Stony Brook University

Data Acquisition Scheduling for Wireless Sensors Networks

A Dissertation Presented

by

Carlos Gamboa

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

December 2008

Stony Brook University

The Graduate School

Carlos Gamboa

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Thomas Robertazzi, Advisor of Dissertation

Professor, Department of Electrical and Computer Engineering

Sangjin Hong, Chairperson of Defense

Associate Professor, Department of Electrical and Computer Engineering

Wendy Tang

Associate Professor, Department of Electrical and Computer Engineering

Esther Arkin

Professor, Department of Applied Mathematics and Statistics

Lawrence Martin

Dean of the Graduate School

Abstract of the Dissertation

Data Acquisition Scheduling for Wireless Sensors Networks

by

Carlos Gamboa

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2008

Wireless sensor networks have been increasingly studied by industry and academic institutions due to their potential application in environmental and geothermal monitoring, security enhancement among others. Such technology would play an important role in future societies and would be the key tool to make improvements in productivity and efficiency fast and on a large scale. In this work it is proposed to use Divisible Load scheduling Theory (DLT) as a mathematical tool to study data load distribution on wireless sensors networks under time and monetary cost constraints. Pursuing this aim, five different studies have been conducted. The first one considered the total monetary cost optimization, and a sensitivity analysis in a single level tree network. The problem of load distribution sequencing for optimizing monetary cost in a single level tree network is reviewed and

simulation results showed that the monetary cost function depends on the fractions of loads assigned to each processor. In addition, a sensitivity analysis performed on the monetary cost function suggested a complex relationship between the cost function and the network parameters. The second study explores such relationship proposing a strategy to study the monetary total cost in a single level tree star network as a function of a non-linear load parameter. The third study investigated three scheduling load protocols on wireless sensors networks. A novel closed form solution was found for each of the protocols presented for optimum finish, reporting and pre-processing time for a single level tree sensor network including an immediate measuring data feature. In the fourth study an adaptable data scheduling methodology is presented. The schedule protocol presented here takes into account the amount of load processed from previous load assignments based on distributing the new incoming load to the sensor network aiming to minimize the total finish time of processing the entire Divisible Load Job Task (DLJT) and reducing the idle state of each processor. Finally, the performance in terms of speedup for a single level tree network with multiple links and cores is investigated.

To my beloved wife and children

Table of Contexts

List of Figures

List of Tables

Acknowledgments

I would like to start by thanking my advisor Dr. Thomas Robertazzi. Besides his valuable advices and time, he showed me that with patience, hard work and creativity an idea could become a fact. This research project is an example of that.

During the past five years different many other people and institutions contributed to this research project in many ways. I would like to thank NSF-AGEP at Stony Brook which, through a Fellowship, provided me with the necessary funding and support during the first year of my graduate school and then allowed me to work as a graduate assistant giving me not only the financial means to cover one more year of studies and research but the opportunity to work with different AGEP students and staff and learn from them. In addition, I would like to thank the Electrical Engineering department especially Debbie Kloppenburg for her logistic support, Professor Sussman-Fort for allowing me to be his teacher assistant and Professor Vera Gorfinkel for her guidance and time while preparing for

the qualifying examination. I would also like to thank the NFS-MARIACHI project for allowing me to work with them during part of my third year of Graduate School.

To finish I would like to thank my family in Colombia: my brother Leonardo, my sister Monica and my mother Elizabeth. Their example of strength facing bad times inspired me to keep working. In the same way I would like to thank my North American family, especially Fred, Bob and Sharon Boutcher.

I would like to thank and dedicate this work to my wife Johanna and my children Gabriel and baby Daniel. This work represents the effort of all.

Chapter 1

Introduction

Among the next generation technology challenges are the design of an energy efficient technology that can be used to acquire data, transmit and process it in a robust and efficient fashion. In addition such technology has to be inexpensive and in some cases easily configurable.

Such systems will be required to be able to work in complex distributed situations to acquire data, process it and eventually to be able to make decisions without the direct intervention of human judgment. Researchers from industry and some academic institutions envision the use of wireless sensor networks as a key technology that could be use to supply such needs.

The first study that used DLT to study wireless sensors networks was presented in [1]. Different scheduling policies were presented where the sensor has a measuring capacity. An energy use strategy for the same model was presented as well. A divisible load is a load that can be arbitrarily partitioned in a linear fashion and can be distributed to more than one processor to achieve a faster solution time.

Initially a linear daisy chain network was considered to study optimal divisible load sharing [2] becoming the first study that considered the DLT optimality principle. Since then more than 15 years of research on Divisible Load Theory has proved that this model is a tractable and suitable tool that can be use in different applications such as parallel and distributed processor network scheduling, data intensive computing, grid computing and metacomputing [3].

Wired network configurations topology studies that have considered optimum processing finish time can be found in [4] and [5]. Here, the general assumption is that in order to obtain an efficient allocation of load on each processor in the network the processors have to stop processing at the same time. Optimal allocation of loads for network topologies including bus networks and tree networks using a set of recursive equations were studied in [6]–[7]. For complex networks, the concept of equivalent networks was presented in [8].

Economic models for computers and telecommunications networks can be referred to [8], [9] and [10]. The first study that related divisible load distributions under monetary cost constrains was presented in [11]. Heuristics algorithms for optimizing the total monetary cost function in a single level tree network function was presented [12]. There, it was shown that the total monetary depends on the order in which a root processor distributes load to its child processors within a network. In addition in [13] the relationship between load distribution and energy use in a single level

tree wired network was introduced as a function of the network parameter used in the DLT model. In [14] the relationship between a nonlinear load and the total monetary cost was explored.

1.1 Research Purpose

It is proposed to study wireless sensors network under energy, monetary cost and finish processing time constraints using Divisible scheduling Load Theory (DLT) with the aim of developing a solid theoretical and practical tool that could be used in the data acquisition processes.

1.1.1 Research Objectives:

1. To develop load distribution policies and sequences for minimal finish time in wireless sensor networks using DLT.

2. To investigate the relationship between total monetary cost and load measured in a wireless sensor network under DLT model.

3. To propose heuristic algorithms to study data load distribution on wireless sensor networks.

1.2 Justification of the Study

Recent studies agreed that wireless sensor networks is a promising technology that can contribute to solve problems where a traditional wired data acquisition systems can not be used due to the cost of deployment or

geographical complexity. Scientists envision that the uses of this technology would reduce costs on the data acquisition process and similarly this technology would allow them to acquire information in strategic places to monitor and sense data constantly such as disaster prevention applications (volcanos eruption, building collapse), security, and scientific studies.

Due to its tractability, DLT has become a reliable mathematical tool to study distribution of data load between processors over the pass 15 years producing more than seventy journal papers. The main aspects (minimum finish time, monetary cost) studied in this research have been researched within the wired network world using DLT. More recently Moges and Robertazzi in 2006 [1] made the first attempt to use DLT in wireless sensor networks obtaining important results and introducing the concept of data measured into the formal DLT model. Despite this important first effort to introduce the DLT as a mathematical tool to characterized load distribution and energy use in wireless sensor networks there is a lot to be considered when describing such technology using DLT. In using DLT it is possible to describe or design schedule policies that take into account energy availability in each sensor to perform a robust and reliable load distribution policy. The classical combinatorial algorithms used in DLT wired sensors networks have to be reviewed and in some cases have to be changed. We attempt to outperform traditional mathematical techniques for the distribution of load, minimal energy of use and monetary cost.

A further general question can be stated as can the nature of divisible

load theory be used to described wireless sensors networks not only from the theoretical point of view but from the practical?

1.3 Research Significance

This research work intends to contribute in advancing knowledge from two aspects on wireless sensors networks:

1. Theoretical: A new mathematical strategy is developed to described the behavior of wireless sensors networks under energy, finish time and monetary cost constraints. In doing so a new set of theoretical assumptions will be considered that would be required to develop algorithms and numerical experiments and simulations in order to compare them and validate their effectiveness. As a consequence this study, we not only attempt to propose a transparent theoretical mechanism to model wireless sensor networks but also hope to contribute to the literature.

2. Practical: No doubt that concentrating efforts to develop this technology can help society to reduce cost in the acquisition of information where conventional mechanism are too expensive or are not specially accessible with conventional wired sensor networks. To include a monetary cost study on this research would help to develop a coherent model that takes into account not only technical aspects but economical aspects that should help realize potentially significant applications of wireless sensor

networks.

1.4 Organization of the Report

This report is organized as follows:

Chapter 2 is considers an innovative load scheduling strategy designed for wireless sensor networks. An analytical expression, for optimal load assignment and finish time is presented. This strategy has the potential to reduce solution time (make span) significantly.

Chapter 3 begins to explore the total monetary cost model for a single level tree network using DLT based on the model presented on [11]. Our main goal pursued in this chapter is to understand how changes on intrinsic parameters of the network affects the total monetary cost function when processing a divisible load. In addition, a combinatorial optimization algorithm was used in [13] is studied and modified to performed a sensitivity analysis.

Chapter 4 a heuristic non-linear total monetary cost function analysis is presented.

Chapter 5 an adaptable data scheduling methodology is presented.

Chapter 6 two different wireless sensor networks architecture using DLT with M parallel links and M parallel cores per processor sensor using the System on Chip technology (SoC) were proposed.

Chapter 7 conclusions are presented.

Chapter 2

Efficient Scheduling for Sensing and Data Reporting in Wireless Sensor Networks

2.1 Summary

This chapter considers an innovative scheduling strategy in which a control processor assigns a load share to be measured by each of N processors organized in a single level tree (star) wireless sensors network. Here processors begin to sense as soon as receiving their own load share assignment rather than waiting for all processors to receive their assignments as done in previous research works. This strategy has the potential to reduce solution time (make span) significantly. We find an analytical expression for optimal load assignment and finish time which is simple to compute and can be implemented in real time.

2.2 Introduction

Wireless sensor networks have been increasingly studied and developed during the last decade due to their potential application fields such as security, geothermal monitoring, traffic control and health care [13], [16]. Among the challenges that this technology faces are the communication constraints (limited bandwidth and transmission energy) which could be the most crucial aspects to be solved for this technology during the next few years. An interesting approach to overcome this critical aspect is to process the measured data and transmit a summarized version of the data measured when the sensor devices and its architecture allows it. Doing so would reduce the amount of data to be transmitted and consequently the energy used for transmission of the reporting data [17].

Another interesting approach was proposed in 2006 [1]. It introduced a novel scheduling strategy which considered single level tree (start) network of N processors and a control processor. In this work, each processor starts to sense when the control processor finishes distributing the entire load share assignments to all of the processors in the network. Here load share assignment means the amount of sensing load that is assigned by the control processor to each of the processors in the network.

As in [1] this paper is done in the context of Divisible Load scheduling Theory (DLT) that has been the focus of attention by researchers [6], [18], [19] and [20] studying data load distribution in parallel and distributed

systems since 1988 [2]. Initially most of the jobs using DLT considered communication and computation as the main parameters of the system to find a optimal divisible (partitionable) load to be processed and transmitted by each processor and link in the network in a minimal amount of time. A partitionable data load is one that can be arbitrarily distributed among the processor in the network and there is no precedence relations between data.

The network architecture considered in this paper includes a control processor that distributes the load share assignment to the other N processors in the network. Thus the processors begin to sense as soon as receiving their own load share assignment rather than waiting for all processors to receive their assignments and after sensing their correspondent fraction of load share each processor returns the result to the control processor. This scheduling protocol is implemented for homogeneous and heterogeneous networks configurations with processors that have the capability to sense and processors that have the potential to sense and compute.

This chapter is organized as follows: In section 2.3, network model and parameters used in this study are presented. In section 2.4, mathematical model and reporting time expressions for optimal allocation of load using divisible load theory are discussed and presented. In section 2.5, performance and evaluation results for the strategies presented in the study are showed and analyzed. Finally, the conclusions for this study

are presented in section 2.6.

2.3 The Network Model and Parameters

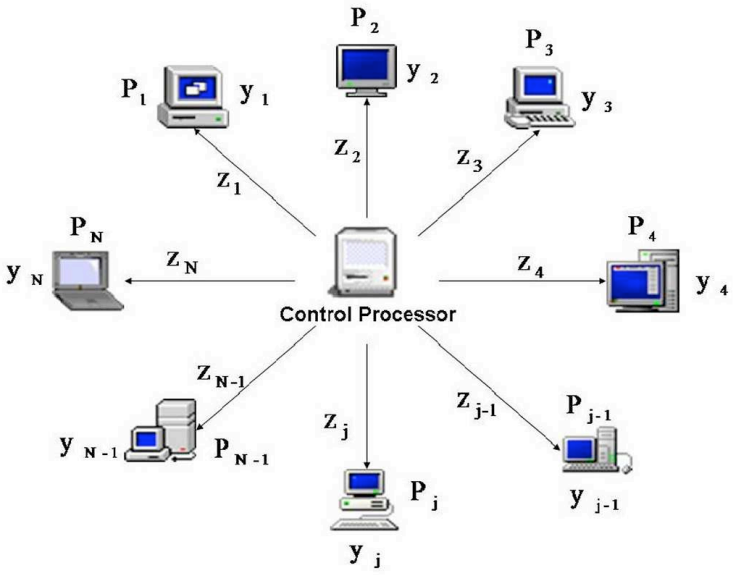

Figure 2.1: Single level tree (star) network with control processor

Consider a single level tree (star) network consisting of N processors and N-1 links as shown in Fig.(2.1). There is a control processor who distributes load share assignments to the other processors in the network sequentially. As soon as each processor receives its own load share assignment it starts to sense. The results are reported back to the control processor sequentially. In some cases, when the network topology allows it, the processor could compute measured data load before transmitting it back to the control processor.

2.3.1 Notations and Definitions

t: Is the time that the control processor takes to assign the measurement instruction to each child processor.

α_j: The load share fraction assigned by the control processor to the j^{th} link-processor pair to be measured.

y_j: A constant that is inversely proportional to the measuring speed of the processor j^{th} in the network.

w_j: The inverse of the computing speed of the j^{th} processor.

z_j: The inverse of the link speed of the j^{th} link.

T_{ms}: Measuring intensity constant: the entire load is processed in $y_j T_{ms}$ seconds by the j^{th} processor.

T_{cp}: Computing intensity constant: the entire load is processed in $w_j T_{cp}$ seconds by the j^{th} processor.

T_{cm}: Communication intensity constant: the entire load can be transmitted in $z_j T_{cm}$ seconds over the j^{th} link.

T_j: Is the total time measured from the beginning of the scheduling process up to the end of the transmission of the data measured by the jth processor.

