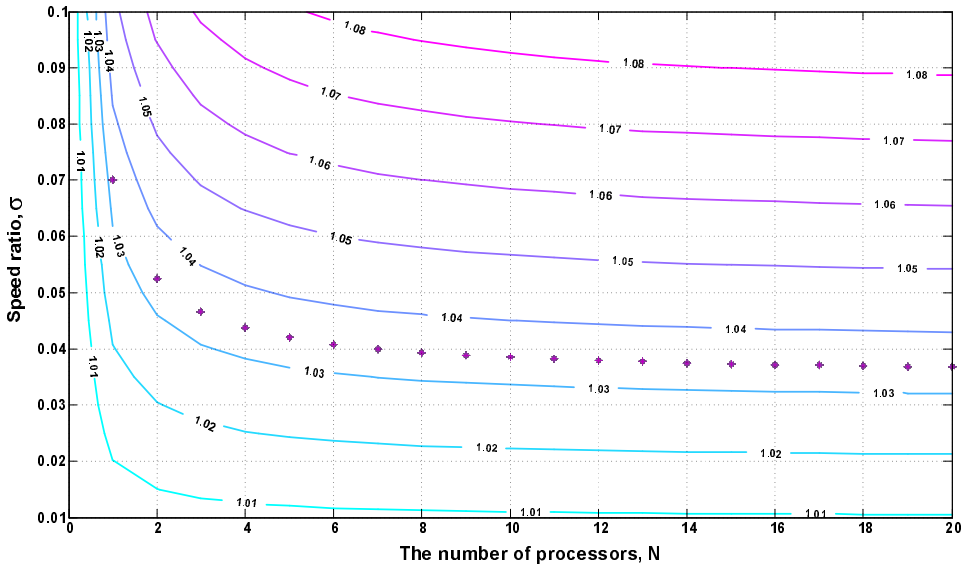


Figure 6.4: Sequential Distribution. Isocost lines with variable, the number of processors, N and speed ratio, σ.

sequential distribution case. Intuitively, this is because a relatively larger amount of load is distributed as more children processors are occupied in the case of the simultaneous load distribution case. Analytically, in the sequential distribution case (see (6.5)), $\alpha_0 \rightarrow \frac{\sigma}{\sigma+1}$ as $N \rightarrow \infty$. On the other hand, in the simultaneous distribution case (see (6.22)), $\alpha_0 \rightarrow 0$ as $N \rightarrow \infty$. In other words, the root processor employing the simultaneous load distribution strategy tends to increase the amount load to be computed in the children processors as the number of children processors increases. Hence, it can be expected that more communication cost is consumed in the simultaneous distribution case.

In Fig. 6.4, the upper bound of speed ratio, σ for the sequential load distribution case to meet a certain monetary cost budget limit analyzed previously in (6.11d) is depicted. Isomaps for isoefficiency were first developed by Drozdowski in [71]. Isocost lines which connect points having same value of the total cost are used to clarify the upper bound of the σ. Here, we set the monetary cost budget, K as 1.035 with variable the speed ratio, σ and the number of processors, N. The asterisks show the upper bound of the σ to meet the monetary cost budget, 1.035. As can be expected, the asterisks follow the isocost line for the $C_{total} = 1.035$. From the Fig. 6.4, it also can be observed that C_{total} has more dependency

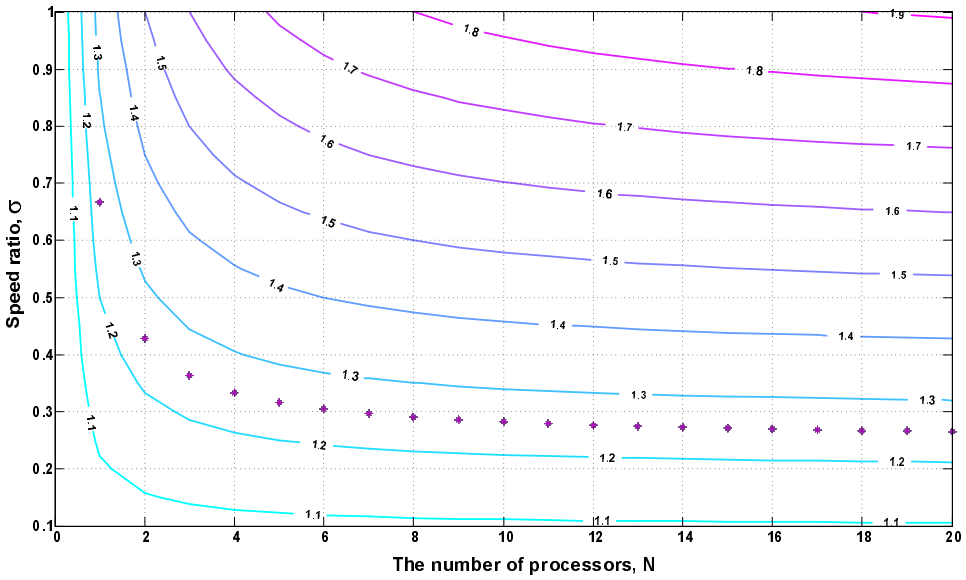


Figure 6.5: Simultaneous Distribution. Isocost lines with variable, the number of processors, N and speed ratio, σ.

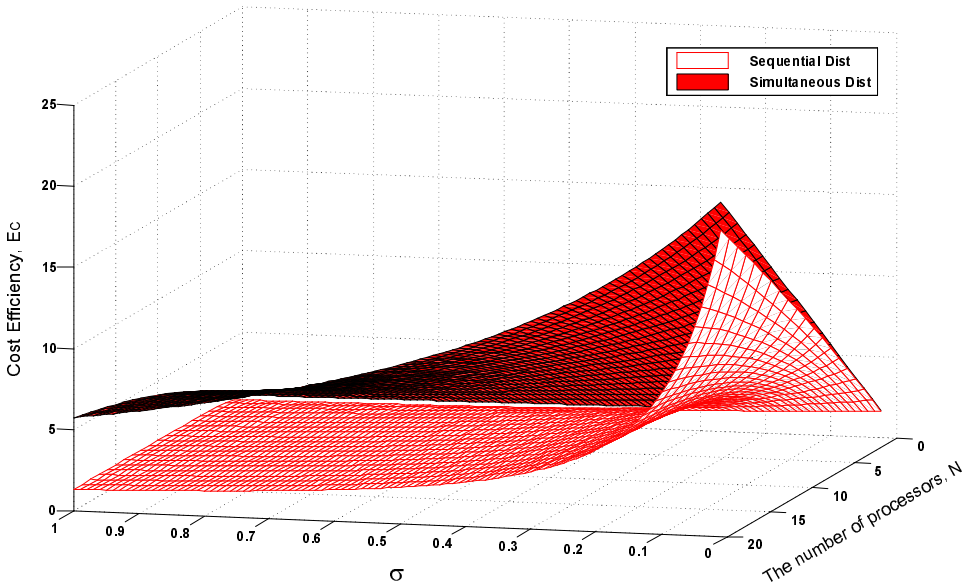


Figure 6.6: Cost efficiency, E_C against the number of processors, N and speed ratio, σ.

on the variable, σ than on N in general. This trend also can be seen from the Fig. 6.3. Likewise, isocost lines for the simultaneous load distribution case is depicted in Fig. 6.5. The simulation is based on the analytical result from (6.27). Here, we set the monetary cost budget, K as 1.25 with variable the speed ratio, σ and the number of processors, N. The value of the β defined as $\left(\frac{K}{wT_{cp}} - c_p\right)\frac{1}{c_l}$ is 0.25. Thus, all of the variable N satisfy the condition, $N > \beta$ for *Case I* so that upper bounds (asterisk points) of σ exist for all N as shown in the Fig. 6.5. Surely, it can be seen that the asterisks follow for the $C_{total} = 1.25$. Similar to the sequential distribution case shown in the Fig. 6.4, C_{total} has more dependency to the variable, σ than N in general.

In Fig. 6.6, the cost efficiency, E_C indicating achieved speedup per unit cost. As we analyzed in the previous section, the speedup, S in the simultaneous load distribution case has a scalable characteristic against the number of processors, N (see (6.31)). Hence, it can be expected that the E_C performance of the simultaneous load distribution case is better in spite of the fact that C_{total} is higher than the sequential load distribution case, but asymptotically saturated (see (6.8) and (6.24)). In the case of the simultaneous load distribution strategy, overall better performance in the cost efficiency can be observed in the Fig. 6.6. With a relatively small speed ratio, σ and the large number of processors, N, a larger improvement in the cost efficiency, E_C is observed. This phenomena implies that the speedup, S rapidly increases with relatively small σ and large N. It seems reasonable due to the fact that the total cost for the small σ is saturated regardless of N as shown in the Fig. 6.3. In an intuitive sense, this is because faster communication speed makes processors contribute more time on load computing as processors spend less time receiving the assigned load from root processor. Thus, improvement in speedup can be expected. The sharper improvement of the E_C in sequential load distribution case implicitly shows that a faster communication speed has more impact on load distribution in sequential distribution. This can be expected in that the load communication delay in the sequential load distribution increases as the number of processors waiting their own load increases. Thus, it would seem reasonable that the faster communication speed seriously contributes to release the delay caused from the

sequential distribution.

6.4 Concluding remarks

In this chapter, monetary network cost trends for sequential and simultaneous load distribution cases are investigated in a homogeneous single-level tree network. From the cost performance comparison between the two load distribution strategies, the effect of the speed of networks, including communication speed and computing speed, on the monetary network cost is investigated. Taking into consideration that monetary network cost is limited, conditions for the network speed ratio to meet the limited network cost are derived. By defining a new parameter, cost efficiency, the performance improvement in parallel processing networks can be analyzed in the sense of the cost of efficient parallel processing. The network requirements to maintain cost efficient parallel processing are studied. Cost efficiency provides explicit integrated information on relationships between monetary network cost and network performance. We believe that this study is a key to achieve cost efficient parallel process networking. This monetary network cost investigation leads to plausible challenges including monetary network cost analysis under time-varying cost coefficients in various network environments and the scalability of cost efficiency.