T_f: Is the time when the last processor finishes reporting. T_f=max(T_1,T_2,...,T_N)

It is assumed that the fractions of load measured are normalized and their addition should sum 1 in the control processor (2.1).

$$1 = \sum_{j=1}^{N} \alpha_j \tag{2.1}$$

2.4 Mathematical Model and Reporting Time

2.4.1 Sequential Reporting Time

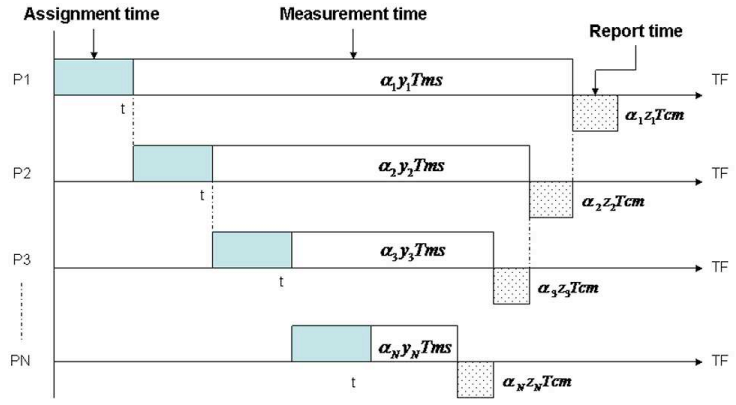

Figure 2.2: Timing diagram for a single level tree network with controller and sequential reporting

The control processor distributes load share assignment to each processor sequentially Fig.(2.2). Each processor starts to measure the data as soon as it receives its load assignment but for this configuration

the processor can only reports its result back sequentially. There is only one channel available for transmission. The mathematical expressions that describe this protocol in terms of the amount of finish time are:

$$T_1 = t + \alpha_1 y_1 T_{ms} + \alpha_1 z_1 T_{cm} \tag{2.2}$$

$$T_2 = 2t + \alpha_2 y_2 T_{ms} + \alpha_2 z_2 T_{cm} \tag{2.3}$$

$$T_3 = 3t + \alpha_3 y_3 T_{ms} + \alpha_3 z_3 T_{cm} \tag{2.4}$$

$$T_N = Nt + \alpha_N y_N T_{ms} + \alpha_N y_N T_{ms} \tag{2.5}$$

It can be seen in Fig.(2.2) that the time required to measure a load share assignment ($\alpha_j y_j T_{ms}$) by the j processor will be equal to the time used by the $j+1$ processor to wait for the assignment (t) and measure ($\alpha_{j+1} y_{j+1} T_{ms}$) and report ($\alpha_{j+1} z_{j+1} T_{cm}$) its result back to the control processor. Thus,

$$\alpha_1 y_1 T_{ms} = t + \alpha_2 y_2 T_{ms} + \alpha_2 z_2 T_{cm} \tag{2.6}$$

$$\alpha_2 y_2 T_{ms} = t + \alpha_3 y_3 T_{ms} + \alpha_3 z_3 T_{cm} \tag{2.7}$$

$$\alpha_{N-3} y_{N-3} T_{ms} = t + \alpha_{N-2}(y_{N-2} T_{ms} + z_{N-2} T_{cm}) \tag{2.8}$$

$$\alpha_{N-2} y_{N-2} T_{ms} = t + \alpha_{N-1}(y_{N-1} T_{ms} + z_{N-1} T_{cm}) \tag{2.9}$$

$$\alpha_{N-1} y_{N-1} T_{ms} = t + \alpha_N(y_N T_{ms} + z_N T_{cm}) \tag{2.10}$$

The previous equation system can be expressed in terms of f_j and s_j

as,

$$\alpha_1 = f_1 + \alpha_2 s_{(1)} \tag{2.11}$$

$$\alpha_2 = f_2 + \alpha_3 s_{(2)} \tag{2.12}$$

$$\alpha_{N-2} = f_{N-2} + \alpha_{N-1} s_{(N-2)} \tag{2.13}$$

$$\alpha_{N-1} = f_{N-1} + \alpha_N s_{(N-1)} \tag{2.14}$$

where

$$f_j = t/y_j T_{ms} \tag{2.15}$$

$$s_j = (y_{j+1} T_{ms} + z_{j+1} T_{cm})/y_j T_{ms} \tag{2.16}$$

The equation system previously presented consists of N-1 recursive equations. Substituting recursively equation (2.14) into equation (2.13) and so on for j processor in term of α_N we have

$$\alpha_j = f_j + \sum_{m=1}^{N-j-1} f_{j+m} \prod_{t=0}^{m-1} s_{j+t} + \alpha_N \prod_{l=0}^{N-j-1} s_{j+l} \tag{2.17}$$

where j=1,2,3,....,N-2.

As mentioned before the normalization equation is the expression that states that the total amount of load is originated in the control processor and has to be 1. In order to have a closed solution for different load share assignments the expression (2.17) is used and evaluated for j=1,2,3,...,N-2. As a consequence,

$$\alpha_1 = f_1 + \sum_{m=1}^{N-2} f_{1+m} \prod_{t=0}^{m-1} s_{1+t} + \alpha_N \prod_{l=0}^{N-2} s_{1+l} \tag{2.18}$$

$$\alpha_2 = f_2 + \sum_{m=1}^{N-3} f_{2+m} \prod_{t=0}^{m-1} s_{2+t} + \alpha_N \prod_{l=0}^{N-3} s_{2+l} \tag{2.19}$$

$$\alpha_3 = f_3 + \sum_{m=1}^{N-4} f_{3+m} \prod_{t=0}^{m-1} s_{3+t} + \alpha_N \prod_{l=0}^{N-4} s_{3+l} \tag{2.20}$$

$$\alpha_{N-2} = f_{N-2} + \sum_{m=1}^{1} f_{N-2+m} \prod_{t=0}^{m-1} s_{N-2+t} + \alpha_N \prod_{l=0}^{1} s_{N-2+l} \tag{2.21}$$

And for $j = N - 1$ the equation is

$$\alpha_{N-1} = f_{N-1} + \alpha_N s_{N-1} \tag{2.22}$$

Substituting the expressions for α from equations (2.18-2.22) into the normalization equation (2.1) α_N can be found as,

$$\alpha_N = \frac{1 - \left[f_{N-1} + \sum_{j=1}^{N-2} f_j + \sum_{m=1}^{N-j-1} f_{j+m} \prod_{t=0}^{m-1} s_{j+t} \right]}{1 + s_{N-1} + \sum_{j=1}^{N-2} \prod_{l=0}^{N-j-1} s_{j+l}} \tag{2.23}$$

Consequently we obtain and expression for α_N (2.23) as a function of the network parameters which allows us to calculate the load share assignments α_{N-1} using equation (2.22) and for α_1, ... , α_{N-2} using equation (2.17).

The minimum finish time for this network configuration using this protocol can be found using equations (2.17) for j=1, (2.23), and

substituting them into (2.2).

$$T_1 = t + [y_1 T_{ms} + z_1 T_{cm}]\,\alpha_1 \tag{2.24}$$

2.4.2 Simultaneous Reporting Finish Time

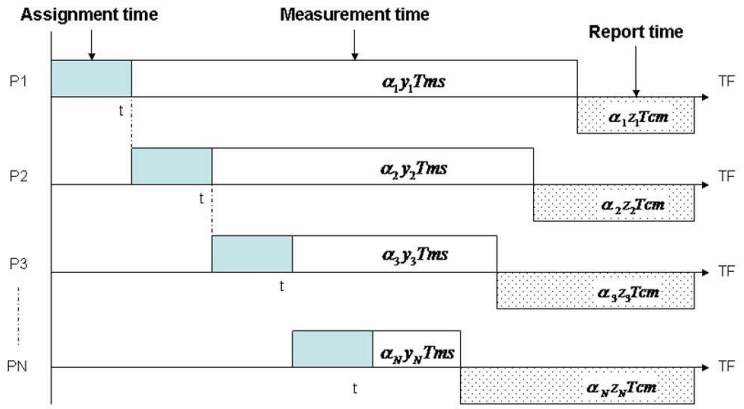

Figure 2.3: Timing diagram for a single level tree network with controller and same reporting time finalization

In Fig.(2.3) a scheduling protocol for a single level tree sensor network is considered. In this case, there is more than one channel for transmission of the data previously measured by the sensors. Consequently each sensor has an assigned channel which will be exclusively used to transmit its data measured to the control processor. Besides the multi channel availability, in this set up it is required that all processors finish at the same time. The total finish time would be described by:

$$T_1 = t + \alpha_1 y_1 T_{ms} + \alpha_1 z_1 T_{cm} \tag{2.25}$$

$$T_2 = 2t + \alpha_2 y_2 T_{ms} + \alpha_2 z_2 T_{cm} \tag{2.26}$$

$$T_3 = 3t + \alpha_3 y_3 T_{ms} + \alpha_3 z_3 T_{cm} \tag{2.27}$$

$$T_N = Nt + \alpha_N y_N T_{ms} + \alpha_N y_N T_{ms} \tag{2.28}$$

As mentioned before the finish time would be the same for each sensor. Thus,

$$T_1 = T_2 = T_2 = T_3 =, ..., = T_N \tag{2.29}$$

Using the equation in terms of the network parameters the system would look like:

$$\alpha_1(y_1 T_{ms} + z_1 T_{cm}) = t + \alpha_2(y_2 T_{ms} + z_2 T_{cm}) \tag{2.30}$$

$$\alpha_2(y_2 T_{ms} + z_2 T_{cm}) = t + \alpha_3(y_3 T_{ms} + z_3 T_{cm}) \tag{2.31}$$

$$\alpha_{N-1} = \frac{t + \alpha_N(y_N T_{ms} + z_N T_{cm})}{(y_{N-1} T_{ms} + w_{N-1} T_{cp}} \tag{2.32}$$

Another way to express the last equation is:

$$\alpha_1 = h_1 + \alpha_2 g_{(1)} \tag{2.33}$$

$$\alpha_2 = h_2 + \alpha_3 g_{(2)} \tag{2.34}$$

$$\alpha_{N-1} = h_{N-1} + \alpha_N g_{(N-1)} \tag{2.35}$$

Here

$$h_j = \frac{t}{(y_j T_{ms} + z_j T_{cm})} \tag{2.36}$$

$$g_j = \frac{(y_{j+1} T_{ms} + z_{j+1} T_{cm})}{(y_j T_{ms} + z_j T_{cm})} \tag{2.37}$$

Notice that the last set of equations (2.33-2.35) are the same obtained for the sequential reporting time protocol equations (2.11-2.14). Using the normalization equation (2.1) and following the same procedure as described for the sequential reporting time protocol we obtain the same expression for distribution of load to be measured for α_N (2.19) but instead of using f_j and s_j, the new equation would be in terms of g_j and h_j.

2.4.3 Load Pre-processing, Sequential Reporting

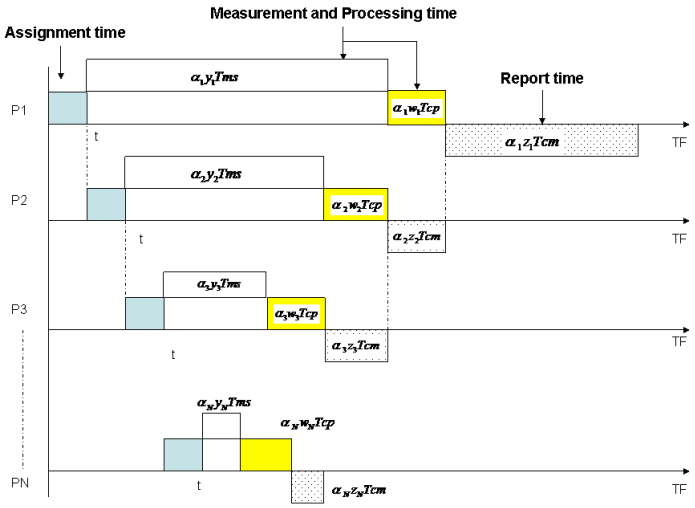

Figure 2.4: Timing diagram for a single level tree network with controller and same sequencing reporting with pre-processing

As is widely known the energy used to transmit data in wireless sensors networks is higher than the amount of energy used to compute it [17] so processing the data load measured before transmitting it is an option that has to be studied. In the following schedule protocol the same network topology used in the previous section is considered but in addition to the measuring capacity the sensors have a processing capability too. Fig.(2.4) shows how the control processor assigns load to be measured to the sensors in the network. As soon as the sensor receives its job starts to measure. The processing of the load measured starts when the sensor has finished to measure the whole fraction of load assigned. The results will be reported by each sensor sequentially.

For this protocol the expression that describes the finish time are:

$$T_1 = t + \alpha_1 y_1 T_{ms} + \alpha_1 w_1 T_{cp} + \alpha_1 z_1 T_{cm} \tag{2.38}$$

$$T_2 = 2t + \alpha_2 y_2 T_{ms} + \alpha_2 w_2 T_{cp} + \alpha_2 z_2 T_{cm} \tag{2.39}$$

$$T_3 = 3t + \alpha_3 y_3 T_{ms} + \alpha_3 w_3 T_{cp} + \alpha_3 z_3 T_{cm} \tag{2.40}$$

$$T_N = Nt + \alpha_N y_N T_{ms} + \alpha_N w_N T_{cp} + \alpha_N y_N T_{ms} \tag{2.41}$$

Following the same procedure from the first protocol studied we express the fraction of load in terms of the network parameters using the following expressions:

$$\alpha_1(y_1 T_{ms} + w_1 T_{cp}) = t + \alpha_2(y_2 T_{ms} + w_2 T_{cp} + z_2 T_{cm}) \tag{2.42}$$

$$\alpha_2(y_2 T_{ms} + w_2 T_{cp}) = t + \alpha_3(y_3 T_{ms} + w_3 T_{cp} + z_3 T_{cm}) \tag{2.43}$$

$$\alpha_{N-1} = \frac{t + \alpha_N(y_N T_{ms} + w_N T_{cp} + z_N T_{cm})}{(y_{N-1} T_{ms} + w_{N-1} T_{cp})} \tag{2.44}$$

Using the normalization equation and solving the recursive equations for α_N it is obtained the flowing expression,

$$\alpha_N = \frac{1 - \left[o_{N-1} + \sum_{j=1}^{N-2} o_j + \sum_{m=1}^{N-j-1} o_{j+m} \prod_{t=0}^{m-1} p_{j+t} \right]}{1 + p_{N-1} + \sum_{j=1}^{N-2} \prod_{l=0}^{N-j-1} p_{j+l}} \tag{2.45}$$

where

$$o_j = \frac{t}{(y_j T_{ms} + w_j T_{cp})} \tag{2.46}$$

$$p_j = \frac{(y_{j+1} T_{ms} + w_{j+1} T_{cp} + z_{j+1} T_{cm})}{(y_j T_{ms} + w_j T_{cp})} \tag{2.47}$$

As in the last sections the mathematical expression obtained for α_N in term of the variables o_j and p_j will be the same as the expression obtained in terms of f_j and s_j form the first scheduling protocol. As a consequence what makes the expressions different from each other are the parameters that defined the pair (f,s), (g,h) and (o,p) for each protocol respectively.

2.5 Performance and Evaluation Results

In order to investigate the relationship between the number of processors and the communication time in the network the expressions for minimum finish time, measuring time and reporting time were used. For each of the protocols previously presented the finish time as a function of the number

of processors in the network for a fixed value of y and changing values of z and for fixed values of z and changing values of y was simulated and plotted.

2.5.1 Sequential Reporting Distribution

Fig.(2.5) shows the finish time versus number of processors variable inverse link speed z. The inverse link speed z is evaluated between 50 and 90. The y inverse measuring speed is fixed to 190. In all the simulation the values for Tcm, Tcp and Tms are equal to one. It can be seen that the amount of processors needed to reached the minimum finish time will depend on z.

On the other hand in Fig.(2.6) the inverse link speed z is fixed to 80 and the inverse measuring speed is variated between 100 to 160. Here it can be seen that increasing the inverse measuring speed will lead us to find a better finish time for this particular example.

2.5.2 Simultaneous Reporting Distribution

The simulations presented in this section are shown in Fig.(2.7) where the inverse link speed z is evaluated between 50 and 90 and y is 190. In addition in Fig.(2.8) results for this schedule protocol with the inverse link speed fixed to 80 and inverse measuring speed y varying between 100 and 160, are shown.

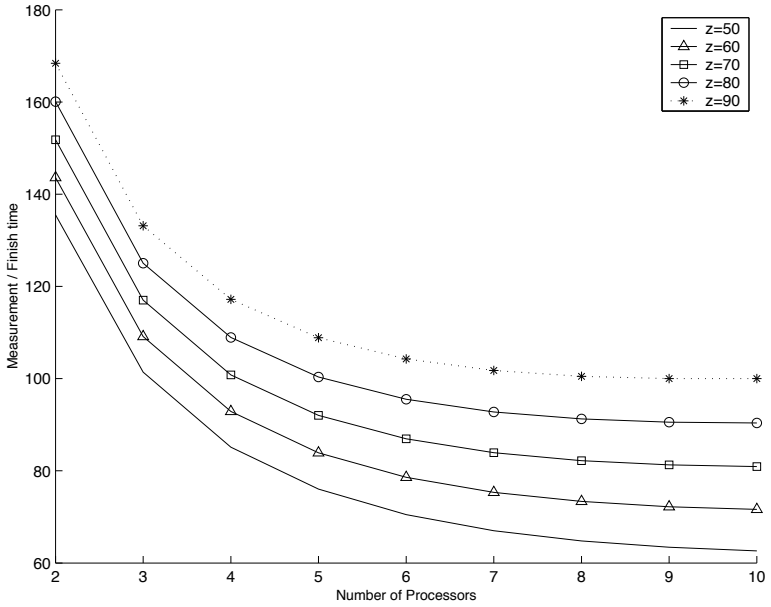

Figure 2.5: Finish time versus number of processors, variable inverse link speed z and fixed load inverse measuring speed y in a single level tree network with control processor and a sequential reporting time.

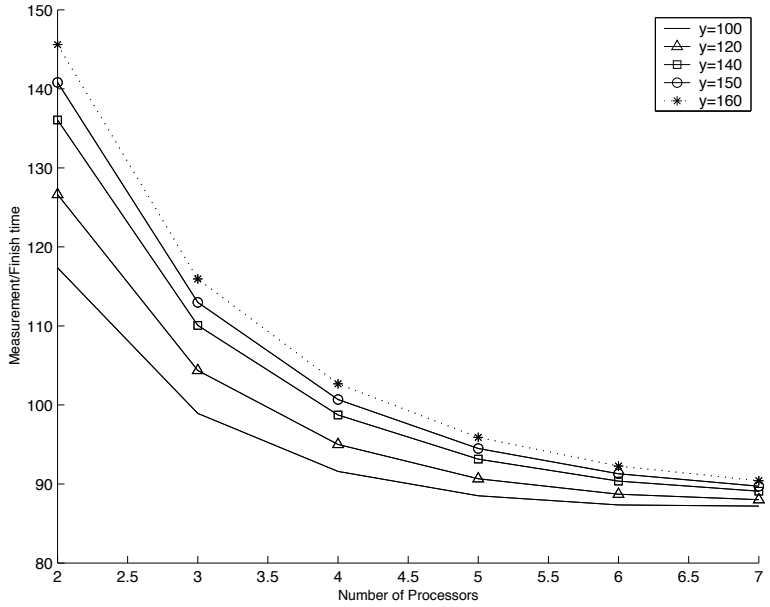

Figure 2.6: Finish time versus number of processors, variable inverse measuring speed y and fixed inverse link speed z in a single level tree network with control processor and a sequential reporting time.

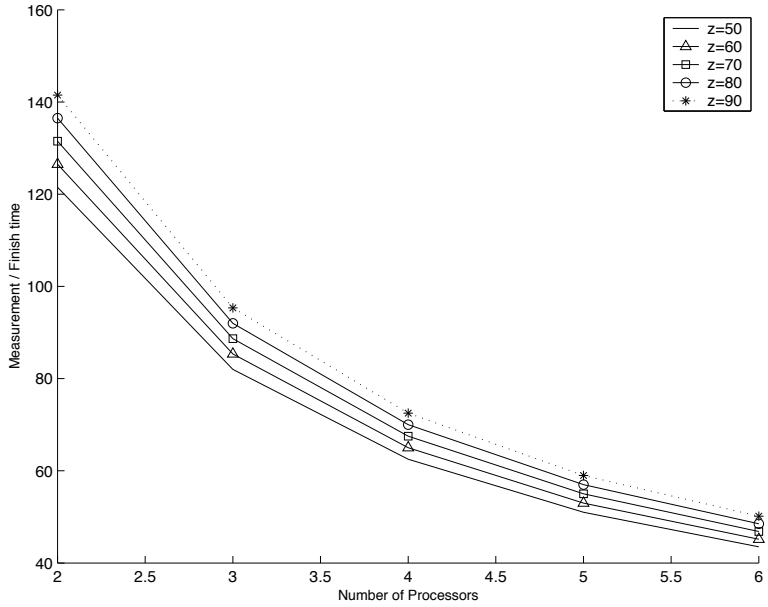

Figure 2.7: Finish time versus number of processors, variable inverse link speed z and fixed inverse measuring speed y in a single level tree network with control processor and a simultaneous reporting time.

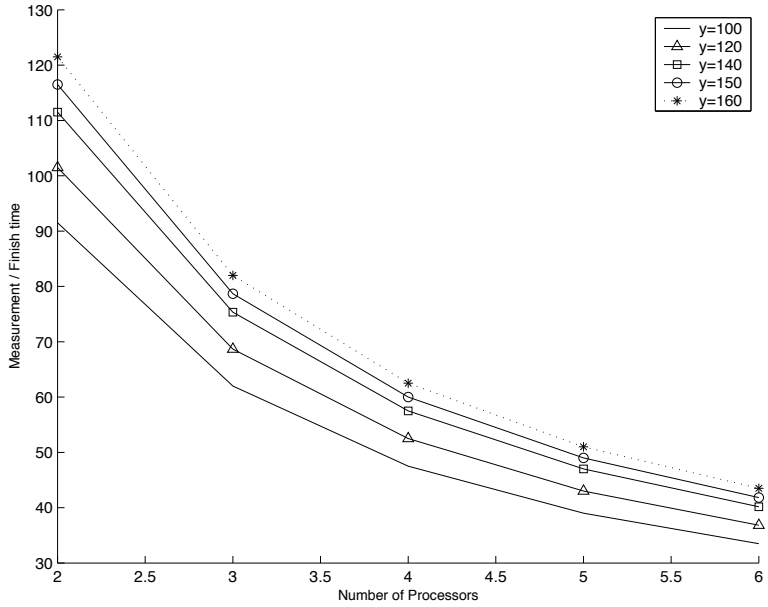

Figure 2.8: Finish time versus number of processors, variable inverse measuring speed y and fixed inverse sensor speed z in a single level tree network with control processor and a simultaneous reporting time.

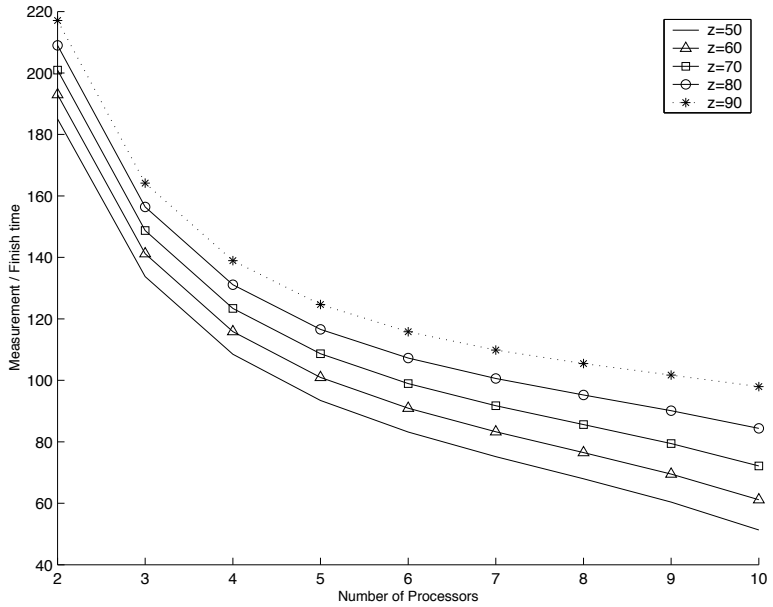

Figure 2.9: Finish time versus number of processors, variable inverse link speed z, fixed inverse measuring speed y and inverse processor speed w in a single level tree network with control processor sequential reporting with pre-processing

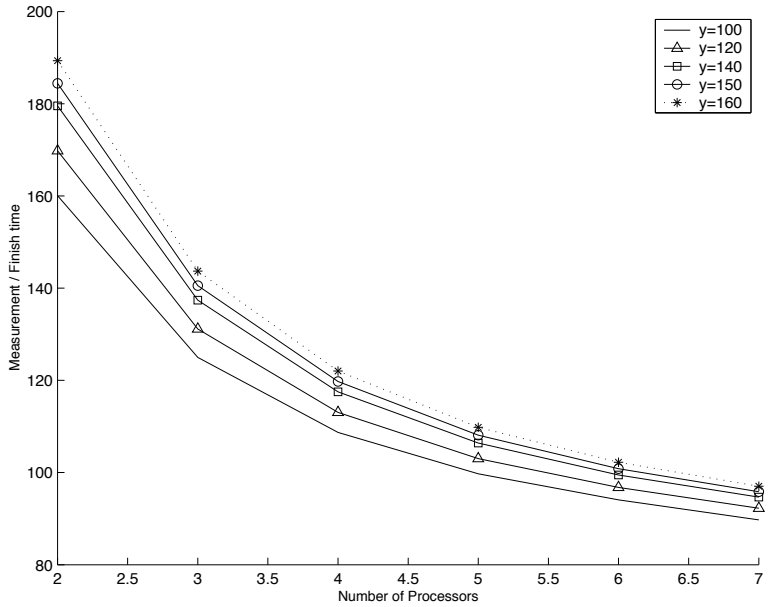

Figure 2.10: Finish time versus number of processors, variable inverse measuring speed y, fixed inverse link speed z and inverse processor speed w in a single level tree network with control processor sequential reporting with pre-processing

2.5.3 Load Pre-processing, Sequential Reporting

As mentioned before, in order to reduce the amount of energy used to report the data back to the control processor a load pre-processing can be done. In this case we include in the model the Tcp constant which will be fixed to 1. The inverse link speed varies from 50 and 90, inverse processor speed is fixed to be 100 and the inverse measuring speed is 190. Fig.(2.9) shows that for this particular example the finish time increases in comparison to the first scheduling protocol. In the case of the variation of the inverse measuring speed against the number of processor Fig.(2.10) the results showed for this particular experiment that introducing a pre-processing increases the finish time.

2.6 Conclusion

A novel closed solution for optimum finish, reporting and pre-processing time was obtained for a single level tree sensor network including immediately measuring data feature. Data and simulation were performed and analyzed for different scheduling protocols. A pre-processing scheduling strategy was proposed for a single level tree sensor network.

Chapter 3

Heuristic Optimization of Sequential Load Distribution

3.1 Summary

The sequencing problem involves optimizing the order in which a root processor should distribute divisible processing load to its children processors. The purpose of this procedure is to complete the processing of divisible load in a minimal amount of time or with a minimal energy use. Applications include third party computing services provided by Application Service Providers as well as wireless sensor networks. Four different heuristic algorithms were tested in order to achieve an efficient solution for the sequencing problem. Consequently, the optimum algorithm (Best Swap) was selected to implement the sensitivity analysis. Such analysis allowed an examination of the relationship between the total cost function, changes in the speed of the links and the processors, and changes in cost of processing and transmission.

3.2 Introduction

With the aim of introducing and reviewing the total monetary cost model on DLT this chapter presents a sensitivity analysis study performed for the total monetary cost function in terms of changes of the network parameters (link speed and processors speed) in a single level three star computer network. This study was a contribution to the research presented on [13] and will be a good reference for future studies on monetary cost and wireless sensors networks under DLT model.

From the prospective of services providers to be able to lease a high performance computer machine to allow customers (researchers, credit card companies, universities) to do processing while paying some cost. Thus developing mathematical expressions that describes the time in which the processor is busy either computing or communicating is an important and complex procedure. Studies that considered DLT model and monetary cost where presented in [11] and [15]. Research showed in [11] that there is an intrinsic relationship between the cost and the sequencing problem. This problem involves optimizing the order in which a root processor should distribute divisible processing load to its processors. The goal of this procedure is to complete the processing of divisible load in a minimal amount of time (or with a minimal energy use in wireless networks).

In this chapter the sequence problem is studied and reviewed using

divisible load scheduling theory. In particular we study sensitivity cost as a function of changes of the processors speed and link speed.

The organization for this chapter is: In section 3.3 the model description and their mathematical expressions are presented using the model presented in [6,11,15,21]. The optimization sequencing problem and the cost sensitivity analysis is discussed in section 3.4. Final results, commentaries and conclusions are described in the section 3.5 and 3.6 respectively.

3.3 Mathematics: Model, Parameters and Expressions

3.3.1 The Model

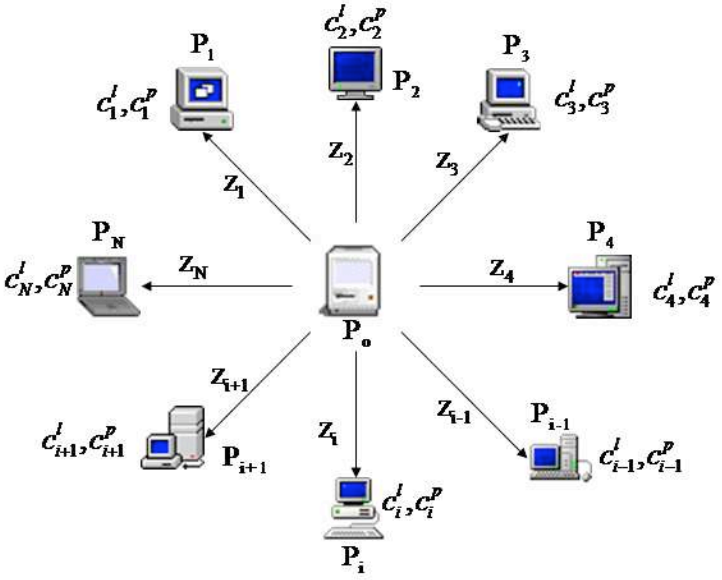

Figure 3.1: Single level three start network

The topology considered in this chapter is a single-level tree network consisting of $(N+1)$ processors and (N) links Fig.(3.1). All the processors are connected to the root processor, p_0, via communication links. The root processor is equipped with a front-end processor and it is the only processor at which the load arrives. The root processor partitions the total load into $(N+1)$ fractions, keeps its own fraction α_0, and distributes the other fractions α_1, α_2,..., α_N to the other processors p_1, p_2,..., p_N respectively and sequentially. Each processor starts to compute only after receiving all its assigned fraction of load. The linear costs c_1^l, c_2^l,..., c_N^l and c_0^p, c_1^p,..., c_N^p are the cost coefficients associated with the links and processors in the network, respectively. Where:

α_i: The load fraction assigned to the i^{th} link-processor pair.

w_i: The inverse of the computing speed of the i^{th} processor.

z_i: The inverse of the link speed of the i^{th} link.

T_{cp}: Computing intensity constant:
the entire load is processed in $w_i T_{cp}$ seconds by the ith processor.

T_{cm}: Communication intensity constant: the entire load can be
transmitted over the bus in $z_i T_{cm}$ seconds over the ith link.

T_f: The finish time: Time when the last processor ceases computation.

Our interest is to review the optimal arrangement (sequence) in which load should be distributed by the root to its processor.

3.3.2 Optimal Finish Time and the Load Distribution

In order to minimize the processing time all processors finish computing at the same time. Otherwise, the processing finish time could be reduced by transferring some fractions of load from busy processors to idle processors [6]. Nevertheless, in [6] is shown that under certain sets of network parameters, in order to minimize the processing finish time, it is not necessary that all processors be utilized. In this model it is assumed that system parameters are such that all processors in the network are utilized to compute the incoming load.

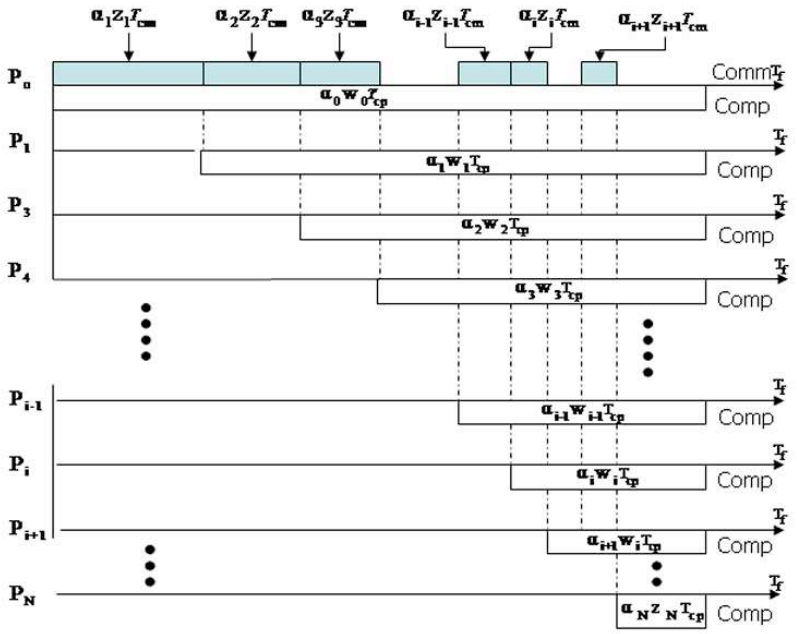

Figure 3.2: Timing diagram sequence distribution

The Gantt-chart-like timing diagram in the Fig.(3.2) represents the

process of sequential load distribution. The communication time and the computation time are shown above the time axis and below the time axis respectively. Using this timing diagram to derive the timing relationships between processors we have [21]:

$$\alpha_i w_i T_{cp} = \alpha_{i+1} z_{i+1} T_{cm} + \alpha_{i+1} w_{i+1} T_{cp} \quad i = 0, ..., N-1 \tag{3.1}$$

$$\alpha_{i+1} = k_i \alpha_i = (\prod_{j=0}^{i} k_j)\alpha_0 \quad i = 0, ..., N-1 \tag{3.2}$$

where:

$$k_i = \frac{w_i T_{cp}}{(z_{i+1} T_{cm} + w_{i+1} T_{cp})} \qquad i = 0, ..., N-1$$

Notice that all processors in the network finish at the same time and there are N equations and $N+1$ unknowns in the equation (3.1) and equation (3.2). The normalization equation is,

$$1 = \sum_{j=0}^{N} \alpha_j \tag{3.3}$$

and is used to solve this system of equations.