Chapter 7

Cost Performance Analysis in Multi-Level Tree Networks

As the size of parallel and distributed computing systems becomes larger, the efficiency in terms of monetary cost in parallel computing systems becomes of significant interest. To maintain cost efficient parallel and distributed computing network service, one needs to have insight into the network monetary cost trends against agile network environmental changes. There have been several algorithmic attempts to minimize total network cost with optimal load distribution sequences [66],[67],[68]. To investigate how effectively the monetary cost is spent for improvements in network performance, a parameter, cost efficiency is defined and examined via isocost lines [73]. The problem of monetary network cost on a homogeneous single-level tree network is considered regarding two load distribution strategies (sequential load distribution and simultaneous load distribution) [74]. Monetary network cost discussed in the previous works linearly depends on network speed parameters and the amount of work load. The time optimal amount of load to be distributed and computed in computing components in networks is obtained from *divisible load theory* (DLT) [1],[2],[3].

In this chapter, the monetary cost in multi-level tree networks is considered. Optimal load scheduling in multi-level tree networks has been taken into account in some previous studies [9],[10]. Based on the time optimal load scheduling techniques for multi-level trees,

the closed form solution for optimal monetary network cost is analyzed. Through analytical investigation of network cost trends, tractable relationship between monetary network cost and optimal processing finish time are achieved.

The rest of the chapter is organized as follows. Section 7.1 discusses the network model and notations used in this chapter. In section 7.2, the mathematical analysis of monetary network cost of multi-level tree networks is conducted based on the DLT. Cost performance and evaluation results for a multi-level tree analyzed in this chapter are presented in Section 7.3. Finally, this chapter concludes with some possible extensions in Section 7.4.

7.1 Problem Formulation and Preliminary Remarks

The distributed computing network model to be considered in this chapter is a multi-level tree network (bus network) with homogeneous speed parameters and cost coefficients. Each subtree network consists of children processors connected to a parent processor via direct communication links. The parent (root) processor has role as a load distributor while also computing itself. The children processors are involved in computing their own load assigned from the root processor via the direct links. Monetary "cost" involving processors' computing and communicating linearly depends on the speed of processors and links and the amount of work load.

The following notation is used in this chapter.

α_i : The load fraction assigned to the ith children processor (where $i = 1, 2, \ldots, N$). Index $i = 0$ is for the root processor.

w : The inverse computing speed of a processor. [sec/load].

z : The inverse communication speed of a link. [sec/load].

T_{cp} : Computing intensity constant. [Dimensionless]. The entire load (amount of unity) can be processed on a processor in time wT_{cp} sec.

T_{cm} : Communication intensity constant. [Dimensionless]. The entire load (amount of unity) can be transmitted over a link in time zT_{cm} sec.

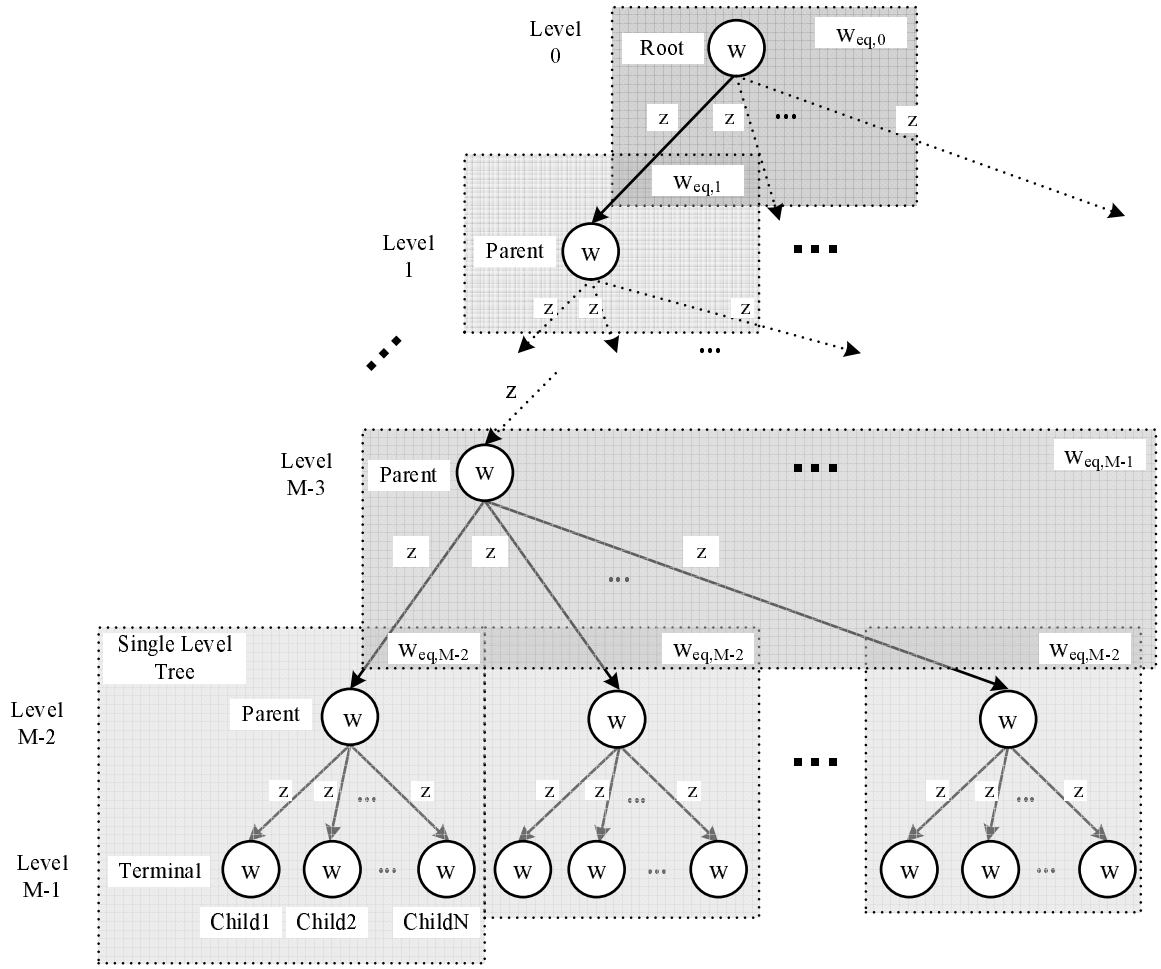


Figure 7.1: M level tree with root (parent) processors with N children processors.

c_p : Static computing cost coefficient. [cost/sec]. The computing cost for the entire load (amount of unity) is $c_p w T_{cp}$ cost.

c_l : Static communication cost coefficient. [cost/sec]. The communication cost for the entire load (amount of unity) is $c_l z T_{cm}$ cost.

T_f^N : The total processing finish time. Time at which a single root processor and N children processors complete their computation. Note that $T_f^{N,M}$ is for M level tree network.

C_{total}^N : The total monetary network cost of a single level tree network with a single root processor and N children processors. Note that $C_{total}^{N,M}$ is for M level tree network.

T_i : The total time that elapses between the beginning of the process at $t = 0$ and the time when i^{th} processor completes its own computation (where $i = 0, 1, \ldots, N$).

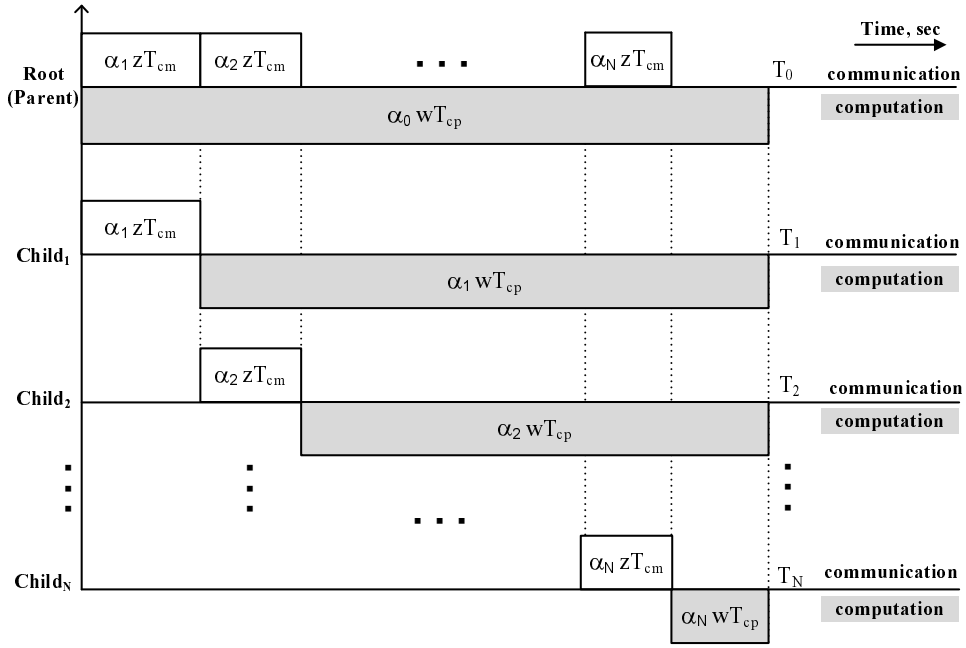


Figure 7.2: Timing diagram of N children processors with a root processor with sequential load distribution.

7.2 Cost analysis of homogeneous tree networks

7.2.1 Single Level Tree

Consider a single level tree network with N children processors (terminal processors at the bottom level $M-1$) interconnected through N links to a root (which is parent processor at level $M-2$, see Fig. 7.1). The root processor distributes loads among the N children processors through the N direct links. We consider the case where the load distribution to the N children processors is performed in sequential fashion (i.e., *sequential distribution*). It is assumed that the root processor is equipped with front-end processor, so that the root processor can compute and communicate at the same time. On the other hand, the children processors are not equipped with front-end processors, so that the children processors start computing only after they receive the whole of the processing load assigned to them (i.e., *staggered start*). The timing diagram for this single level tree distributed computing system is depicted in Fig. 7.2. In advance of the analysis of the network cost, the optimal amount of load traversing the network needs to be found. The optimality of the load fraction

is guaranteed by sustaining a minimum processing finish time. Intuitively, the minimum processing finish time can be achieved when all processors finish their own processing at the same instant (i.e., $T_0 = T_1 = \ldots = T_N$). This is because otherwise some processors would be idle while others were still busy [1]. Regarding this condition for the optimality, it has been known on a intuitive basis that network elements should be kept constantly busy for good performance.