Using the previous system of recursive equations the expression found in [21] for α_0, α_N are:

$$\alpha_0 = \left[1 + \sum_{i=1}^{N} \left[\prod_{j=0}^{i-1} k_j\right]\right]^{-1}$$

$$= \frac{1}{D} \prod_{i=1}^{N} (z_i T_{cm} + w_i T_{cp}) \tag{3.4}$$

$$\alpha_N = \frac{1}{D} \prod_{i=0}^{N-1} (w_i T_{cp}) \tag{3.5}$$

where:

$$D = \prod_{i=1}^{N}(z_i T_{cm} + w_i T_{cp}) + \sum_{n=1}^{N}\left(\left(\prod_{i=0}^{n-1}(w_i T_{cp})\right)\left(\prod_{i=n+1}^{N}(z_i T_{cm} + w_i T_{cp})\right)\right) \quad (3.6)$$

3.3.3 Sequencing Distribution

Another interesting and fundamental concept is related with the order in which a root processor should distribute the load to be processed. This problem was treated on [21] and basically considered a single level tree network with the following ordered set:

$$\theta = p_0, (l_1, p_1), (l_2, p_2), (l_3, p_3)..., (l_{j-1}, p_{j-1}), (l_j, p_j), (l_{j+1}, p_{j+1}), ..., (l_N, p_N) \quad (3.7)$$

The first processor that is assigned a fraction of load is p_1. Such load is assigned by p_0 then this processor assigns load faction to processor p_2 and so on. The link-processor pairs is then defined as (l_j, p_j) and (l_{j+1}, p_{j+1}) and in this ordered set may not necessarily be physically adjacent. Thus, any change on the organization of the network would lead a change in the distribution of the load. As a consequence, sequencing is a technique that changes one arranged set to another ordered set. There are not physical changes in the network.

3.3.4 Link-Processor and Total Monetary Cost

The monetary cost for processing a fraction of load at any processor is defined as the cost incurred from utilizing the processor and its

corresponding link in order to process the assigned fraction of load [21]. The cost is defined only in terms of accounting for the duration during which the resource is busy serving the assigned divisible load. The cost coefficients associated with links and processors are fixed values.

The cost model parameters are defined as follows:

c_n^l: The communication cost per second of utilizing the n^{th} link.

c_n^p: The computing cost per second of utilizing the n^{th} processor.

$c_n^p w_n$: The computing cost per unit load of utilizing the n^{th} processor.

$c_n^l z_n$: The communication cost per unit load of utilizing the n^{th} link.

$(c_n^p w_n + c_n^l z_n)$: The processing cost per unit load of the n^{th} link-processor pair.

The total monetary cost is the linear summation of the individual cost incurred at each processor-link pair. This individual cost depends on the assigned fraction of load assigned to each processor. The total cost of processing an entire load is the main expression to be optimized and analyzed in terms of the sensitivity analysis.

$$C_0 = c_0^p w_0 T_{cp} \tag{3.8}$$

$$C_n = c_n^l z_n T_{cm} + c_n^p w_n T_{cp} \qquad n = 1, ..., N \tag{3.9}$$

C_n: the cost of processing the entire of load on the n^{th} processor.

$\alpha_n C_n$: the cost of processing the assigned fraction of load (α_n) on the n^{th} processor.

The total cost, C_{total}, is defined as:

$$C_{total} = \alpha_0 C_0 + \sum_{n=1}^{N} \alpha_n C_n \tag{3.10}$$

By substituting α_0 and all α_n from the equations (3.4) and (3.5) described before into equation (3.10) the total monetary cost expression is obtained explicitly in terms of the j^{th} and $(j+1)^{st}$ link-processor pairs as [21]:

$$\begin{aligned} C_{total} = \frac{1}{D} \Bigg\{ & \prod_{i=1}^{N} (z_i T_{cm} + w_i T_{cp})(c_0^p w_0 T_{cp}) \\ & + \sum_{n=1}^{j-1} \left[\left(\prod_{i=0}^{n-1} (w_i T_{cp}) \right) \left(\prod_{i=n+1}^{N} (z_i T_{cm} + w_i T_{cp})(c_n^l z_n T_{cm} + c_n^p w_n T_{cp}) \right) \right] \\ & + \prod_{i=0}^{j-1} (w_i T_{cp}) \prod_{i=j+1}^{N} (z_i T_{cm} + w_i T_{cp})(c_j^l z_j T_{cm} + c_j^p w_j T_{cp}) \\ & + \prod_{i=0}^{j} (w_i T_{cp}) \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp})(c_{j+1}^l z_{j+1} T_{cm} + c_{j+1}^p w_{j+1} T_{cp}) \\ & + \sum_{n=j+2}^{N} \left[\prod_{i=0}^{n-1} (w_i T_{cp}) \prod_{i=n+1}^{N} (z_i T_{cm} + w_i T_{cp})(c_n^l z_n T_{cm} + c_n^p w_n T_{cp}) \right] \Bigg\} \end{aligned} \tag{3.11}$$

Where D is:

$$D = \prod_{i=1}^{N} (z_i T_{cm} + w_i T_{cp}) + \sum_{n=1}^{j-1} \left(\prod_{i=0}^{n-1} (w_i T_{cp}) \prod_{i=n+1}^{N} (z_i T_{cm} + w_i T_{cp}) \right)$$

$$+\prod_{i=0}^{j-1}(w_iT_{cp})(z_{j+1}T_{cm}+w_{j+1}T_{cp})\prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp})$$
$$+\prod_{i=0}^{j-1}(w_iT_{cp})(w_jT_{cp})\prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp})$$
$$+\sum_{n=j+2}^{N}\left(\left(\prod_{i=0}^{n-1}(w_iT_{cp})\right)\left(\prod_{i=n+1}^{N}(z_iT_{cm}+w_iT_{cp})\right)\right) \tag{3.12}$$

3.4 Monetary Cost Optimization

3.4.1 Optimization Sequencing Problem

The heuristic algorithms used here will, at each iteration, swap the logical position of two processors which are logically adjacent in the load distribution sequence if it leads to a cost improvement. Four different algorithms were tested in order to solve the sequencing problem which involves optimizing the order in which a root processor should distribute divisible processing load to its children in order to complete the processing of divisible load in a minimal amount of time or with a minimal energy [13]:

(1) **Adjacent Swap algorithm**: Swaps adjacent link processor pairs until an improvement is greater than a threshold is reached.

(2) **Best Swap algorithm**: Chooses the best swap between all possible swaps in the network (N Combined 2).

(3) **Random Swap algorithm**: Picks two random nodes and swaps them

if and only if such swap yields a cost improvement.

(4) **First Swap algorithm**: Deterministically uses the first possible swap if such swap yields an improvement in the cost function and keeps trying all the possible swaps until the threshold value prevents it from obtaining more improvement in the cost function.

To choose a heuristic algorithm several tests were made using the algorithms described before. In Fig.(3.3) the three first levels represent the methodology used to perform the different tests using the these algorithms. Random values were chosen for the network parameters which were constrained to have communication time less than computation time. Consequently, a heterogeneous network was used to compare the algorithms.

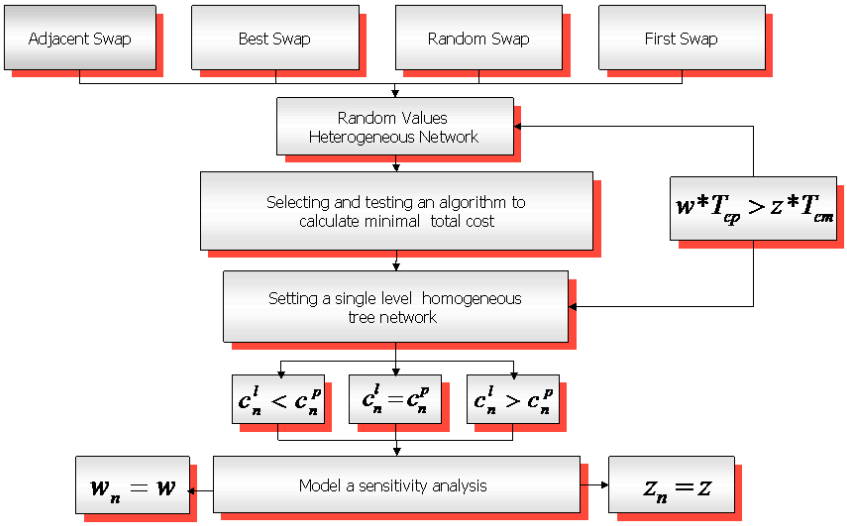

Figure 3.3: Methodology of the sensitivity analysis performed

3.4.2 Cost Sensitivity Analysis

To understand how the changes in the processor speed and link speed affect the total cost in a single level tree network we implemented a sensitivity analysis. In such a topology the speed of processors is equal for all processors and all links speeds are identical. That is w_i=w and z_i=z respectively.

For w the sensitivity expression is:

$$Sensitivity(w) = \frac{C_{total}(w_{nom}) - C_{total}(w_{\text{var}})}{w_{nom} - w_{\text{var}}} \tag{3.13}$$

For z the sensitivity function is:

$$Sensitivity(z) = \frac{C_{total}(z_{nom}) - C_{total}(z_{\text{var}})}{z_{nom} - z_{\text{var}}} \tag{3.14}$$

Where:

w_{nom}: Nominal value of the processor speed to be compared.

z_{nom}: Nominal value of the link speed to be compared.

w_{var}: Changed value of the processor speed to be compared.

z_{var}: Changed value of the link speed to be compared.

In a homogeneous bus network the optimal distribution sequence is where load distribution is in non-decreasing order of the sum of the link and processor costs.

The sensitivity analysis was studied in three situations where the cost of the link and the processor is $c_n^l < c_n^p$, c_n^l=c_n^p, $c_n^l > c_n^p$.

The sensitivity analysis was implemented using the more efficient algorithm in order to achieve the optimal distribution sequence. This algorithm is presented in the next section.

3.5 Results and Commentaries

3.5.1 Algorithm Selected

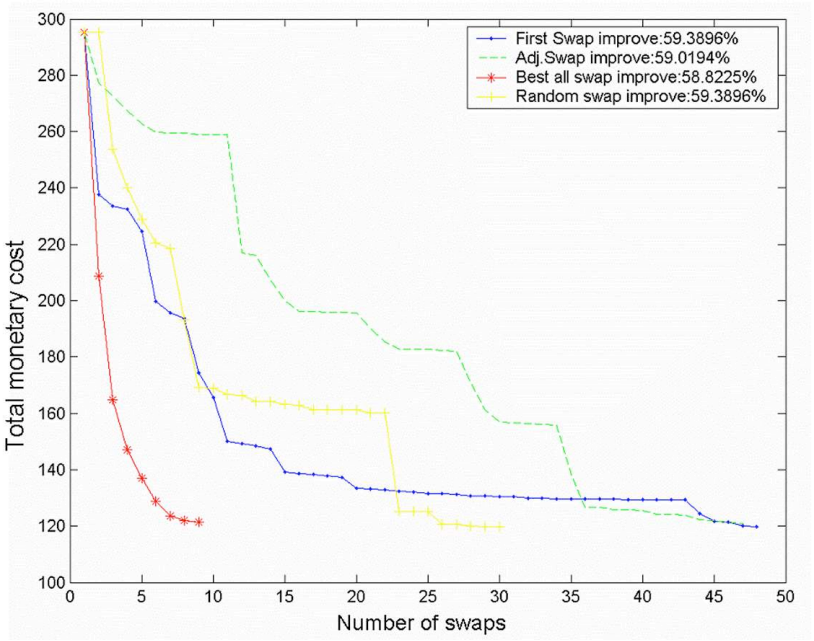

Figure 3.4: Simulation results for heuristic algorithms tested

Fig.(3.4) depicts a test for a 10 children network. The Best Swap algorithm proved to be a strong and consistent algorithm in finding an optimal solution in all the simulations with an almost vertically descending cost function as it is shown. For this particular example the numbers of swaps is presented in Table 1:

Algorithm	Improvement(%)	Swaps
Adjacent Swap	59.0194	47
Best Swap	58.8225	9
Random Swap	59.3896	30
First Swap	59.3896	48

Table 3.1: Results improvement and numbers of swaps, heuristic algorithms tested

For the other three algorithms it was found that they have almost the same improvement but that take a larger number of swaps.

3.5.2 Sensitivity Analysis Results

Due to its effectiveness the Best Swap algorithm was used to implement the sensitivity analysis.

For homogeneous network configuration with 10 children processors the parameters used here were:

w_{nom}: 20.

z_{nom}: 10.

w_{var}: [11...40].

z_{var}: [1...10].

Random values were chosen for the cost of link and the processor. However, three set of cost processors and cost of the links were constrained to $c_n^l < c_n^p$, $c_n^l = c_n^p$, and $c_n^l > c_n^p$. Thus, three configurations of the network for those processor and link costs were studied.

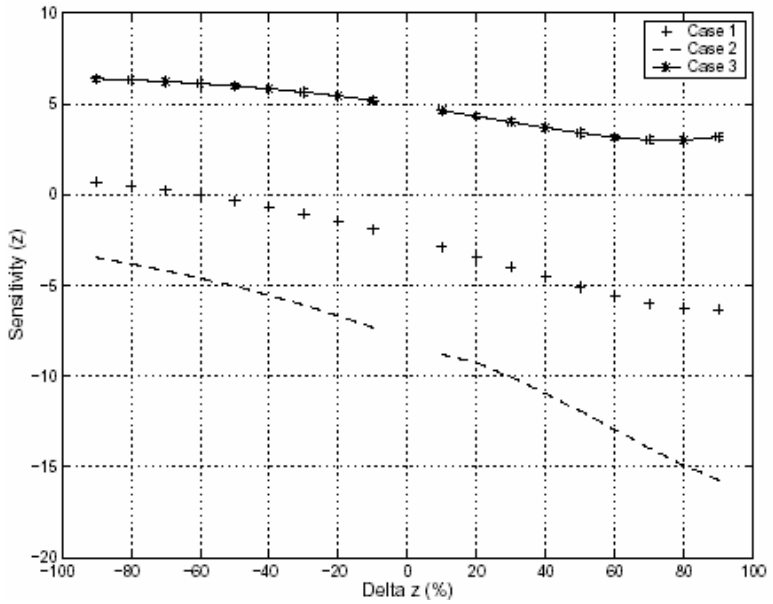

Figure 3.5: Sensitivity of monetary cost to link speed change

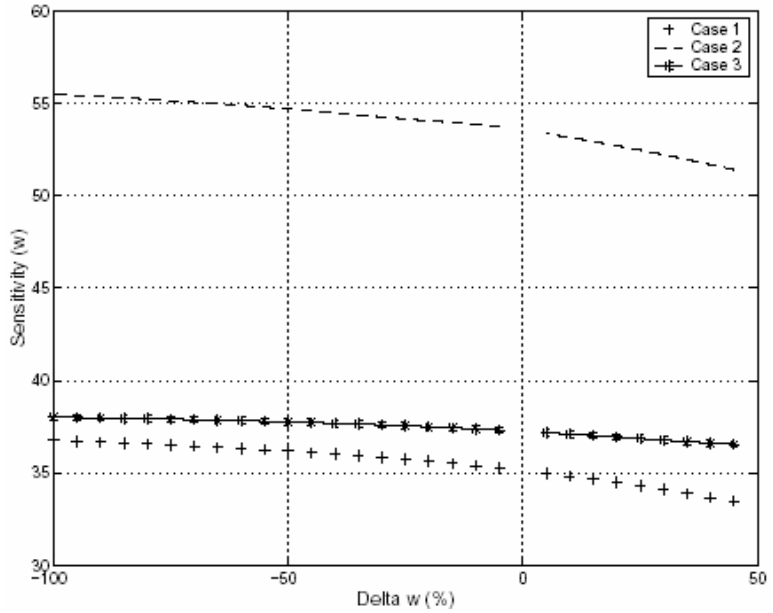

Figure 3.6: Sensitivity of monetary cost to processor speed change

In Fig.(3.5) the result of the analysis of sensitivity for z is shown. Notice that when $c_n^l < c_n^p$ the total cost function has a decreasing behavior. Otherwise, when $c_n^l > c_n^p$ the total function cost is increasing and this is reflected in the sensitivity axis.