Considering the instant of processing termination at the each processor composing the single level tree from the Fig. 7.2, the N equations can be set as

$$\begin{aligned} T_0 &= \alpha_0 w T_{cp} \\ T_1 &= \alpha_1 z T_{cm} + \alpha_1 w T_{cp} \\ T_2 &= (\alpha_1 + \alpha_2) z T_{cm} + \alpha_2 w T_{cp} \\ &\vdots \\ T_N &= \sum_{i=1}^{N} \alpha_i z T_{cm} + \alpha_N w T_{cp} \end{aligned} \tag{7.1}$$

By the optimality condition (i.e., $T_0 = T_1 = \ldots = T_N$), the N corresponding recursive time equations can be formulated as

$$\alpha_i w T_{cp} = \alpha_{i+1}(z T_{cm} + w T_{cp}) \qquad i = 0, 1, \ldots, N-1 \tag{7.2}$$

The N recursive equations can be reformulated as

$$\alpha_i = q^i \alpha_0 \qquad i = 1, 2, \ldots, N \tag{7.3}$$

where, $q = \frac{w T_{cp}}{z T_{cm} + w T_{cp}}$, $0 < q < 1$.

To obtain $N + 1$ unknown $\alpha_{0,1,\ldots,N}$, an additional normalization equation which states that

the fractions of the total load should sum to one is applied :

$$\sum_{i=0}^{N} \alpha_i = 1 \tag{7.4}$$

By applying $N+1$ equations ((7.3) and (7.4)), one can obtain the optimal amount of load fraction, α_0 as

$$\alpha_0 = \frac{1-q}{1-q^{N+1}} \tag{7.5}$$

By substituting α_0 into (7.3), the other N solutions for the optimal load fractions, $\alpha_{1,2,\ldots,N}$ can be obtained.

From (7.5), the total minimum processing finish time with N children processors can be obtained based on the optimality condition ($T_0 = T_1 = \ldots = T_N$)

$$T_f^N = T_0 = \alpha_0 w T_{cp} = \left(\frac{1-q}{1-q^{N+1}}\right) w T_{cp} \tag{7.6}$$

With the information about the amount of optimal load fractions, the monetary cost consumed for the communication process and load computing process can be analyzed by employing the time invariant cost coefficients, c_l and c_p, communication cost coefficient and computing cost coefficient, respectively. The monetary cost model can be formulated as a linear multiplication of the cost coefficients and time duration regarding load processing . The total monetary cost demand, C_{total}^N for both communication via the individual links and computing in each of the N children processors and a root processor can be expressed as a summation of individual processing costs :

$$C_{total}^N = c_l z T_{cm} \sum_{i=1}^{N} \alpha_i + c_p w T_{cp} \sum_{i=0}^{N} \alpha_i \tag{7.7}$$

From (7.7), using (7.3), (7.4), and (7.5), the total cost consumption at the single level tree can be rewritten as

$$C_{total}^{N} = c_l z T_{cm} \left(\frac{q - q^{N+1}}{1 - q^{N+1}} \right) + c_p w T_{cp} \tag{7.8}$$

At this moment, it seems natural to ask if a trade-off exists between the total cost and the processing finish time according to the change of the number of children processors involving parallel processing, N. As a metric to indicate the trend of the trade-off, let ρ_n be the ratio the amount of variation in total processing finish time to the amount of variation in the total cost as the number of children processor is shifted by n :

$$\rho_n = \frac{\Delta T_f^n}{\Delta C_{total}^n} = \frac{|T_f^N - T_f^{N-n}|}{C_{total}^N - C_{total}^{N-n}} \tag{7.9}$$

Here, the total minimum processing finish time with $N-n$ (where $n = 1, 2, \ldots, N$) children processors can be given as a function of T_f^N :

$$\begin{aligned}
T_f^{N-1} &= \left(\frac{1 - q^{N+1}}{1 - q^N} \right) T_f^N \\
T_f^{N-2} = \left(\frac{1 - q^{N+1}}{1 - q^{N-1}} \right) T_f^N &= \left(\frac{1 - q^N}{1 - q^{N-1}} \right) T_f^{N-1} \\
&\vdots \\
T_f^{N-n} = \left(\frac{1 - q^{N+1}}{1 - q^{N-n+1}} \right) T_f^N &= \left(\frac{1 - q^{N-n+2}}{1 - q^{N-n+1}} \right) T_f^{N-n+1} \\
n &= 0, 1, \ldots, N-1
\end{aligned} \tag{7.10}$$

Similarly, total cost consumption in which $N-n$ children processors are involved can be obtained as

$$\begin{aligned} C_{total}^{N-1} &= c_l z T_{cm}\left(\frac{q-q^N}{1-q^N}\right) + c_p w T_{cp} \\ C_{total}^{N-2} &= c_l z T_{cm}\left(\frac{q-q^{N-1}}{1-q^{N-1}}\right) + c_p w T_{cp} \\ &\vdots \\ C_{total}^{N-n} &= c_l z T_{cm}\left(\frac{q-q^{N-n+1}}{1-q^{N-n+1}}\right) + c_p w T_{cp} \\ n &= 0, 1, \ldots, N-1 \end{aligned} \tag{7.11}$$

From (7.10), we can reformulate the equations for total monetary cost for networks involving $N-n$ children processors as

$$\begin{aligned} C_{total}^{N-1} &= \beta\left(\frac{T_f^{N-1}}{T_f^{N-2}}\right) + c_p w T_{cp} \\ C_{total}^{N-2} &= \beta\left(\frac{T_f^{N-2}}{T_f^{N-3}}\right) + c_p w T_{cp} \\ &\vdots \\ C_{total}^{N-n} &= \beta\left(\frac{T_f^{N-n}}{T_f^{N-n-1}}\right) + c_p w T_{cp} \\ n &= 0, 1, \ldots, N-1 \end{aligned} \tag{7.12}$$

where, $\beta = q c_l z T_{cm}$.

Using (7.10) and (7.12), (7.9) can be rewritten as

$$\rho_n = \frac{|T_f^N - T_f^{N-n}|}{\beta\left(\frac{T_f^N}{T_f^{N-1}} - \frac{T_f^{N-n}}{T_f^{N-n-1}}\right)} \tag{7.13}$$

From (7.10), (7.13) can be simplified as

$$\begin{aligned}
\rho_n &= \frac{\left(|1-\frac{1-q^{N+1}}{1-q^{N-n+1}}|\right)T_f^N}{\beta\left(\frac{1-q^N}{1-q^{N+1}}-\frac{1-q^{N-n}}{1-q^{N-n+1}}\right)} \\
&= \frac{\left(|1-\frac{1-q^{N+1}}{1-q^{N-n+1}}|\right)\left(\frac{1-q}{1-q^{N+1}}\right)wT_{cp}}{\beta\left(\frac{1-q^N}{1-q^{N+1}}-\frac{1-q^{N-n}}{1-q^{N-n+1}}\right)} \\
&= \frac{wT_{cp}}{\beta}\left(\frac{(|q^{N+1}-q^{N-n+1}|)(1-q)}{(1-q^N)(1-q^{N-n+1})-(1-q^{N-n})(1-q^{N+1})}\right) \\
&= \frac{wT_{cp}}{\beta}\left(\frac{q^{N-n+1}-q^{N+1}}{q^{N-n}-q^N}\right) = \frac{wT_{cp}}{\beta}q \\
&= \frac{1}{\sigma c_l}
\end{aligned} \tag{7.14}$$

where, $\sigma = \frac{zT_{cm}}{wT_{cp}}$. In a common sense, $\sigma < 1$ because the communication speed is generally faster than the computing speed. From (7.14), the ratio ρ_n inversely commensurate with the amount of σ. In other words, ρ_n increases as the link speed becomes faster than the computing speed (i.e., $\sigma \to 0$).

7.2.2 Multilevel Tree

As depicted in Fig. 7.1, multi-level tree (M level tree) is composed of a single root processor at level 0, intermediate parent processors with their own N children processors (subtrees). The intermediate parent processors can be considered as children processors of upper level parent processors. We call the children processors at the bottom level $M-1$ terminal processors. The M level tree with N children processors has total N^M terminal processors. As we mentioned in the previous section, the load distribution in a subtree follows sequential distribution and staggered start. The load distribution among inter-levels employs virtual cut through switching techniques. The virtual cut through switching technique allows relay of load distribution among inter-levels without store and forward delays. Analysis of multi-level tree for closed form solution for the optimal amount of load fraction is feasible by

collapsing a subtree into an equivalent processor [7]. The equivalent processor preserves the same characteristics of the original tree in senses of minimum processing finish time and total cost.

By equating the processing finish time on a single equivalent processor and the minimum processing finish time on the single level tree (a subtree) analyzed in the previous section (see (7.6))

$$1 \cdot w_{eq} T_{cp} = T_f^N \tag{7.15}$$

we can obtain inverse computing speed of the equivalent processor replacing the single level tree as

$$w_{eq} = \alpha_0 w = \left(\frac{1-q}{1-q^{N+1}} \right) w \tag{7.16}$$

On the way of collapsing single level tree into an equivalent processor, we can also obtain an expression for $c_{p,eq}$. Here, $c_{p,eq}$ is a constant that is computing cost coefficient of an equivalent processor that aggregates a subtree. By equating the total cost on a single equivalent processor with w_{eq} and (7.8)

$$1 \cdot c_{p,eq} w_{eq} T_{cp} = C_{total}^N \tag{7.17}$$

From (7.17), using (7.16) and (7.8), one can find $c_{p,eq}$ as

$$c_{p,eq} = c_l \frac{z T_{cm}}{w T_{cp}} \left(\frac{q - q^{N+1}}{1-q} \right) + c_p \left(\frac{1 - q^{N+1}}{1-q} \right) \tag{7.18}$$

Likewise, staring at the bottom level (level $M-1$), subtrees can be collapsed into equivalent processors with equivalent inverse computing speed, w_{eq} and with equivalent computing cost coefficient, $c_{p,eq}$. For example, if one collapses the entire M level tree depicted in Fig. 7.1 into an $M-1$ level tree, then the terminal processors in the collapsed $M-1$ level tree

will be N^{M-1} equivalent processors. This collapsing procedure will continue until the root processor at level 0 and its children processors (which will be N equivalent processors) are replaced by a single equivalent processor.

The timing diagram for each subtree with (equivalent) children processors is identical with Fig. 7.2 except for the fact that the inverse computing speed of the N children processors is collapsed into $w^M_{eq,k}$. Here, k indicates the level of (equivalent) children processors being considered (i.e., $k = 1, 2, \ldots, M-2$). The superscript M denotes equivalent processing speed regarding M level tree. Note that in our case, $w^M_{eq,M-2} = w_{eq}$, where w_{eq} is given as (7.16). Thus, similarly, we can formulate N corresponding equations from a subtree with a parent processor with w at the level $k-1$ and N children processors with $w^M_{eq,k}$ at the bottom level k.

$$\begin{aligned} T_0 &= \alpha_0 w T_{cp} \\ T_1 &= \alpha_1 z T_{cm} + \alpha_1 w^M_{eq,k} T_{cp} \\ T_2 &= (\alpha_1 + \alpha_2) z T_{cm} + \alpha_2 w^M_{eq,k} T_{cp} \\ &\vdots \\ T_N &= \sum_{i=1}^{N} \alpha_i z T_{cm} + \alpha_N w^M_{eq,k} T_{cp} \end{aligned} \tag{7.19}$$

Note that we use same symbol α for the fraction of load as we used in analyzing the single level tree. Surely, each α for different k has different value.

By the optimality condition (i.e., $T_0 = T_1 = \ldots = T_N$), the N corresponding recursive time

equations can be formulated as

$$\begin{aligned}
\alpha_1 &= r_k \alpha_0 \\
\alpha_2 &= q_k \alpha_1 \\
\alpha_3 &= q_k \alpha_2 \\
&\vdots \\
\alpha_N &= q_k \alpha_{N-1}
\end{aligned} \tag{7.20}$$

The above N equations can be rewritten as

$$\begin{aligned}
\alpha_1 &= r_k \alpha_0 \\
\alpha_i &= q_k^{i-1} \alpha_1 \qquad i = 2, 3, \ldots, N
\end{aligned} \tag{7.21}$$

where $r_k = \frac{wT_{cp}}{zT_{cm} + w_{eq,k}^M T_{cp}}$ and $q_k = \frac{w_{eq,k}^M T_{cp}}{zT_{cm} + w_{eq,k}^M T_{cp}}$.
Using the normalization equation (7.4), α_0 can be found as

$$\alpha_0 = \frac{1}{1 + r_k \left(\sum_{i=0}^{N-1} q_k^i \right)} \tag{7.22}$$

By substituting α_0 into (7.21), N solutions for the optimal fraction of load can be achieved as

$$\alpha_i = q_k^{i-1} r_k \left(\frac{1}{1 + r_k \left(\sum_{i=0}^{N-1} q_k^i \right)} \right) \qquad i = 1, 2, \ldots, N \tag{7.23}$$

Following a similar step shown in the previous section, by equating the processing finish time on a single equivalent processor with $w_{eq,k-1}^M$ to the optimal finish time of a subtree at

the level k, $\alpha_0 wT_{cp}$, one has :

$$1 \cdot w^M_{eq,k-1} T_{cp} = \alpha_0 wT_{cp} = \left(\frac{1}{1 + r_k \left(\sum_{i=0}^{N-1} q_k^i \right)} \right) wT_{cp} \tag{7.24}$$

Now, the inverse computing speed of the equivalent processor (which replaces the subtree with children processors with $w^M_{eq,k}$) at level $k-1$ can be obtained as follows:

$$w^M_{eq,k-1} = \left(\frac{1}{1 + r_k \left(\sum_{i=0}^{N-1} q_k^i \right)} \right) w \tag{7.25}$$

This collapsing process will continue until the root processor at level 0 and its children processors with $w^M_{eq,1}$ are collapsed to a single equivalent processor with $w^M_{eq,0}$. Thus, the total processing finish time on the whole M level tree in which each subtree has its N children processors can be written as follows:

$$T_f^{N,M} = 1 \cdot w^M_{eq,0} T_{cp} \tag{7.26}$$

At this moment, it seems nature to ask how much speedup improvement can be achieved by adding a level of tree. To analyze $w^M_{eq,0}$ is inverse computing speed of an equivalent processor that replaces the whole M level tree. From (7.25), $w^M_{eq,0}$ can be written as

$$\begin{aligned} w^M_{eq,0} &= \left(\frac{1}{1 + r_1 \left(\sum_{i=0}^{N-1} q_1^i \right)} \right) w = \left(\frac{1}{1 + r_1 \left(\frac{1-q_1^N}{1-q_1} \right)} \right) w \\ &= \frac{w}{1 + \left(\frac{1-q_1^N}{\sigma} \right)} \end{aligned} \tag{7.27}$$

Here, $q_1 = \frac{w^M_{eq,1} T_{cp}}{zT_{cm} + w^M_{eq,1} T_{cp}}$ and $\sigma = \frac{zT_{cm}}{wT_{cp}}$.

Based on the pattern of tree collapsing sequence, following equality condition can be ob-

tained:

$$w_{eq,1}^{k} = w_{eq,0}^{k-1} \qquad k = 3, 4, \ldots, M \tag{7.28}$$

Applying (7.28), q_1 can be rewritten as $\frac{w_{eq,0}^{M-1} T_{cp}}{zT_{cm} + w_{eq,0}^{M-1} T_{cp}}$.
Assuming that $\frac{zT_{cm}}{w_{eq,0}^{M-1} T_{cp}} \ll 1$, $q_1^N \approx \frac{1}{1 + N \frac{zT_{cm}}{w_{eq,0}^{M-1} T_{cp}}}$. Thus, (7.27) can be reformulated as a function of $w_{eq,0}^{M-1}$ as :

$$\begin{aligned} w_{eq,0}^{M} &= \left(\frac{\sigma(w_{eq,0}^{M-1} T_{cp} + NzT_{cm})}{\sigma w_{eq,0}^{M-1} T_{cp} + (\sigma + 1) NzT_{cm}} \right) w = \left(\frac{w_{eq,0}^{M-1} T_{cp} + NzT_{cm}}{w_{eq,0}^{M-1} T_{cp} + N(zT_{cm} + wT_{cp})} \right) w \\ &= \left(1 - \frac{NwT_{cp}}{w_{eq,0}^{M-1} T_{cp} + N(zT_{cm} + wT_{cp})} \right) w \end{aligned} \tag{7.29}$$

Here, (7.29) gives us an general idea that an equivalent inverse computing speed aggregating whole M level tree, $w_{eq,0}^{M}$ can be calculated from information of an equivalent inverse computing speed aggregating whole $M-1$ level tree, $w_{eq,0}^{M-1}$ based on the assumption, $\frac{zT_{cm}}{w_{eq,0}^{M-1} T_{cp}} \ll 1$. In a recursive manner, $w_{eq,0}^{M}$ can be obtained from $w_{eq,0}^{M-m}$ where $m = 1, 2, \ldots, M-1$.
From (7.7), the total cost consumption in a subtree with a parent processors at level $k-1$ with w and N children processors at level k with $w_{eq,k}^{M}$ can be obtained as

$$\begin{aligned} C_{total,N}^{k} &= c_l zT_{cm} \sum_{i=1}^{N} \alpha_i + c_p \alpha_0 wT_{cp} + c_{p,eq,k}^{M} w_{eq,k}^{M} T_{cp} \sum_{i=1}^{N} \alpha_i \\ &= (c_l zT_{cm} + c_{p,eq,k}^{M} w_{eq,k}^{M} T_{cp}) \sum_{i=1}^{N} \alpha_i + c_p \alpha_0 wT_{cp} \\ &= (c_l zT_{cm} + c_{p,eq,k}^{M} w_{eq,k}^{M} T_{cp}) r_k \alpha_0 \Big(\sum_{i=0}^{N-1} q_k^i \Big) + c_p \alpha_0 wT_{cp} \end{aligned} \tag{7.30}$$

Here, $c_{p,eq,k}^{M}$ indicates equivalent computing cost efficient at level k (where, $k = 1, 2, \ldots, M-2$). The superscript M denotes equivalent computing cost coefficient regarding M level tree. Uniquely, $c_{p,eq,M-2}^{M} = c_{p,eq}$, where $c_{p,eq}$ is (7.18).

Using (7.24), (7.30) can be rewritten as

$$C^k_{total,N} = (c_l z T_{cm} + c^M_{p,eq,k} w^M_{eq,k} T_{cp}) r_k \frac{w^M_{eq,k-1}}{w} \Big(\sum_{i=0}^{N-1} q^i_k \Big) + c_p w^M_{eq,k-1} T_{cp}$$
$$= \Big(\Big(\frac{c_l \sigma + c^M_{p,eq,k} \beta_k}{\sigma + \beta_k} \Big) \Big(\sum_{i=0}^{N-1} q^i_k \Big) + c_p \Big) w^M_{eq,k-1} T_{cp} \tag{7.31}$$

where $\sigma = \frac{z T_{cm}}{w T_{cp}}$ and $\beta_k = \frac{w^M_{eq,k}}{w}$.

By equating the total cost on a single equivalent processor at level $k-1$ and $C^k_{total,N}$, we can obtain

$$1 \cdot c^M_{p,eq,k-1} w^M_{eq,k-1} T_{cp} = C^k_{total,N} \tag{7.32}$$

From (7.32), using (7.31), the equivalent computing cost coefficient of the equivalent children processors at level $k-1$ can be found as

$$c^M_{p,eq,k-1} = \Big(\frac{c_l \sigma + c^M_{p,eq,k} \beta_k}{\sigma + \beta_k} \Big) \Big(\sum_{i=0}^{N-1} q^i_k \Big) + c_p = \Big(\frac{c_l \sigma + c^M_{p,eq,k} \beta_k}{\sigma + \beta_k} \Big) \Big(\frac{1 - q^N_k}{1 - q_k} \Big) + c_p \tag{7.33}$$

Here, q_k can be rewritten as $\frac{\beta_k}{\sigma + \beta_k}$. Assuming that that $\frac{\sigma}{\beta_k} \ll 1$ (i.e, $\frac{z T_{cm}}{w^M_{eq,k} T_{cp}} \ll 1$), $q^N_k \approx \frac{1}{1 + N \frac{\sigma}{\beta_k}}$. Thus, (7.33) can be rewritten as

$$c^M_{p,eq,k-1} = \Big(\frac{c_l \gamma_k + c^M_{p,eq,k}}{\gamma_k + 1} \Big) \Big(\frac{\gamma_k + 1}{\gamma_k + \frac{1}{N}} \Big) + c_p \tag{7.34}$$

where, $\gamma_k = \frac{\sigma}{\beta_k}$.