In Fig.(3.6) we present the results for the computation of sensitivity when speed processor is varied. In this figure is evident that for all the w variations in the three configurations for the of cost of the processors the and link have positive values for the sensitivity axis.

The behavior of the sensitivity analysis for the total monetary cost presented here is for specific values of parameters. That is, this is not a general result for all possible values of the parameters. In addition this behavior is due to the complexity of the expression defined for (3.5) and its relationship with the total cost (3.11). Hence, results obtained in this study show some of the representative curves of sensitivity of a single level tree star network with specific network parameters.

3.6 Conclusions

Based on the result the Best Swap algorithm was found to outperform the other algorithms presented. It was found that all algorithms improved the cost function in some cases obtaining a cost improvement of more than 50%.

It was shown that only when $c_n^l > c_n^p$, the total cost function for this configuration network increases in proportion to changes in w and z.

Chapter 4

Optimizing a Divisible Load Nonlinear Cost Function

4.1 Summary

The behavior of load distribution for loads with nonlinear monetary computing cost is studied.

4.2 Introduction

Companies are using third party machines as a result of a corporate interest to create computer utilities for leasing. This tendency may lead researchers and developers to use high performance parallel devices and algorithms while paying some monetary charge. The monetary cost associated with the production and operation of such machines leads to a requirement to lease their processing in a time efficient manner. Indeed, there is an intrinsic relationship between the cost and the sequencing problem [21]. This problem involves optimizing the order in which a root processor should

distribute divisible processing load to its processors. In [22] the scheduling of nonlinear loads was studied. In this paper we explore the monetary cost as a function of nonlinear processing loads.

4.3 The Model

Consider a single level tree (star) network consisting of N+1 processors and N links which have related corresponding computation c_n^p and communication cost c_n^l. Here the c_n's are linear monetary cost coefficients. The root processor will receive all the load and distribute to each child processor their assigned fraction of load sequentially.

4.3.1 Notations and Definitions

α_i:The load fraction assigned to the i^{th} link-processor pair.

w_i: The inverse of the computing speed of the i^{th} processor.

z_i: The inverse of the link speed of the i^{th} link.

T_{cp}: Computing intensity constant: the entire load is processed in $w_i T_{cp}$ seconds by the ith processor.

T_{cm}: Communication intensity constant: the entire load can be transmitted over the bus in $z_i T_{cm}$ seconds over the ith link.

4.4 Total Monetary Cost

Total monetary cost is the linear summation of the individual costs incurred at each processor-link pair. This individual cost depends on the assigned fraction of load. The monetary cost expression found in [21] was modified expressing the load as a function of β, equation(1). Here β is a parameter needed to express the nonlinearity characteristic of the load.

$$C_{total} = \alpha_0^{\beta} c_0^p w_0 T_{cp} + \sum_{n=1}^{N} \alpha_n^{\beta} (c_n^p w_n T_{cp} + c_n^l z_n T_{cm}) \tag{4.1}$$

In order to get a solution for the sequencing problem the Best Swap Algorithm presented in [13] was chosen due to its consistency in finding an optimal solution and was modified to include the β parameter.

4.5 Results and Commentaries

A heterogeneous single level tree network with N=4 processors was modeled. Random values were chosen and fixed for different parameters in the network. An algorithm was run changing the β parameter (β>1 and β<1). The distribution of load in the network may explain the total monetary cost behavior. In Fig. (4.1) is shown that for large values of β an almost equal load distribution is generated. As consequence, it is expected that the minimum total monetary cost grows with an almost constant rate for large values of β. It was found that for values of β approaching 0 most of the load is assigned to the processor (P3).

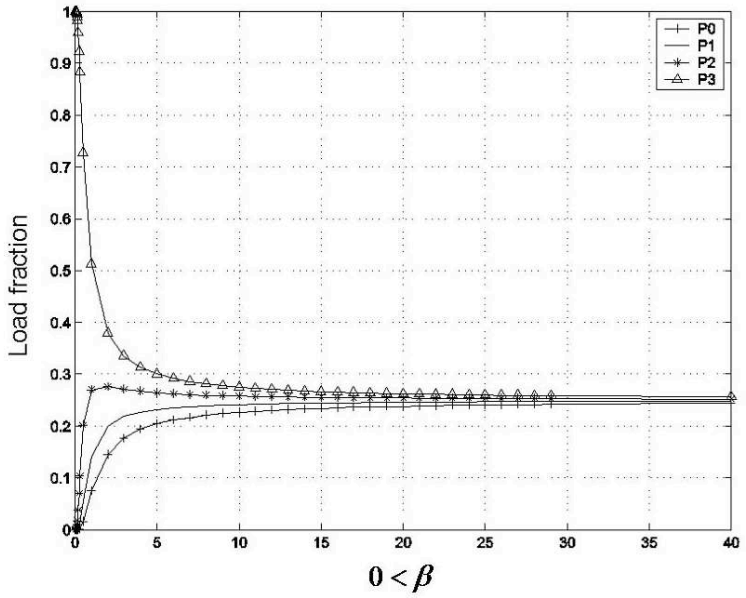

Figure 4.1: Load distribution for a single level tree network Beta >0

4.6 Conclusions

This load distribution behavior is related to the convexity and concavity of (4.1) for different values of beta.

Chapter 5

Adaptive multi scheduling protocol for divisible loads on wireless sensor networks

5.1 Summary

This chapter considers an innovative scheduling mechanism to assign divisible load jobs to wireless sensors in a single level tree network. The schedule protocol presented here takes into account the amount of load processed from previous load assignments based on distributing the new incoming load to the sensor network aiming to minimize the total finish time of processing the entire Divisible Load Job Task (DLJT) and reducing the idle state of each processor. Here a DLJT is defined as the total addition of individual loads that share a common characteristic and belong to a data set. Each element of the DLJT can be arbitrarily partitioned and distributed by the root processor to the rest of the wireless sensors.

5.2 Introduction

Potential applications of wireless sensor networks require the design of a robust data acquisition strategy that take into account possible sensor failure and reacts to it to minimize the risk of losing data samples or measurements while minimizing energy. Practical situations are envisioned where the numbers of nodes that belong to a WSN are small and the entire performance of the network is affected if an inadequate performance in a node occurs while processing load or acquiring data measurements.

Until now the schedule protocols on wireless sensors networks using DLT have assumed that the nodes involved in the processing of the load can successfully finish processing the entire load assigned by the root processors under different controlled conditions scenarios. Thus in the previous chapters the study of different scheduled protocols addressed different distributions of load on sensor networks assuming successful functioning of the node while processing the load assigned in order to estimate the finish time.

In this chapter a methodology is proposed to overcome possible scenarios when a node on the network fails to process its assigned fraction of load, compromising the entire DLJT execution on the network. A technique to find the best finish time solution is proposed for a constant DLJT arrival rate. In addition a particular scenario is studied when the arrival load of each element of the DLJT rate is constant and one of the sensors on the

network can not finish processing the previous load. This is a very practical situation as predictions on available processor and link effort may diverge from reality as time proceeds resulting in deviations in the timing of earlier implemented schedules. As a result, in this paper a heuristic algorithm is developed to estimate the total finish time of processing the DLJT.

5.3 The Divisible Load Job Task based reference model

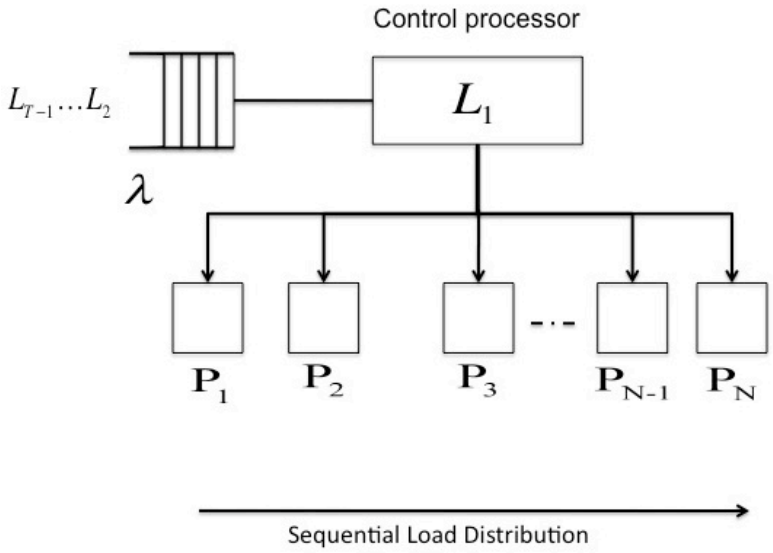

Figure 5.1: General DLJT's system

The divisible load job task reference model is presented Fig.(5.1). The DLJT system consists on a central root processor with buffer capabilities linked to a set of sensors in a single level tree network topology. Equation (5.1) describes the DLJT incoming task (DLJT) assigned to this system

(e.g. the sensors are asked to acquired data samples of the temperature under certain conditions such relative humidity). The task assigned to the root processor is composed by individual load/job assignments (L), each one arriving with of λ rate and the size of this task is T. Under this scenario it is assumed that the network will be available for processing incoming load/jobs so that there is no overlapping among jobs. The L load/job can be arbitrary partitioned, hence, the control processor will distribute the L load/job to its children processors sequentially. The goal here is to process different fraction of load/job assigned to the sensors while minimizing the finish time. Thus all processors need to finish at the same time [5].

$$DLJT = \sum_{jt=1}^{T} L_j \tag{5.1}$$

Fig.(5.2) shows the Gantt-chart like timing diagram for this DLJT system for a constant arrival rate λ and sequential distribution The mathematical expressions that describe the total finish time of a DLJT task is shown in equation (5.2)

$$DLJT_{FT} = T_{f1}(L_1, \lambda_1)+T_{f2}(L_2, \lambda_2)+....+T_{f(T-1)}(L_{T-1}, \lambda_{T-1})+T_{f(T)}(L_T, \lambda_T) \tag{5.2}$$

The time that the load/job spends on the network can be found using the classical Divisible Load Theory equations for a sequential distribution and simultaneous finish time. In [6] it was proved that the optimal solution

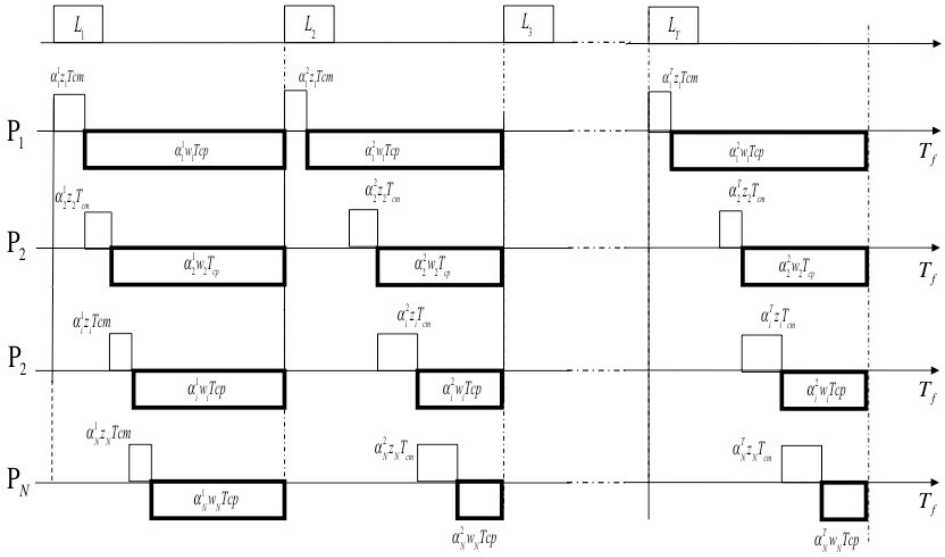

Figure 5.2: DLJT Load distribution finish time diagram

was obtained when all processors finish at the same time. T_f denotes the finish time.

$$T_f(\lambda, L_T, P_1) = T_f(\lambda, L_T, P_2) \tag{5.3}$$

$$T_f(\lambda, L_T, P_2) = T_f(\lambda, L_T, P_3) \tag{5.4}$$

$$T_f(\lambda, L_T, P_{N-1}) = T_f(\lambda, L_T, P_N) \tag{5.5}$$

Thus analyzing Fig.(5.2) for a particular finish time $T_{f(T)}(L_T, \lambda_T)$ a set of N-1 equations can be derived and are described as follows:

$$\alpha_1^T w_1 T_{cp} = \alpha_2^T w_2 T_{cp} + \alpha_2^T w_2 T_{cp} \tag{5.6}$$

$$\alpha_2^T w_2 T_{cp} = \alpha_3^T w_3 T_{cp} + \alpha_3^T w_3 T_{cp} \tag{5.7}$$

$$\alpha_{i-1}^{T} w_{i-1} T_{cp} = \alpha_{i}^{T} w_{i} T_{cp} + \alpha_{i}^{T} w_{i} T_{cp} \tag{5.8}$$

$$\alpha_{N}^{T} w_{N} T_{cp} = \alpha_{N}^{T} w_{N} T_{cp} + \alpha_{N}^{T} w_{N} T_{cp} \tag{5.9}$$

Equation (5.10) represents the relationship of the total amount of load/job, L_T, that had arrived to the control processor and the individual partitions that the control processor distributes to its children processors.

$$\alpha_{1}^{T} + \alpha_{2}^{T} + \alpha_{3}^{T} + \cdot + \alpha_{i}^{T} + \cdot + \alpha_{N-1}^{T} + \alpha_{N}^{T} = L_T \tag{5.10}$$

The previous equation system can be expressed in terms of s_j as,

$$\alpha_1 = \alpha_2^T s_{(1)} \tag{5.11}$$

$$\alpha_2 = \alpha_3^T s_{(2)} \tag{5.12}$$

$$\alpha_i = \alpha_{i+1}^T s_{(i)} \tag{5.13}$$

$$\alpha_{N-2} = \alpha_{N-1}^T s_{(N-2)} \tag{5.14}$$

$$\alpha_{N-1} = \alpha_N^T s_{(N-1)} \tag{5.15}$$

Where

$$s_i = (w_{i+1} T_{cp} + z_{i+1} T_{cm}) / w_i T_{cp} \tag{5.16}$$

Using the equation system previously presented and the equation (5.10) the expression for the load assigned to the first processor can be found as,

$$\alpha_1^T(\lambda, L_T) = \frac{L_T}{1 + \sum_{i=1}^{N-1} \prod_{j=1}^{i} \frac{1}{S_{(j)}}} \tag{5.17}$$

For a particular load/job assignment for processor one (P_1) the corresponding finish time will be,

$$T_f(L_T(\lambda_T), P_1) = (Tcp \cdot w_1 + Tcm \cdot z_1)\alpha_1^T(L_T(\lambda_T)) \tag{5.18}$$

In general equation (5.18) can be expressed in terms the optimal fraction of load/job assignment. Thus equation (5.19) and equation (5.20) represents the total finish time of a DLJT when assigned to this network topology.

$$DLJT_{FT} = \sum_{jt=1}^{T} Tf_{jt}(L_{jt}(\lambda_{jt})) \tag{5.19}$$

$$DLJT_{FT} = (Tcp \cdot w_1 + Tcm \cdot z_1) \sum_{jf=1}^{T} \frac{L_{jt}(\lambda_{jt})}{1 + \sum_{i=1}^{N-1} \prod_{j=1}^{i} \frac{1}{S_{(j)}}} \tag{5.20}$$

The control processor will check status of the individual load/job assigned to its sensor at finish time prior distributing the next L load/job to the sensor network.

5.4 The Divisible Load Job Task the adaptive model

In the previous section DLJT batch jobs are assigned to the network under the assumption that not there is not a overlapping among DLJT jobs. This implied that the network was able to serve or resolve a DLJT task before an incoming one was assigned to the network. In this section two different cases are studied, considering the DLJT task being processed on the network and the incoming one arriving to be served.