The total cost on the whole M level tree in which each subtree has N children processors can be written as follows:

$$C^M_{total,N} = 1 \cdot c^M_{p,eq,0} w^M_{eq,0} T_{cp} \tag{7.35}$$

From (7.34), $c_{p,eq,0}^{M}$ is given as

$$c_{p,eq,0}^{M} = \left(\frac{c_l\gamma_1 + c_{p,eq,1}^{M}}{\gamma_1 + 1}\right)\left(\frac{\gamma_1 + 1}{\gamma_1 + \frac{1}{N}}\right) + c_p \tag{7.36}$$

Using a similar intuition employed in (7.28), following equality condition can be also obtained:

$$c_{p,eq,1}^{k} = c_{p,eq,0}^{k-1} \qquad k = 3, 4, \ldots, M \tag{7.37}$$

By applying (7.37), (7.36) can be reformulated as

$$c_{p,eq,0}^{M} = \left(\frac{c_l\gamma_1 + c_{p,eq,0}^{M-1}}{\gamma_1 + 1}\right)\left(\frac{\gamma_1 + 1}{\gamma_1 + \frac{1}{N}}\right) + c_p \tag{7.38}$$

Here, $\gamma_1 = \frac{\sigma}{\beta_1} = \frac{zT_{cm}}{w_{eq,1}^{M}T_{cp}} = \frac{zT_{cm}}{w_{eq,0}^{M-1}T_{cp}}$ from (7.28). The equivalent computing cost coefficient replacing the whole M level tree, $c_{p,eq,0}^{M}$ can be calculated from information of equivalent computing speed, $w_{eq,0}^{M-1}$ and equivalent computing cost coefficient, $c_{p,eq,0}^{M-1}$ of $M-1$ level tree based on the assumption, $\frac{zT_{cm}}{w_{eq,k}^{M}T_{cp}} \ll 1$. In a recursive manner, $c_{p,eq,0}^{M}$ can be obtained from $c_{p,eq,0}^{M-m}$ where $m = 1, 2, \ldots, M-1$.

At this moment, it seems nature to ask trade-off between total cost and processing finish time according to the change of the entire level of tree, M.

The total cost decrement according to the decrease in the number of levels of the tree by m (from level M to level $M-m$) is given as

$$\Delta C_{total,N}^{M} = C_{total,N}^{M} - C_{total,N}^{M-m} = c_{p,eq,0}^{M} w_{eq,0}^{M} T_{cp} - c_{p,eq,0}^{M-m} w_{eq,0}^{M-m} T_{cp} \tag{7.39}$$

Likewise, the variation in the minimum processing finish time can be expected as the level of tree is changed. It seems reasonable that longer processing time is needed to terminate entire processing as the level of tree involving in parallel processing decreases. The increment in the total processing finish time as decreasing the level of tree by m (from level M to level

$M-m$) is given as

$$\Delta T_f^{N,M} = |T_f^{N,M} - T_f^{N,M-m}| = |w_{eq,0}^{M} T_{cp} - w_{eq,0}^{M-m} T_{cp}| \tag{7.40}$$

Now, let ρ_m be the ratio the amount of increase in total processing finish time to the amount of decrease in the total cost. ρ_m shows trend in the trade-off in total cost and processing finish time.

$$\begin{aligned} \rho_m &= \frac{\Delta T_f^{N,M}}{\Delta C_{total,N}^{M}} = \frac{|T_f^{N,M} - T_f^{N,M-m}|}{C_{total,N}^{M} - C_{total,N}^{M-m}} \\ &= \frac{|w_{eq,0}^{M} T_{cp} - w_{eq,0}^{M-m} T_{cp}|}{c_{p,eq,0}^{M} w_{eq,0}^{M} T_{cp} - c_{p,eq,0}^{M-m} w_{eq,0}^{M-m} T_{cp}} \end{aligned} \tag{7.41}$$

As reminding that $w_{eq,0}^{M}$ and $c_{p,eq,0}^{M}$ can be expressed as a function of $w_{eq,0}^{M-m}$ and $c_{p,eq,0}^{M-m}$ respectively in a recursive way (see (7.29) and (7.38)), (7.41) can be reformulated as a function of variables, $w_{eq,0}^{M-m}$ and $c_{p,eq,0}^{M-m}$ at the $M-m$ level tree as

$$\rho_m = \frac{|f(w_{eq,0}^{M-m}) - w_{eq,0}^{M-m}|}{g(w_{eq,0}^{M-m}, c_{p,eq,0}^{M-m}) f(w_{eq,0}^{M-m}) - c_{p,eq,0}^{M-m} w_{eq,0}^{M-m}} \tag{7.42}$$

Here, $f(w_{eq,0}^{M-m})$ and $g(w_{eq,0}^{M-m}, c_{p,eq,0}^{M-m})$ denote recursive functions of (7.29) and (7.38) respectively. In a similar manner, ρ_m can be express as a function of variables $w_{eq,0}^{M}$ and $c_{p,eq,0}^{M}$ at the M level tree. The derivation regarding this is not considered due to the similarity.

The speedup, S indicating the degree of improvement in total processing finish time as employing parallel computing can be formulated as the ratio of the total processing finish time on a single processor (root processor) to the total processing finish time on whole M level tree. Thus, the speedup, S_N^M can be given as

$$S_N^M = \frac{T_f^{0,0}}{T_f^{N,M}} = \frac{1 \cdot wT_{cp}}{1 \cdot w_{eq,0}^{M} T_{cp}} = \frac{w}{w_{eq,0}^{M}} \tag{7.43}$$

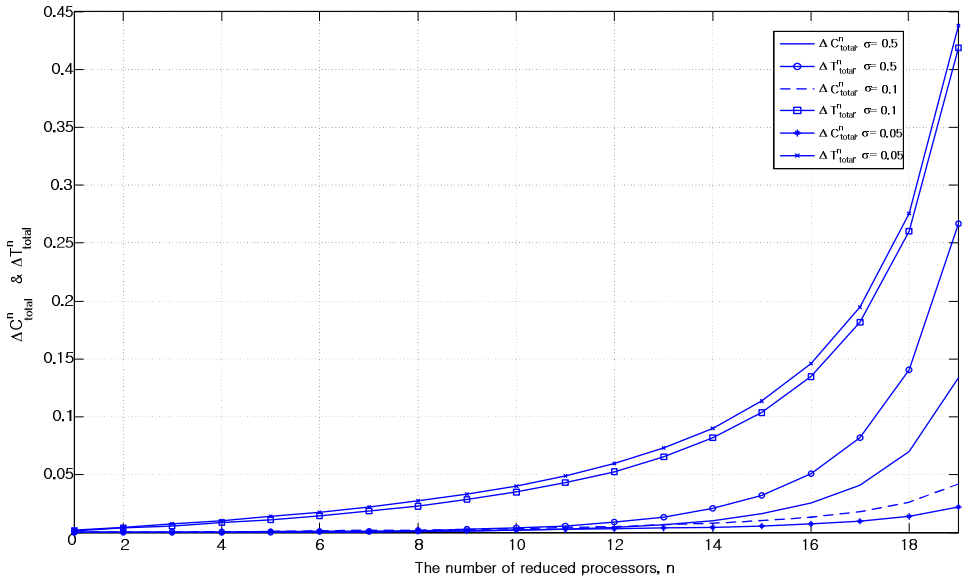


Figure 7.3: ΔT_f^n and ΔC_{total}^n vs. n with $\sigma = 0.5, 0.1$, and 0.05.

Cost efficiency, E_C, indicating how cost effectively the parallel processors can be used to process particular tasks, defined in [74] as

$$E_C = \frac{S}{C_{total}} = \frac{S_N^M}{C_{total,N}^M} \tag{7.44}$$

can be obtained by using (7.35) and (7.43).

7.3 Cost Performance Evaluation

In a homogeneous single level tree, the trend of the amount of variation in total processing finish time, ΔT_f^n and the amount of variation in the total monetary cost, ΔC_{total}^n as the number of children processors is reduced by n is depicted in Fig. 7.3. Simulation is performed with network parameters, $w = 1$, $T_{cp} = 1$, $T_{cm} = 1$, $c_p = 1$, and $c_l = 1$ according to three different values of $z = 0.5, 0.1$, and 0.05. Thus, three different speed ratios $\sigma = 0.5, 0.1$, and 0.05 can be reasoned. Initially the tree has 20 children processors. The number of children processors is reduced by 1 until only 1 child processor is left. From the Fig. 7.3, it can be seen that the curves for ΔT_f^n and ΔC_{total}^n with identical value of σ are more sparse when

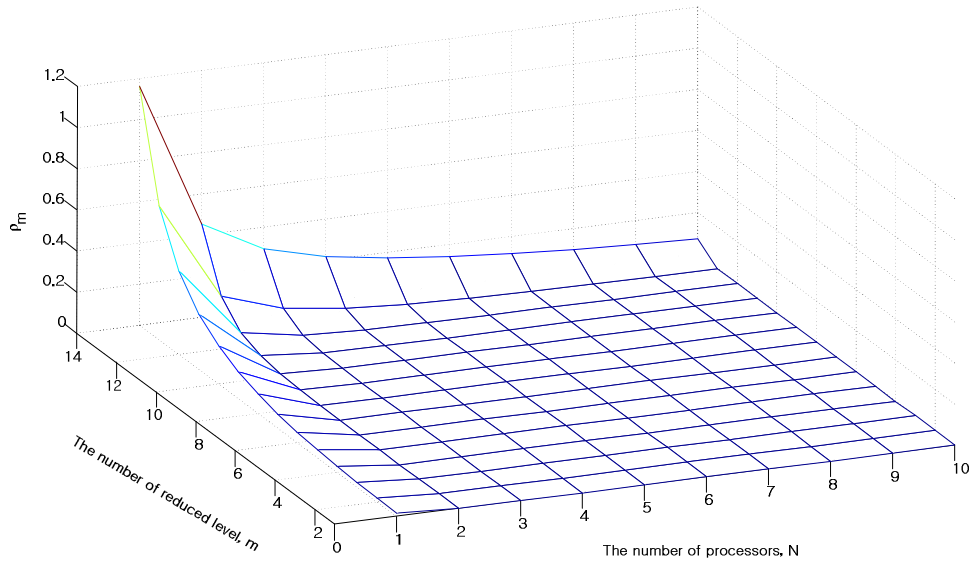


Figure 7.4: ρ_m vs. the number of processors, N and m.

the value of σ is small. This corresponds to the analytical result in (7.14) that the ratio ρ_n is inversely proportional to σ. The simulation data from the Fig. 7.3 clearly shows that the ratio ρ_n has a constant value according to the three different values of σ (i.e., $\rho_n = 2, 10,$ and 20).

Also, ρ_m, the ratio the amount of variation in total processing finish time, $\Delta T_f^{N,m}$ to the amount of variation in the total monetary cost, $\Delta C_{total}^{N,m}$ as the total number of levels of the tree is reduced by m is depicted in Fig. 7.4. A homogeneous multi-level tree with $M = 15$, $N = 10$, $w = 1$, $z = 0.1$, $T_{cp} = 1$, $T_{cm} = 1$, $c_p = 1$, and $c_l = 1$ is modeled. The degree of the ratio ρ_m increases more rapidly as the number of tree levels decrement increases and the number of children processors, N decrease.

$C_{total,N}^{M}$ depicted in Fig. 7.5 clarifies the trend of ρ_m in the Fig. 7.4. It can be seen that the degree of the increment of the total cost increases as the tree becomes smaller in the number of levels (see Fig. 7.5). Hence, $\Delta C_{total}^{N,m}$ has a large value especially when the amount of tree level decrement in m increases, and the number of children processors, N decreases as shown in the Fig. 7.4. The saturation phenomenon, in [7], [41], [42] of the total processing finish time, $T_f^{N,M}$ as N and M increases confirms the observation in the Fig. 7.4.

Fig. 7.6 shows speedup of the multi-level tree. The speedup characteristic of the multi-

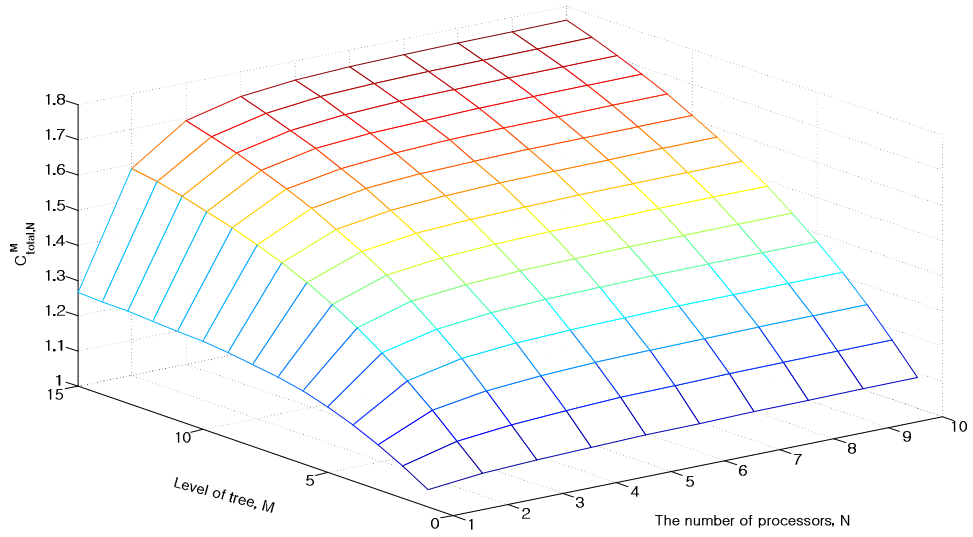


Figure 7.5: Total cost, $C^M_{total,N}$ vs. the number of processors, N and level of tree, M.

level tree was studied in [41]. Combining information of both Fig. 7.5 and Fig. 7.6, the cost efficiency, E_C, defined as (7.44), indicating achievable speedup per unit cost can be plotted as Fig. 7.7. Interestingly, it can be seen that the cost efficiency increases at initial point with low level of tree (moving along the lines for M). Then, the overall cost efficiency has a trend of a smooth decrease as the number of tree levels increase. This phenomenon can be explained as a large degree of increment of the speedup for a small level tree results in the sharp increments of the cost efficiency against total cost consumption showing a relatively smooth increment as shown in the Fig. 7.5. On the other hand, the saturation trend of the speedup shown in the Fig. 7.6 against increasing total cost along the axis for M contributes to the decreasing trend of the cost efficiency as M increases. As seen along the axis for N in Fig. 7.5 and Fig. 7.6, both speedup and total cost have a distinct increase for small number of children processors. Thus, the saturated cost efficiency for large numbers of children processors along the axis for N is reasonable.

7.4 Concluding remarks

In this chapter, monetary network cost trends for homogeneous multi-level tree networks are investigated. Closed form solution and computational techniques are presented for the

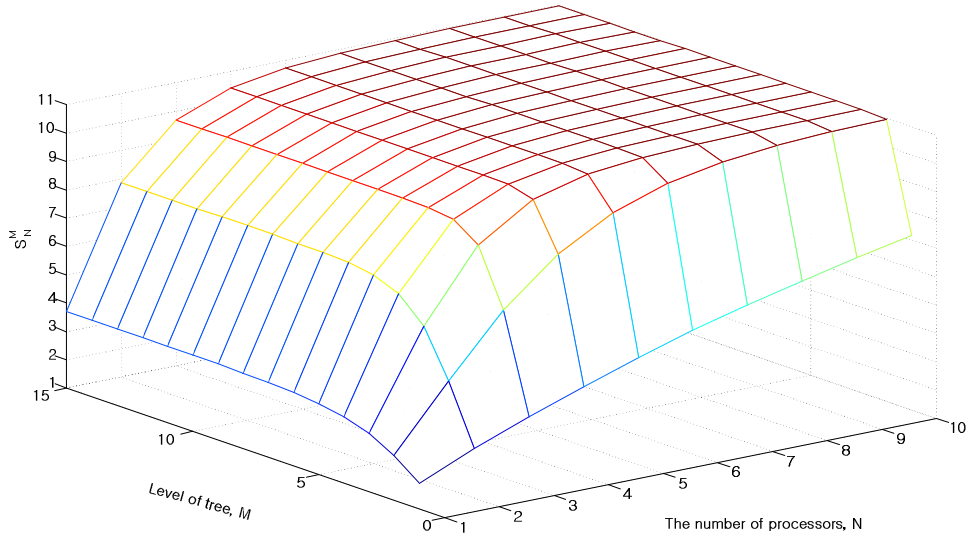


Figure 7.6: Speedup, S_N^M vs. the number of processors, N and level of tree, M.

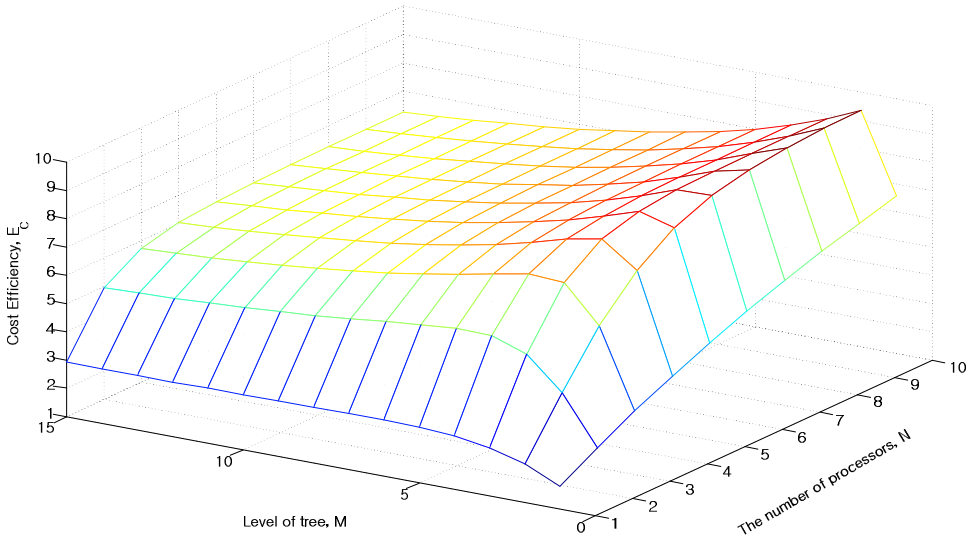


Figure 7.7: Cost efficiency, E_C vs. the number of processors, N and level of tree, M.

determination of optimal network cost in a multi-level tree network. By analyzing the trade-off between total monetary network cost and total processing finish time according to the change of network variables, trends for the monetary cost for multi-level tree can be investigated. Information on network conditions to maintain cost efficient parallel processing is obtained through simulation. By investigating the cost efficiency of a multi-level tree, relationships between monetary network cost and network performance can be understood. We believe that this study is a key to achieve optimal monetary network cost for other complex networks of processors and links.

Chapter 8

Future Work

Large-scale parallel processing that focuses on computer systems that utilize thousands of processors and beyond is a very active research area. It is given the goal of many researchers world-wide to enhance science-by-simulation through installing large-scale multi-petaflop systems at the start of the next decade. Large-scale systems, referred to by some as extreme-scale and ultra-scale, have many important research aspects that need detailed examination in order for their effective design, deployment, and utilization to take place. These include handling the substantial increase in multi-core on a chip, the ensuing interconnection hierarchy and communication.

In this dissertation I have been applying DLT to various network topologies and their network performance evaluation. DLT has been proven that its mathematical tractability is very powerful and can provide evaluations and predictions of network performance. Further, I see DLT as one of the outstanding means of network scheduling for the ongoing development of various high performance network architectures of increasing system scale. Going to Terascale / petascale high performance computing (HPC) systems and beyond means that the number of components (cores, interconnect, storage) within such a system will grow enormously. It is natural that these highly parallel systems will raise questions about reliable information about resource management and scheduling. Thus, it can be expected that adaptation of DLT is a promising technique for a new era of HPC, especially for

resource allocation, computing power adjusted according to computing load, and network performance prediction in a parallel paradigm.