5.4.1 CASE 1: Allocation time of $L_T(\lambda_t)$ load/job smaller or bigger than the finish time $Tf(L_T(\lambda_{t-1}))$

Fig.(5.3) shows the Gantt-chart diagram for a DLJT task when a delay or shift occurs. On a real system implementation this delay or shift can be generated by failure on the system transmitting L load/jobs from a DLJT task to the control processor. In this case instead of retransmitting the entire DLJT set of load/job L_T the total finish time of processing the entire set will take into account this delay or shift. Thus the finish time for a DLJT when a load is L_T delayed compared to the resolution time on the network of the load/job L_{T-1} can be found as

$$DLJT_{delayed} = DLJT_{FT} + T_F delay_M \tag{5.21}$$

Using equation (5.19) the total finish time of a DLJT delayed is shown in the following equation, where M represents the total number of delays

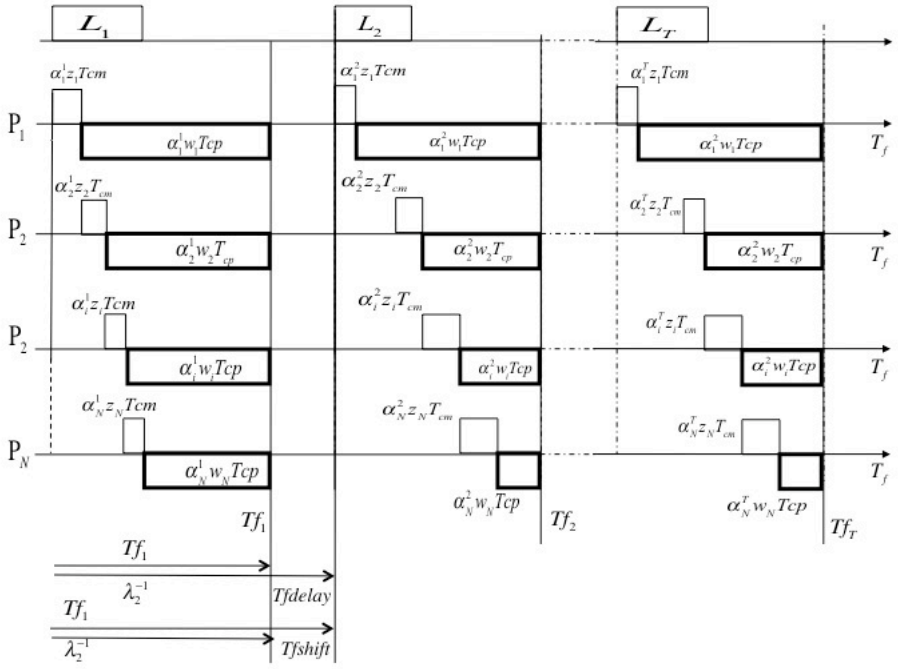

Figure 5.3: DLJT Load distribution finish time diagram including arrival delay

per run.

$$DLJT_{delayed} = \sum_{jt=1}^{T} Tf_{jt}(L_{jt}(\lambda_{jt})) + \sum_{m=1}^{M} (Tf(L_m(\lambda_{m-1})) - \lambda_m^{-1}) \quad (5.22)$$

Another situation can occur when for the host submitter of L_T load/job, a particular DLJT task is sent to the control processor before the previous L_{T-1} and the network finishes its entire processing. In this case the control processor will queue the L_T until the network is available for processing. The calculation of this network delay will define the proper shift time. Here the shift time is defined as the amount of time of a load/job has to wait at

the control process until is served by the network. This time can be found using the equation presented below

$$DLJT_{shifted} = \sum_{jt=1}^{T} Tf_{jt}(L_{jt}(\lambda_{jt})) + \sum_{m=1}^{M} (\lambda_m^{-1} - Tf(L_m(\lambda_{m-1}))) \quad (5.23)$$

Here M is the number of delays or shifts per DLJT task. M satisfies the following constraint:

$$0 \leq M \leq N - 1 \quad (5.24)$$

where $N = T$, the size of DLJT task.

In the particular case when there is a homogeneous L job/task assignment from a DLJT set the previously presented equations (5.22) and (5.23) can be expressed respectively as

$$DLJT_{delayed} = M \cdot Tfdelay + N \cdot Tf \quad (5.25)$$

and

$$DLJT_{shifted} = M \cdot Tfshift + N \cdot Tf \quad (5.26)$$

5.4.2 CASE 2: Adapted DLJT scheduled protocol for a partial sensor failure

On wireless sensor networks retransmission of data is an expensive operation. Retransmission of data could be caused by a failure of a node

to process its particular load/job α_i^T. Being able to resume operation when a wireless sensor node fails partially while avoiding retransmission of load due to this failure will lead to a mitigation of the transmission energy and increase of the utilization of the network. In order to do so the following methodology is proposed to handle this partial failure on one node of the wireless sensor network.

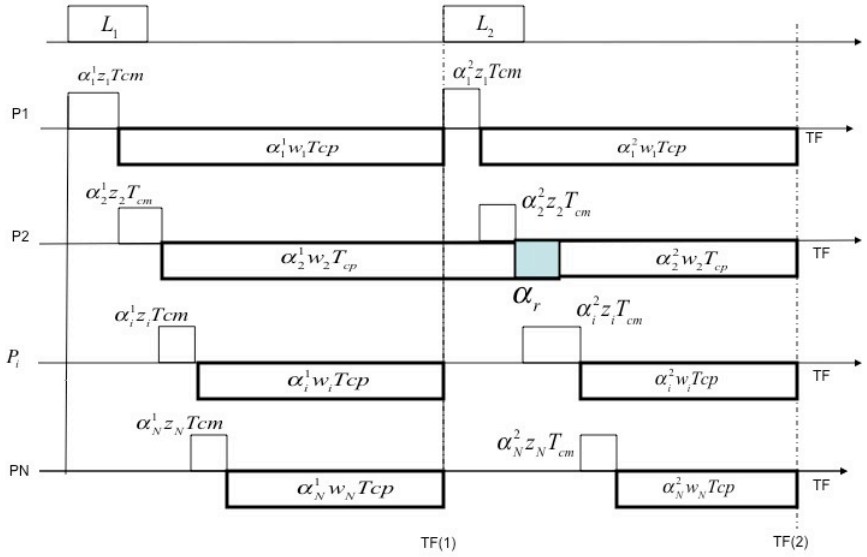

Figure 5.4: DLJT Load distribution finish time diagram one node partial failure

In Fig.(5.4) is shown that sensor P2 fails partially when processing its α_2^1 assignation here where the subindex shows the processor number and the super index specifies the L load/job assignment on the network. Since with this wireless sensor network technology, transmitting and processing data at the same time is avoided due to lack of front end processors, the

sensor has two possible operational states: receving state (communication) or processing state (never both at the same time). The consequence of this failure is reflected on the next allocation of load/job L_2 since the second processor is still processing load/job from the previous L_1 allocation and incoming load can not be allocated. Thus a remainder load α_r would still need to be processed in the L_2 load/job while avoiding delay the entire L_2 load distribution. A possible simple solution will be uses similar strategy presented before in Case 1, buffer the L load/job and shift it distribution a more interesting approach would be to been able to schedule the incoming load (even if the temporal failure is detected) and distribute the incoming L load/job taking into account these excess of load on the failing processor Fig.(5.5) shows how this can be achieved.

Using Fig.(5.4) a set of equation can be derived to calculate the new allocation of load which will take into account the remainder load on the failed processor.

The new finish time will take into account this delay of processing the failure on per a processor basis. Thus the delay is assigned to the finish time equation of the failed node. This situation is described below when processor number 2 fails partially to process the entire load:

The optimal finish time can be found using the same methodology as shown [6] and it can be express using equation (5.18). Thus,

$$Tf(L_2(\lambda_1), P_1) = (Tcp \cdot w_1 + Tcm \cdot z_1)\alpha_1^2(L_2(\lambda_1)) \qquad (5.27)$$

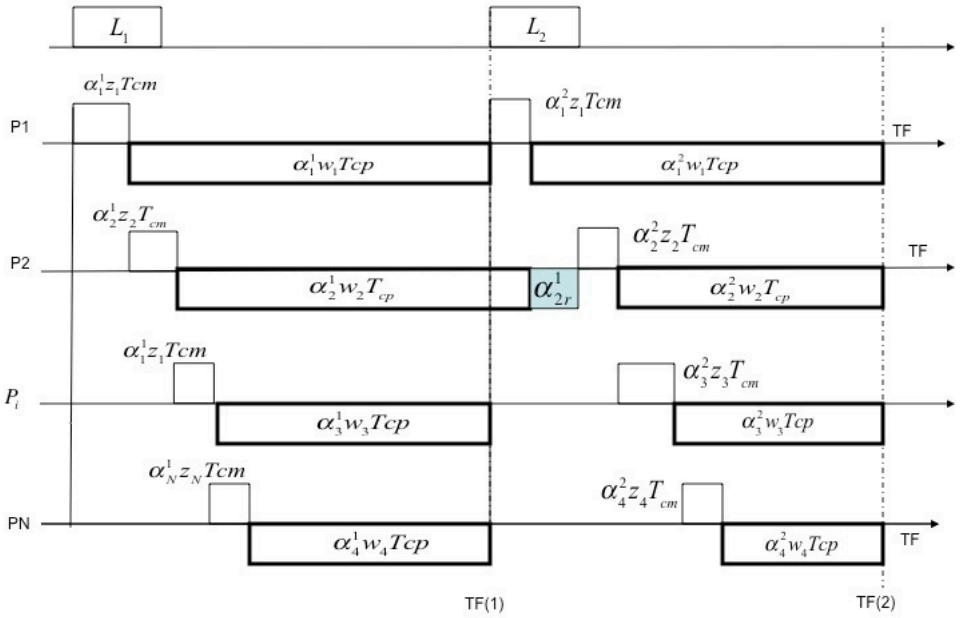

Figure 5.5: DLJT Load distribution finish time diagram one node partial failure

The corresponding new fractions of load/job assigned to each processor can be found using the following set of equations

$$\alpha_1^2 w_1 T_{cp} = \alpha_{2r}^1 w_2 T_{cp} + \alpha_2^2 z_2 T_{cp} + \alpha_2^2 w_2 T_{cp} \tag{5.28}$$

$$\alpha_2^2 w_2 T_{cp} = \alpha_3^2 z_3 T_{cp} + \alpha_3^2 w_3 T_{cp} \tag{5.29}$$

$$\alpha_3^2 w_3 T_{cp} = \alpha_4^2 z_4 T_{cp} + \alpha_4^2 w_4 T_{cp} \tag{5.30}$$

The previous equation system can be expressed in terms of s_j.

$$\alpha_1^2 = \alpha_2^2 s_{(1)} + f_{(1)} \tag{5.31}$$

$$\alpha_2^2 = \alpha_3^2 s_{(2)} \tag{5.32}$$

$$\alpha_3^2 = \alpha_4^2 s_{(4)} \tag{5.33}$$

Equation (5.60) is used to find the corresponding values for s_j . In this case a new parameter is introduced f_j ,

$$f_1 = (\alpha_{2r}^1 w_2 T_{cp} T_{cp})/w_1 T_{cp} \tag{5.34}$$

This parameter includes the amount of load that still needs to be processed when there is a partial failure on sensor 1.

The corresponding normalization equation (5.35) for this example is

$$L_2 + \alpha_{2r}^1 L_1 = \sum_{i=1}^{4} \alpha_i^2 \tag{5.35}$$

Using this equation system the α_1^2 load/job fraction assigned for this processor can be find as:

$$\alpha_1^2 = \frac{L_2 + \alpha_{2r}^1 L_1 + f_1(\sum_{i=1}^{3} \prod_{j=1}^{i} \frac{1}{s_j})}{1 + \sum_{i=1}^{3} \prod_{j=1}^{i} \frac{1}{s_j}} \tag{5.36}$$

And the α_4^2 load/job fraction assigned is:

$$\alpha_4^2 = \frac{L_2 + \alpha_{2r}^1 L_1 - f_1(\sum_{i=1}^{3} \prod_{j=1}^{i} s_j)}{1 + \sum_{i=1}^{3} \prod_{j=1}^{i} s_j} \tag{5.37}$$

Notice when there is no load remaining to be processed, equation(5.37) is similar to equation (5.17).

When a different node fails for example node 3 it can be shown that, on this exercise, the solution for α_1 and α_4 the will depend on the following equation system:

$$\alpha_1^2 = \alpha_2^2 s_{(1)} \tag{5.38}$$

$$\alpha_2^2 = \alpha_3^2 s_{(2)} + f_{(2)} \tag{5.39}$$

$$\alpha_3^2 = \alpha_4^2 s_{(4)} \tag{5.40}$$

Where

$$f_2 = (\alpha_{3r}^1 w_3 T_{cp})/w_2 T_{cp} \tag{5.41}$$

And the solution for α_1 and α_4

$$\alpha_1^2 = \frac{L_2 + \alpha_{3r}^1 L_1 + f_2(\sum_{i=2}^{3} \prod_{j=1}^{i} \frac{1}{s_j})}{1 + \sum_{i=1}^{3} \prod_{j=1}^{i} \frac{1}{s_j}} \tag{5.42}$$

and the α_4^2 load/job fraction assigned is:

$$\alpha_4^2 = \frac{L_2 + \alpha_{3r}^1 L_1 - f_2(\sum_{i=2}^{3} \prod_{j=1}^{i} s_j)}{1 + \sum_{i=1}^{3} \prod_{j=1}^{i} s_j} \tag{5.43}$$

The same can be shown when processor 4 fails,

$$\alpha_1^2 = \alpha_2^2 s_{(1)} \tag{5.44}$$

$$\alpha_2^2 = \alpha_3^2 s_{(2)} \tag{5.45}$$

$$\alpha_3^2 = \alpha_4^2 s_{(4)+f_{(3)}} \tag{5.46}$$

The corresponding set of equations will be

Where

$$f_2 = (\alpha_{4r}^1 w_3 T_{cp})/w_2 T_{cp} \tag{5.47}$$

and the solution for α_1 and α_4

$$\alpha_1^2 = \frac{L_2 + \alpha_{4r}^1 L_1 + f_3(\sum_{i=3}^{3} \prod_{j=1}^{i} \frac{1}{s_j})}{1 + \sum_{i=1}^{3} \prod_{j=1}^{i} \frac{1}{s_j}} \tag{5.48}$$

and the α_4^2 load /job fraction assigned is:

$$\alpha_4^2 = \frac{L_2 + \alpha_{4r}^1 L_1 - f_3(\sum_{i=3}^{3} \prod_{j=1}^{i} s_j)}{1 + \sum_{i=1}^{3} \prod_{j=1}^{i} s_j} \tag{5.49}$$

The previous example is done for 4 sensors and a DLJT of 2 load/job. The following section presents a suitable expression for a N sensor network scenario. In the case of N single level tree network with one node P_i failure per L load/job assignment the following set of equations can be derived:

Using Fig.(5.6) a equation system in term of the same s_j and f_j for the $i + 1^{th}$ failing processor is shown below:

$$\alpha_1^2 = \alpha_2^2 s_{(1)} \tag{5.50}$$

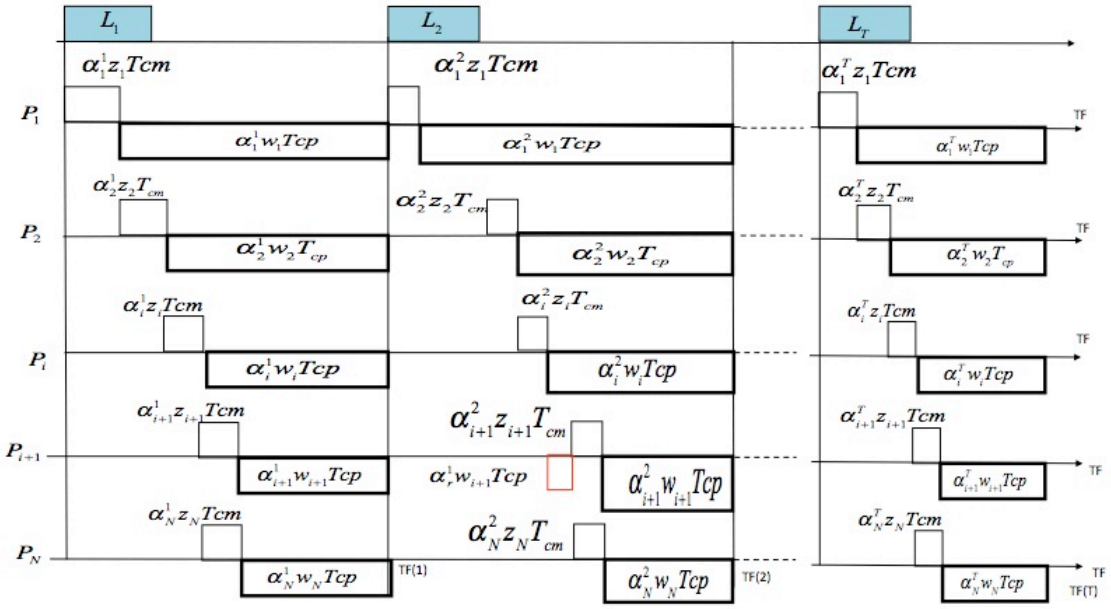

Figure 5.6: DLJT Load distribution finish time diagram one node partial failure

$$\alpha_2^2 = \alpha_3^2 s_{(2)} \tag{5.51}$$

$$\alpha_i^2 = \alpha_{i+1}^2 s_{(i)+f_{(i)}} \tag{5.52}$$

$$\alpha_{i+1}^2 = \alpha_{i+2}^2 s_{(i+1)} \tag{5.53}$$

$$\alpha_{N-1}^2 = \alpha_N^2 s_{(N-1)} \tag{5.54}$$

$$\tag{5.55}$$

The normalization equation will be,

$$L_2 + \alpha_{(pfail)r}^1 = \sum_{i=1}^{N} {\alpha_i}^2 \tag{5.56}$$

Solutions for this equation system for the α_1^2 and α_N^2 are presented below:

$$\alpha_1^2 = \frac{L_2 + \alpha_{pfailr}^1 L_1 + f_{(pfail-1)}(\sum_{i=(pfail-1)}^{N-(pfail-1)} \prod_{j=1}^{i} \frac{1}{s_j})}{1 + \sum_{i=1}^{N-1} \prod_{j=1}^{i} \frac{1}{s_j}} \tag{5.57}$$

$$\alpha_N^2 = \frac{L_2 + \alpha_{pfailr}^1 L_1 - f_{(pfail-1)}(\sum_{i=(pfail-1)}^{N} \prod_{j=1}^{i} s_j)}{1 + \sum_{i=1}^{N-1} \prod_{j=1}^{i} s_j} \tag{5.58}$$