As an initial extended work, parallel processing based on algorithms of nonlinear computational complexity is being studied. I believe that this work is significant for being a break through of the traditional idea underlying DLT, a linear relationship between temporal concept and the amount of load. A mathematical treatment of the concepts of computation time as a measure of complexity in distributed computation corresponds to the basic purpose of DLT.

Further, based on the theoretical framework of DLT, research can extended to experimental studies with real grid and clustered computing environments to create a more realistic and general DLT paradigm [65],[75].

Bibliography

[1] V. Bharadwaj, D. Ghose, V. Mani, and T. G. Robertazzi, *Scheduling Divisible Loads in Parallel and Distributed Systems*, IEEE Computer Society Press, Los Alamitos CA, 1996.

[2] V. Bharadwaj, D. Ghose, and T. G. Robertazzi, "A New Paradigm for Load Scheduling in Distributed Systems," *Special Issue of Cluster Computing on Divisible Load Scheduling*, Vol. 6, No. 1, 2003, pp. 7-18.

[3] T. G. Robertazzi, "Ten Reasons to Use Divisible Load Theory," *Computer*, Vol 36, No. 5, 2003, pp. 63-68.

[4] Y. C. Cheng and T. G. Robertazzi, "Distributed Computation with Communication Delays," *IEEE Trans. Aerospace and Electronic Systems*, Vol. 24, No. 6, 1988, pp. 700-712.

[5] Y. C. Cheng and T. G. Robertazzi, "Distributed Computation for a Tree Network with Communication Delays," *IEEE Trans. Aerospace and Electronic Systems*, Vol. 26, No. 3, 1990, pp. 511-516.

[6] S. Bataineh and T. G. Robertazzi, "Bus Oriented Load Sharing for a Network of Sensor Driven Processors," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 21, No. 5, 1991, pp. 1202-1205.

[7] S. Bataineh, T. Y. Hsiung, and T. G. Robertazzi, "Closed Form Solutions for Bus and

Tree Networks of Processors Load Sharing a Divisible Job," *IEEE Trans. Computers*, Vol. 43, No. 10, 1994, pp. 1184-1196.

[8] H. J. Kim, G. I. Jee, and J. G. Lee, "Optimal Load Distribution for Tree Network Processors," *IEEE Trans. Aerospace and Electronic Systems*, Vol. 32, No. 2, 1996, pp. 607-612.

[9] J. Sohn and T. G. Robertazzi, "Optimal Divisible Job Load Sharing on Bus Networks," *IEEE Trans. Aerospace and Electronic Systems*, Vol. 32, No. 1, 1996, pp. 34-40.

[10] J. Blazewicz and M. Drozdowski, "The Performance Limits of a Two-dimensional Network of Load Sharing Processors," *Foundations of Computing and Information Sciences*, Vol. 21, No. 1, 1996, pp. 3-15.

[11] J. Blazewicz and M. Drozdowski, "Scheduling divisible jobs on hypercubes," *Parallel Computing*, Vol. 21, No. 12, 1995, pp. 1945-1956.

[12] J. Blazewicz, M. Drozdowski, F. Guinand, and D. Trystram, "Scheduling a Divisible Task in a Two-dimensional Toroidal Mesh," *Discrete Applied Mathematics*, Vol. 94, No. 1, 1999, pp. 35-50.

[13] M. Drozdowski and W. Glazek, "Scheduling Divisible Loads in a Three-Dimensional Mesh of Processors", *Parallel Computing*, Vol. 25, No. 4, 1999, pp. 381-404.

[14] K. Li, "Speedup of Parallel Processing of Divisible Loads on k-Dimensional Meshes and Tori," *The Computer Journal*, Vol. 46, No. 6, 2003, pp. 625-631.

[15] K. Li, "Improved Methods for Divisible Load Distribution on k-Dimensional Meshes using Pipelined Communications," *IEEE Trans. Parallel and Distributed Systems*, Vol. 14, No. 12, 2003, pp. 1250-1261.

[16] J. Sohn and T. G. Robertazzi, "Optimal Time-varying Load Sharing for Divisible Loads," *IEEE Trans. Aerospace and Electronic Systems*, Vol. 34, No. 3, 1998, pp. 907-922.

[17] V. Bharadwaj, D. Ghose, and V. Mani, "Optimal Sequencing and Arrangement in Distributed Single-Level Networks with Communication Delays," *IEEE Trans. Parallel and Distributed Systems*, Vol. 5, No. 9, 1994, pp. 968-976.

[18] M. Drozdowski and M. Lawenda, "The Combinatorics in Divisible Load Scheduling," *Foundations of Computing and Decision Sciences*, Vol. 30, No. 4, 2005, pp. 297-308.

[19] M. Drozdowski and M. Lawenda, "Scheduling Multiple Divisible Loads," *The Int'l J. High Performance Computing Applications*, Vol. 20, No. 1, 2006, pp. 19-30.

[20] M. Drozdowski and P. Wolniewicz, "Divisible Load Scheduling in Systems with Limited Memory," *Cluster Computing*, Vol. 6, No. 1, 2003, pp. 19-29.

[21] X. Li, V. Bharadwaj, and C. C. Ko, "Optimal Divisible Task Scheduling on Single-Level Tree Networks with Buffer Constraints," *IEEE Trans. Aerospace and Electronic Systems*, Vol. 36, No. 4, 2000, pp. 1298-1308.

[22] M. Drozdowski and P. Wolniewicz, "Optimum Divisible Load Scheduling on Heterogeneous Stars with Limited Memory," *European Journal of Operational Research*, Vol. 172, No. 2, 2006, pp. 545-559.

[23] H. P. L. Diana, V. Bharadwaj, and B. A. David, "On the Design of High-Performance Algorithms for Aligning Multiple Protein Sequences in Mesh-Based Multiprocessor Architectures," *Journal of Parallel and Distributed Computing*, Vol. 67, No. 9, 2007, pp. 1007-1017.

[24] D. Ghose and H. J. Kim, "Load Partitioning and Trade-Off Study for Large Matrix Vector Computations in Multicast Bus Networks with Communication Delays," *Journal of Parallel and Distributed Computing*, Vol. 55, No. 1, 1998, pp. 32-59

[25] V. Bharadwaj and S. Ranganath, "Theoretical and Experimental Study of Large Size Image Processing Applications using Divisible Load on Distributed Bus Networks", *Image and Vision Computing*, 2002.

[26] C. T. Teo, V. Bharadwaj, and J. Jingxi, "Handling Large-Size Discrete Wavelet Transform on Network-Based Computing Systems: Parallelization via Divisible Load Paradigm," to appear in the *Journal of Parallel and Distributed Computing*, 2009.

[27] X. Lin, L. Ying, J. Deogun, and S. Goddard, "Real-Time Divisible Load Scheduling for Cluster Computing," *The 13th IEEE Real-Time/Embedded Technology and Applications Symposium (RTAS)*, 2007.

[28] X. Lin, L. Ying, J. Deogun, and S. Goddard, "Real-Time Divisible Load Scheduling with Different Processor Available Times," *The Int'l Conf. Parallel Processing (ICPP)*, 2007.

[29] V. Bharadwaj, D. Ghose, and V. Mani, "Multi-installment Load Distribution in Tree Networks With Delays," *IEEE Trans. Aerospace and Electronic Systems*, Vol. 31, No. 2, 1995, pp. 555-567.

[30] V. Bharadwaj, H. F. Li, and T. Radhakrishnan, "Scheduling Divisible Loads in Bus Networks with Arbitrary Processor Release Time," *Computer Math. Applic*, Vol. 32, No. 7, 1996, pp. 57-77.

[31] V. Bharadwaj and G. Barlas, "Scheduling Divisible Loads with Processor Release Times and Finite Size Buffer Capacity Constraints in Bus Networks," *Cluster Computing*, Vol. 6, No. 1, 2003, pp. 63-74.

[32] V. Bharadwaj and H. M. Wong, "Scheduling Divisible Loads on Heterogeneous Linear Daisy Chain Networks with Arbitrary Processor Release Times," *IEEE Trans. Parallel and Distributed Systems*, Vol. 15, No. 3, 2004, pp. 273-288.

[33] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *IEEE Computer*, Vol. 38, No. 4, 2002, pp. 393-422.

[34] Z. J. Haas and S. Tabrizi, "On Some Challenges and Design Choices in Ad-hoc Communications," *IEEE MILCOM '98*, 1998.

[35] W. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Trans. Wireless Communications*, Vol. 1, No. 4, 2002, pp. 660-670.

[36] S. Lindsey and C. S. Raghavendra, "Pegasis: Power-Efficient Gathering in Sensor Information System," *Proc. of IEEE Aerospace Conference*, 2002, pp. 924-935.

[37] M. Moges and T. G. Robertazzi, "Wireless Sensor Networks: Scheduling for Measurement and Data Reporting," *IEEE Trans. Aerospace and Electronic Systems*, Vol. 42, No. 1, 2006, pp. 327-340.

[38] C. F. Gamboa and T. G. Robertazzi, "Efficient Scheduling for Sensing and Data Reporting in Wireless Sensor Networks," *2006 Conf. Information Sciences and Systems*, 2006.

[39] H. Li, S. Jiang, and G. Wei, "Information-Accuracy-Aware Jointly Sensing Nodes Selection in Wireless Sensor Networks," *Mobile Sensor Network Conference*, 2006, pp. 736-747.

[40] J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," *Computer*, Vol. 34, No. 8, 2001, pp. 57-66.

[41] J. T. Hung and T. G. Robertazzi, "Scalable Scheduling for Clusters and Grids using Cut Through Switching," *Int'l J. Computers and Applications*, Vol. 26, No. 3, 2004, pp. 147-156.

[42] J. T. Hung and T. G. Robertazzi, "Divisible Load Cut Through Switching in Sequential Tree Networks," *IEEE Trans. Aerospace and Electronic Systems*, Vol. 40, No. 3, 2004, pp. 968-982.

[43] W. R. Heinzelman, Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *Proc. of 33rd IEEE Annual Hawaii International Conference on Systems Sciences*, 2000.

[44] O. Younis and S. Fahmy, "An Experimental Study of Routing and Data Aggregation in Sensor Networks," *Proc. IEEE Int'l Workshop on Localized Communication and Topology Protocols for Ad Hoc Networks (LOCAN)*, 2005.

[45] Relativistic Heavy Ion Collider (RHIC): http://www.bnl.gov/rhic/.

[46] The STAR Experiment at RHIC: http://www.star.bnl.gov/.

[47] ATLAS Computing Technical Design Report, ATLAS-TDR-017, 2005: http://cdsweb.cern.ch/record/837738.

[48] The TeraPaths End-to-End QoS Networking Project: http://www.terapaths.org.

[49] The Lambda Station project: http://www.lambdastation.org.

[50] The Phoebus Project: http://e2epi.internet2.edu/phoebus.html.

[51] On-demand Secure Circuits and Advance Reservation System (OSCARS): http://www.es.net/oscars/.

[52] T. Lehman, X. Yang, C. P. Guok, N. S. V. Rao, A. Lake, J. Vollbrecht, and N. Ghani, "Control Plane Architecture and Design Considerations for Multi-Service Multi-Layer, Multi-Domain Hybrid Networks," *INFOCOM 2007 High Speed Networks Workshop*, 2007, pp. 67-71.

[53] S. Bradley, F. Burstein, L. Cottrell, B. Gibbard, D. Katramatos, Y. Li, S. McKee, R. Popescu, D. Stampf, and D. Yu, "TeraPaths: A QoS-enabled Collaborative Data Sharing Infrastructure for Peta-scale Computing Research," *Proc. of Computing in High Energy and Nuclear Physics*, 2006, pp. 13-17.

[54] D. Katramatos, B. Gibbard, D. Yu, and S. McKee, "TeraPaths: End-to-End Network Path QoS Configuration using Cross-Domain Reservation Negotiation," *Proc. of the 3rd Int'l Conf. Broadband Communications, Networks, and Systems*, 2006, pp. 1-5.

[55] D. Katramatos, B. Gibbard, D. Yu, and S. McKee, "The TeraPaths Testbed: Exploring End-to-End Network QoS," *Proc. of the 3rd Int'l Conf. Testbeds and Research Infrastructure for the Development of Networks and Communities and Workshops (TridentCom 2007)*, 2007, pp. 21-23.

[56] ESnet: http://www.es.net.

[57] Internet2: http://www.internet2.edu/network/dc/.

[58] K. Park, G. Kim, and M. Crovella, "On the Relationship Between File Sizes, Transport Protocols, and Self-Similar Network Traffic," *Proc. the Int'l Conf. Network Protocols*, 1996, pp. 171-180.

[59] K. Ko and T. G. Robertazzi, "Scheduling in an Environment of Multiple Job Submissions," *Proc. of the 2002 Conf. Information Sciences and Systems*, 2002.

[60] H. M. Wong, D. Yu, V. Bharadwaj, and T. G. Robertazzi,"Data Intensive Grid Scheduling: Multiple Sources with Capacity Constraints," *Proc. of the IASTED International Conference on Parallel and Distributed Computing and Systems*, 2003.

[61] S. Viswanathan, V. Bharadwaj, and T. G. Robertazzi,"Resource Aware Distributed Scheduling Strategies for Large-Scale Computational Cluster/Grid Systems," *IEEE Trans. Parallel and Distributed Systems*, Vol. 18, No. 10, 2007, pp. 1450-1461.

[62] M. Moges, D. Yu, and T. G. Robertazzi,"Grid Scheduling Divisible Loads from Multiple Sources via Linear Programming," *16th IASTED Int'l Conf. Parallel and Distributed Computing and Systems*, 2004.

[63] M. Moges, D. Yu and T. G. Robertazzi,"Divisible Load Scheduling with Multiple Multiple Sources: Closed Form Solutions," *2005 Conf. Information Sciences and Systems*, 2005.

[64] T. G. Robertazzi and D. Yu,"Multi-Source Grid Scheduling for Divisible Loads," *2006 Conf. Information Sciences and Systems*, 2006.

[65] D. Yu and T. G. Robertazzi,"Divisible Load Scheduling for Grid Computing," *Proc. of the IASTED Int'l Conf. Paralle and Distributed Computing and Systems*, 2003.

[66] J. Sohn, T. G. Robertazzi, and S. Luryi, "Optimizing Computing Costs Using Divisible Load Analysis," *IEEE Trans. Parallel and Distributed Systems*, Vol. 9, No. 3, 1998, pp. 225-234.

[67] S. Charcranoon, T. G. Robertazzi, and S. Luryi, "Parallel Processor Configuration Design with Processing/Transmission Costs," *IEEE Trans. Computers*, Vol. 49, No. 9, 2000, pp. 987-991.

[68] S. Charcranoon, T. G. Robertazzi, and S. Luryi, "Load Sequencing for a Parallel Processing Utility," *Journal of Parallel and Distributed Computing*, Vol. 64, No. 1, 2004, pp. 29-35.

[69] K. Choi and T. G. Robertazzi, "Divisible Load Scheduling in Wireless Sensor Network with Information Utility," *Proc. of IEEE Int'l Performance Conf. Computers and Communication (IPCCC '08)*, 2008, pp. 9-17.

[70] M. Drozdowski and L. Wielebski, "Efficiency of Divisible Load Processing," *Parallel Processing and Applied Mathematics*, Vol. 3019, 2004, pp. 175-180.

[71] M. Drozdowski and L. Wielebski, "Isoefficiency Maps for Divisible Computations," to appear *IEEE Trans. Parallel and Distributed Systems.*

[72] A. Gupta and V. Kumar, "Performance Properties of Large Scale Parallel Systems," *Journal of Parallel and Distributed Computing*, Vol. 19, No. 3, 1993, pp. 234-244.

[73] K. Choi and T. G. Robertazzi, "Isocost Performance Evaluation Using Parallel Divisible Load Scheduling," submitted to *the 29th Conf. Computer Communications* (INFOCOMM 2010), 2009.

[74] K. Choi and T. G. Robertazzi, "Cost Performance Analysis in Parallel Computing Networks with Divisible Load Scheduling," to appear in *the 21st IASTED Int'l Conf. Parallel and Distributed Computing and Systems*, 2009.

[75] M. Drozdowski and P. Wolniewicz, "Experiments with Scheduling Divisible Tasks in Clusters of Workstations," *EURO-Par-2000*, LNCS 1900, Springer-Verlag, 2000, pp. 311-319.

Appendix A

Discretization of Continuous Integration

With the special characteristic that the status of the capacity reservation can be described as a function composed with step functions, one can derive a discrete expression for an integration process within a continuous domain. We can rewrite eq (4.3) based on Fig. 4.2 as

$$C(t) = \sum_{i=1} \Big(\sum_{k=1}^{i} D_k \Big) \Big\{ U(t - t_i) - U(t - t_{i+1}) \Big\} \tag{A.1}$$

where

$$D_k = \begin{cases} \textit{negative real} & \text{for VP setup at } t_k \\ \textit{positive real} & \text{for VP teardown at } t_k \\ 0 & \text{for simultaneous VP setup and VP teardown at } t_k \end{cases}$$

In our example, D_2 equals 0 based on the assumption that VP setup and VP teardown occur at the same time, t_2 (see Fig. 4.2). By substituting eq (A.1) into $\int_{T-W}^{T} C(t)dt$ from

(4.6), we have

$$\int_{T-W}^{T} C(t)dt = \sum_{i=1} \Big(\sum_{k=1}^{i} D_k \Big) \int_{T-W}^{T} \Big\{ U(t-t_i) - U(t-t_{i+1}) \Big\} dt \tag{A.2}$$

Here,

$$\int_{T-W}^{T} U(t-t_i)dt = \begin{cases} W & \text{if } t_i \le T-W \\ T-t_i & \text{if } T-W < t_i \le T \\ 0 & otherwise \end{cases}$$

$$\int_{T-W}^{T} U(t-t_{i+1})dt = \begin{cases} W & \text{if } t_{i+1} \le T-W \\ T-t_{i+1} & \text{if } T-W < t_{i+1} \le T \\ 0 & otherwise \end{cases}$$

Thus,

$$\begin{aligned} \int_{T-W}^{T} C(t)dt &= \Big\{ \sum_{i=1}^{n} \Big(\sum_{k=1}^{i} D_k \Big) - \sum_{i=1}^{n-1} \Big(\sum_{k=1}^{i} D_k \Big) \Big\} W + \Big\{ \sum_{i=n+1}^{m} \Big(\sum_{k=1}^{i} D_k \Big) (T-t_i) - \\ &\quad \sum_{i=n}^{m-1} \Big(\sum_{k=1}^{i} D_k \Big) (T-t_{i+1}) \Big\} \\ &= W \sum_{k=1}^{n} D_k + T \cdot \Big(\sum_{i=n+1}^{m} \sum_{k=1}^{i} D_k - \sum_{i=n}^{m-1} \sum_{k=1}^{i-1} D_k \Big) - \sum_{i=n+1}^{m} t_i \Big(\sum_{k=1}^{i} D_k - \sum_{k=1}^{i-1} D_k \Big) \\ &= W \sum_{k=1}^{n} D_k + T \cdot \Big(\sum_{k=1}^{m} D_k - \sum_{k=1}^{n} D_k \Big) - \sum_{i=n+1}^{m} t_i \Big(\sum_{k=1}^{i} D_k - \sum_{k=1}^{i-1} D_k \Big) \end{aligned} \tag{A.3}$$

where the integer value, n is the index i satisfying

$t_i \le T-W < t_{i+1}$ and m is the index i satisfying

$t_i \le T < t_{i+1}$.

Using the fact that,

$$\begin{aligned}
\sum_{k=1}^{n} D_k &= C((T-W)) \\
\sum_{k=1}^{m} D_k &= C((T)) \\
\sum_{k=1}^{i} D_k &= C(t_i) \\
\sum_{k=1}^{i-1} D_k &= C(t_{i-1})
\end{aligned} \tag{A.4}$$

finally, we can simplify (A.3) further to a form independent of variable D_k as

$$\int_{T-W}^{T} C(t)dt = T \cdot C(T) - (T-W) \cdot C(T-W) - \sum_{i=n+1}^{m} t_i \Big(C(t_i) - C(t_{i-1}) \Big) \tag{A.5}$$