The finish time for the L_2 load/job processing can be found using equation (5.27), thus

$$Tf(L_2(\lambda_1), P_1) = (Tcp \cdot w_1 + Tcm \cdot z_1) \frac{L_2 + \alpha_{pfailr}^1 L_1 + f_{(pfail-1)}(\sum_{i=(pfail-1)}^{N-(pfail-1)} \prod_{j=1}^{i} \frac{1}{s_j})}{1 + \sum_{i=1}^{N-1} \prod_{j=1}^{i} \frac{1}{s_j}} \tag{5.59}$$

Where s_j was defined in equation (5.60)

$$s_j = (w_{j+1} T_{cp} + z_{j+1} T_{cm}) / w_j T_{cp} \tag{5.60}$$

and f_j as

$$f_j = (\alpha_{2pfail}^1 w_{j+1} T_{cp}) / w_j T_{cp} \tag{5.61}$$

After the control processor recalculates load/job fractions it will assigned them to sensors. The incoming loaf of incoming L jobs the control

processor will queue until they are able to be assigned to the network. The finish total time of the DLJT will be:

$$DLJT_{FT} = \sum_{jt=1}^{pfail-1} Tf_{jt}(L_{jt}(\lambda_{jt})) + Tf_{pfail}(L_{pfail}(\lambda_{pfail})) + \sum_{jt=pfail+1}^{T} Tf_{jt}(L_{jt}(\lambda_{jt})) \tag{5.62}$$

5.4.3 Results and Commentaries

In the following section different tests performed on a homogeneous single level tree network are presented. The parameters for the setup of this simulation are summarized below:

$$w_i = 100$$

$$z_i = 70$$

$$Tcp = 1$$

$$Tcm = 1$$

The $DLJT_4$ represents the size of the load/job task to be processed using the 10 sensor networks. Therefore there are 4 L load/jobs to be processed by the network.

Fig.(5.7) is obtained when simulating a partial failure on sensor ID 2 during the entire $DLJT_4$. The vertical axis represent the load/job distribution among sensors. The horizontal axis represents the sensor number. The star legend in the graph shows the load distribution on the sensor when every sensor is able to processes the entire load assigned. In

this case this allocation of load will lead to obtain an optimal finish time [6]. On the other hand, when sensor 2 is fails to process its entire load in every L_i load/job the graphs overlaps, and shows that the amount of load reallocated to the failing processor is smaller than the optimal one. Notice the intersection point where the graphs pass across. After this the load/job fractions assigned to the nodes when the failure is presented are smaller than the initial optimal scenario.

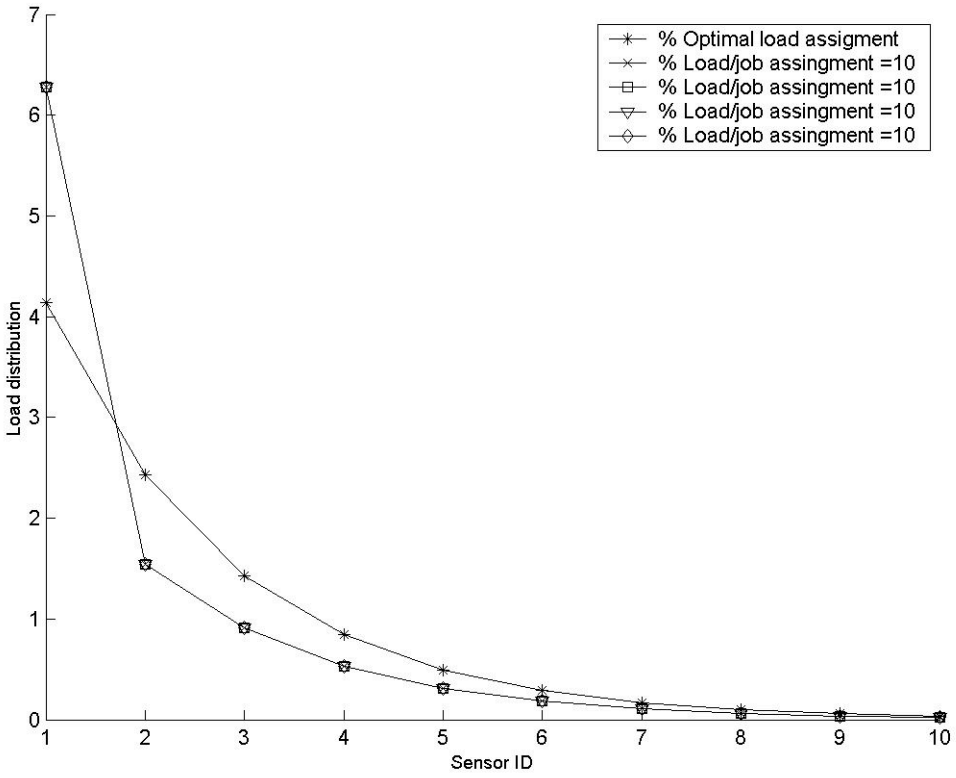

Figure 5.7: DLJT Load distribution finish time diagram one node 50 percent on Sensor ID 2 partial failure

Fig.(5.8) is present the percentage of load that is redistributed when a sensor fails. The vertical axis shows the percentage of load that has to be reallocated by the control processor when a failure is detected on the sensors, in this case sensor 2. The horizontal axis describes the sensor id in this wireless network. The positive values on the horizontal axis refers to an increase of the load on the sensor as opposed the negative sign, which means that the sensor node had less load reallocated.

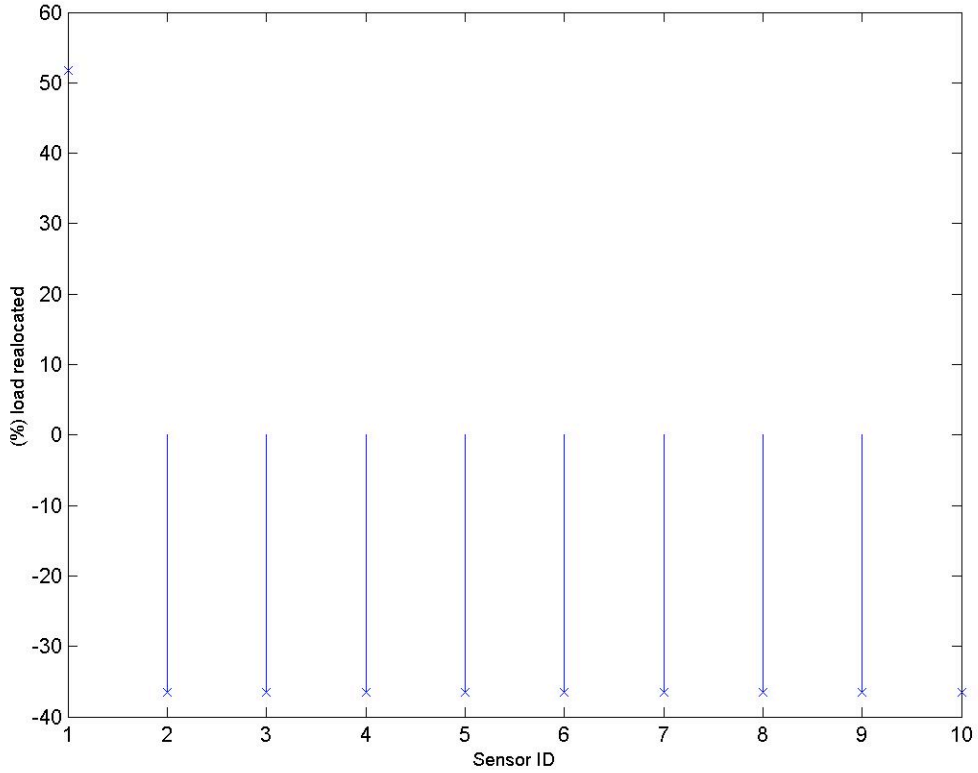

Figure 5.8: DLJT Percentage load relocated for a 50 percent on Sensor ID 2 partial failure

In Fig.(5.9) a case where sensor node 5 fails is presented. Notice that the crossing point, as in the Fig.(5.7) the crossing point among graph is located before the node that fails. This situation is also reflected on Fig.(5.11) for a sensor partial failure on sensor number 8.

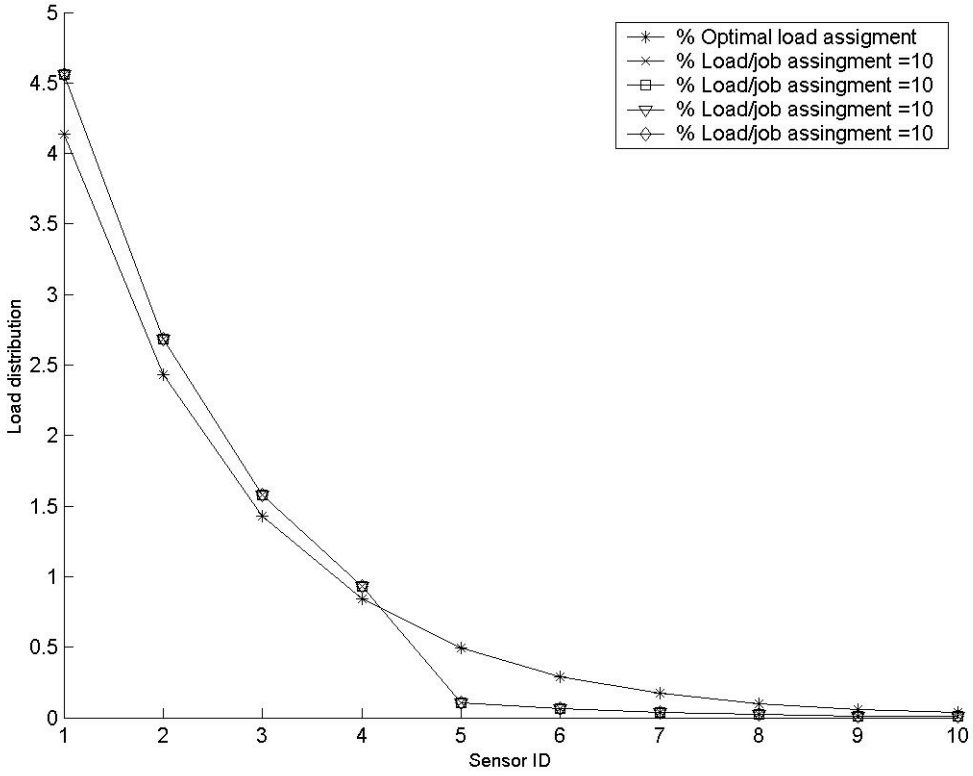

Figure 5.9: DLJT Load distribution finish time diagram one node 50 percent on Sensor ID 5 partial failure

As seen on previous plots for a failing sensors when sensor 8 fails the intersection point among the two graphs is located on the sensor preceding the failing sensor. When sensor node 8 fails the percentage of load change

can be seen in Fig.(5.12). Notice that the percentage of load relocated to other sensors is considerable small compared to the previous percentages relocated when the sensor 2 and 5 failed as presented on Fig.(5.8),(5.9).

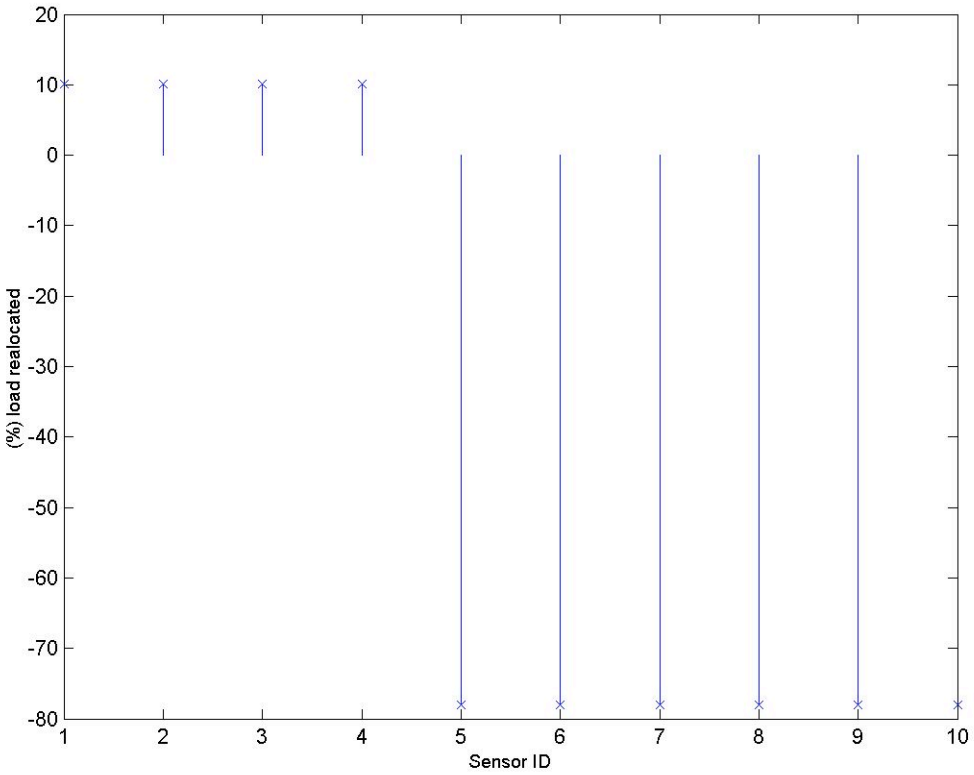

Figure 5.10: DLJT Percentage load relocated for a 50 percent on Sensor ID 5 partial failure

Fig.(5.13) relates the percentage of finish time change presented on the vertical axis of the figure obtained by simulating partial failures on different nodes on the wireless sensor network. Thus, by looking at the outer plot of the graph for sensor 2 when this nodes fails 10 percent of processing it load

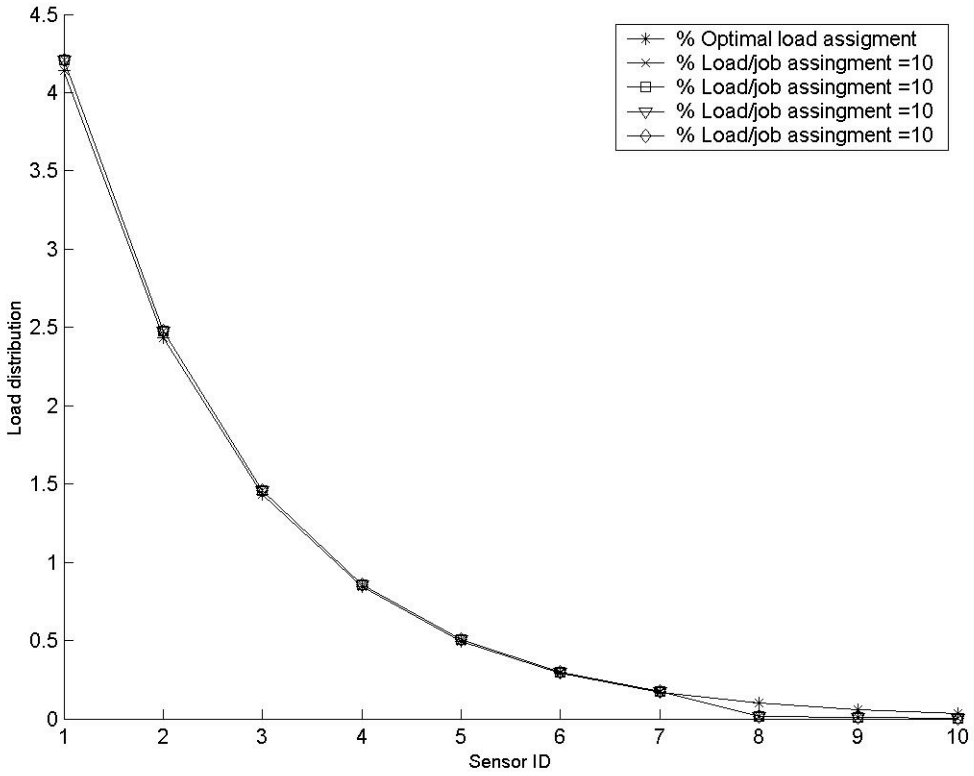

Figure 5.11: DLJT Load distribution finish time diagram one node 50 percent on Sensor ID 8 partial failure

assignment the amount of finish time increased is 62 percent with respect to the finish time obtained when the node is working on normal conditions. On the other hand if the sensor 9 fails to process 10 percent of its load assigned the finish time is affected only 37 percent, the graph that shows this behavior is located on the inner part of the chart.

For percentages values exceeding more than half of it original load/job L assigned the change of the finish time is small. Thus when the load

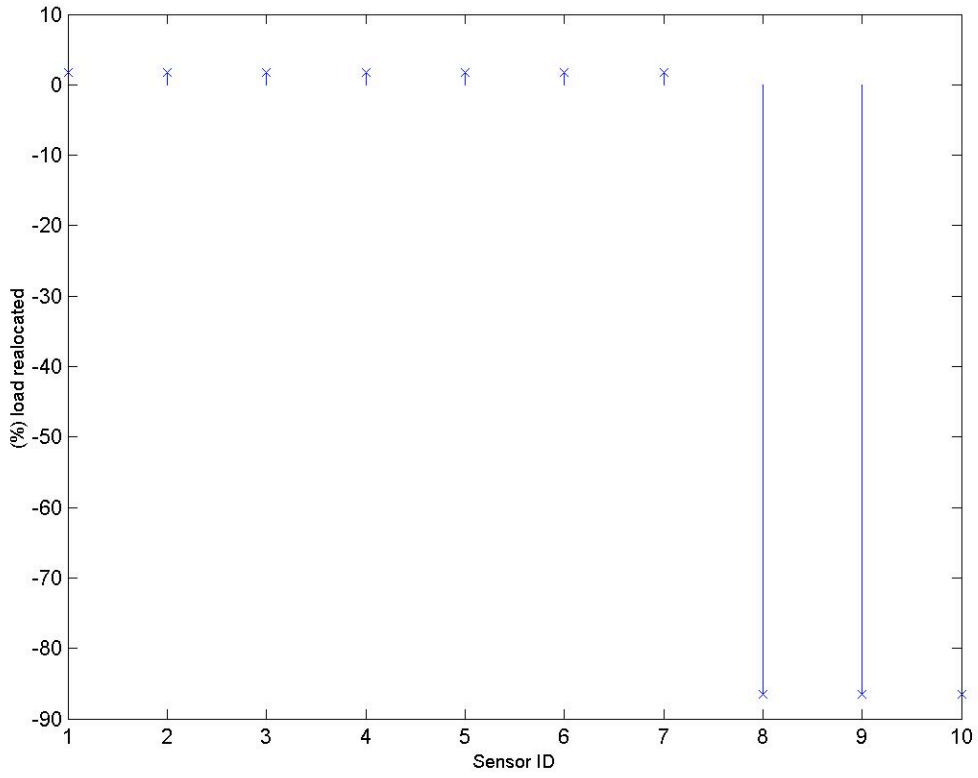

Figure 5.12: DLJT Percentage load relocated for a 50 percent on Sensor ID 8 partial failure

assigned fails to process the 80 percent of its load/job the changes on different sensor simulated it terms of resolution finish time is small only a 5 percent is increased. Noticed also that for different simulation of sensor failure for an 80 percent of load remain the finish time is higher for sensor 2 than sensor 9.

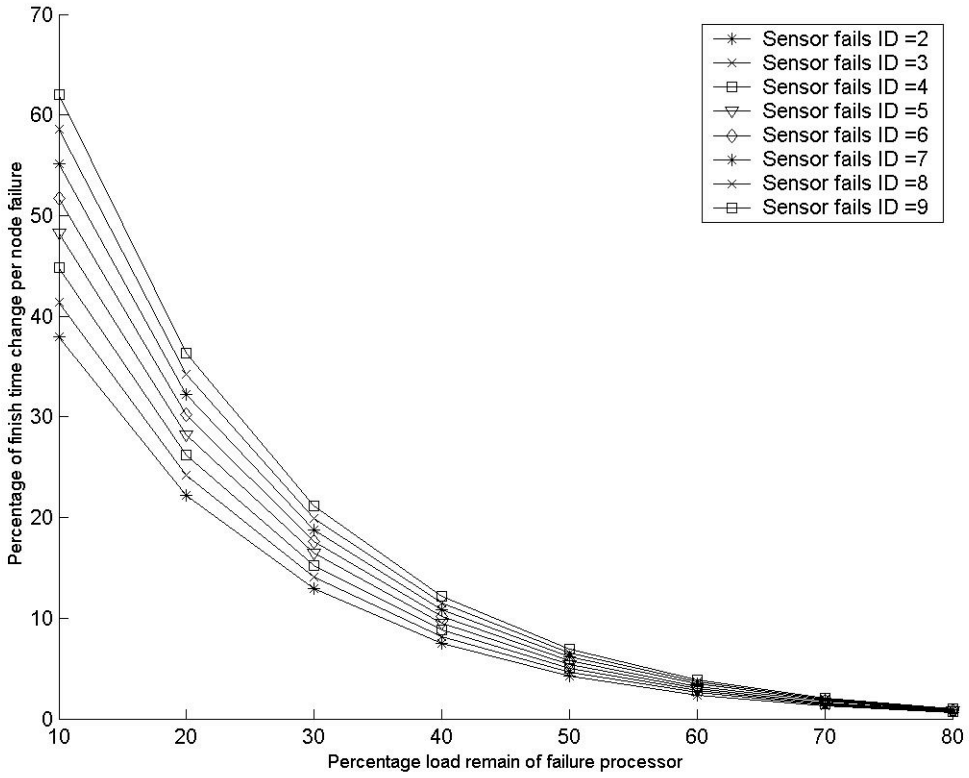

Figure 5.13: DLJT percentage finish time per percentage load failure

5.5 Conclusion

A novel strategy to processes different consecutive load assigned using divisible load theory on wireless sensor networks was presented. A heuristic methodology to handle different load distributions adapted to the current state of the load being processed on the wireless sensors was developed. A novel schedule mechanism was presented. Simulations showed that for a particular test wireless sensor network scenario when a partial failure

of a sensor occurs, if there is a new incoming load to be distributed, the preceding processor will increase the amount of load to be processed compared to its value in normal conditions of sensor operation.

Chapter 6

On Chip Interconnections for Wireless Sensor Networks using Divisible Load Theory

6.1 Summary

Most recent goal of the microprocessor industry is to develop multi-core architectures that currently range from 16 core to eventually hundreds core architecture system. This architecture would be able to allocate identical cores ordered on a single chip processor. Along with this technology the systems on a chip (SoCs) are increasingly investigated. This systems are composed of many cores and are allocated on the same microprocessor. Individual chip components that belong to the same system would require to have reliable interconnection networks that allows the intercommunication among core on the same chip. The network infrastructure that studies this interconnection topics is named as on-chip intercommunication networks (OCIN) or also know as Network on Chip

(NoC)[23]. Due to the nature of the Divisible Load Theory model which is based on the fact that the load to be process is arbitrarily partitionable it is envisioned to start to understand the feasibility of the applicability of this tool on SoCs. Speedup metrics for different DLT schedule protocols considered and applied to different multilink and multi-core topologies showing promising improvement when using multi-core architecture. Here speedup is the ratio of computation time on one processor to computation time on the entire network tree that could be implement on SoCs system via OCIN interconnection.

6.2 Introduction

Minimizing the transmission energy on wireless sensor networks is highly desirable due to the complexity on the conditions and the constraints of accessing power sources on applications where this type of technology could be deployed (environmental monitoring, security surveillance among others). Different approaches are being studied such as designing efficient protocols for transmission of data over wireless channels to reduce the signal to noise ratio.

When using WSN for data acquisition some application might require a pre filter or processed version of the data being collected by the network sensor array so instead of sending the entire raw data sample the node could send a synthesized version of the data measured. By transmitting a summarized and shorter version of the data collected the amount of energy

required for transmission would decrease. Thus being able to pre process samples in situ would enable a reduction in the energy of transmission used by the wireless sensors when reporting back the status of the area monitored.

The new emerging technology of system on a chip, SoCs, can be considered as promising technology to be integrated into the wireless sensors networks. Composed by several processors on a single chip the sensor SoC would allow an increase in the node processing capability which would be required when preprocessing the measured load is required. On chip interconnection networks (OCIN) bring the infrastructure to communicate between different cores on the same chip. Advantages of this technology are high bandwidth, low latency, low power communication compared to dedicated wiring devices. Different research efforts are on this technology are motivated on different areas such as, applications for embedded systems and personal electronics devices.

This chapter contributes to the research area of NoC and SoC by proposing different DLT scheduled protocols as tool to model distribution of load on the SoC. By doing so a possible more efficient network topology could be designed and deployed on wireless sensor networks.

6.3 The model

Due to the realistic and tractable nature DLT model and analysis is a suitable tool to be able to model interactions among different cores

located on a chip. On this section are presented different DLT scheduled policies that will be used on this study presented on [24]. Contrary to the methodology used initially on [24] to obtain optimal distribution of loads per processor basis, a simpler way to handle the mathematical relationships among processors is proposed. The focus is to obtain the speedup expression for a network topology.

The classical network topology often studied on DLT is a single level tree network show on Fig.(6.1)

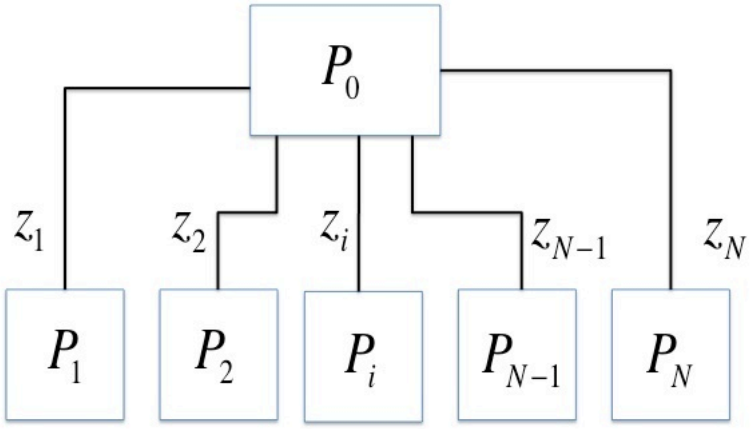

Figure 6.1: Single level tree network

This network topology considers only one channel of communication z_i per root processor to children processor pair. Also only one processor is

available on every child on the network.

The variables or parameters used on this model are:

α_i: The load share fraction assigned to the i^{th} link-processor pair.

w_i: The inverse of the computing speed of the i^{th} processor.

z_i: The inverse of the link speed of the i^{th} link.

T_{cp}: Computing intensity constant: the entire load is processed in $w_i T_{cp}$ seconds by the i^{th} processor.

T_{cm}: Communication intensity constant: the entire load can be transmitted in $z_i T_{cm}$ seconds over the i^{th} link.

T_i: Is the total time measured from the beginning of the scheduling process up to the end of the transmission of the data measured by the ith processor.

T_f: Is the time when the last processor finishes reporting. T_f=max(T_1,T_2,...,T_N)

Three different scheduling protocols will be reviewed for this network topology. The mathematical representation obtained for this based model will allow to extrapolate it to two different network topologies that are envisioned that could be apply or design using SoCs.

6.3.1 Sequential distribution, staggered start

As presented on different studies in past chapters, on this Gantt chart-like timing diagram the load distribution on the network is presented. The horizontal axis represents the time, communication time is presented above the axis and computation time is presented below the axis.

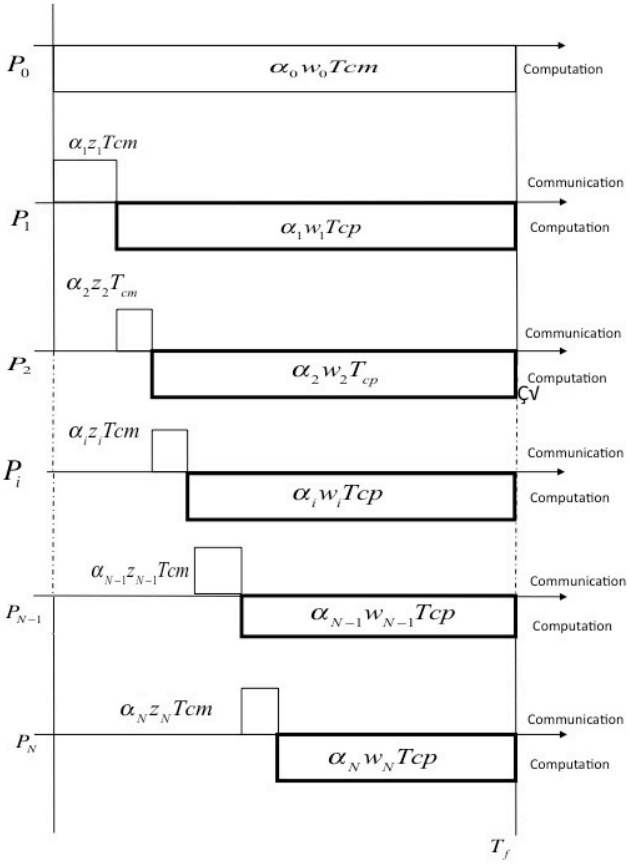

Figure 6.2: Timing diagram of single level tree with sequential distribution and staggered start

In order to find the optimal load distribution on each processor all

processors need to finish at the same time [24].

$$Tf(P_0) = Tf(P_1) \tag{6.1}$$

$$Tf(P_2) = Tf(P_3) \tag{6.2}$$

$$Tf(P_{N-1}) = Tf(P_N) \tag{6.3}$$

The equations below presented states that the communication and processing on the preceding processor is equal to the processing time of the current processor.

$$\alpha_0 w_0 T_{cp} = \alpha_1 z_1 T_{cm} + \alpha_1 w_1 T_{cp}$$

$$\alpha_1 w_1 T_{cp} = \alpha_2 z_2 T_{cm} + \alpha_2 w_2 T_{cp}$$

$$\alpha_i w_i T_{cp} = \alpha_{i+1} z_{i+1} T_{cm} + \alpha_{i+1} w_{i+1} T_{cp}$$

$$\alpha_{N-1} w_{N-1} T_{cp} = \alpha_N z_N T_{cm} + \alpha_N w_N T_{cp}$$

The normalization equation for N+1 processor is :

$$\alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 + \cdot + \alpha_i + \cdot + \alpha_{N-1} + \alpha_N = 1 \tag{6.4}$$

Expressing this equation in terms S_i

$$\alpha_0 = \alpha_1 S_{(0)} \tag{6.5}$$

$$\alpha_1 = \alpha_2 S_{(1)} \tag{6.6}$$

$$\alpha_i = \alpha_{i+1} S_{(i)} \tag{6.7}$$

$$\alpha_{N-1} = \alpha_N S_{(N-1)} \tag{6.8}$$

where

$$S_i = \frac{(z_{i+1} T_{cm} + w_{i+1} T_{cp})}{w_i T_{cp}} \tag{6.9}$$

After solving the previous equation system for α_0 with the normalization equation the following expression is obtained:

$$\alpha_0 = \frac{1}{1 + \sum_{i=0}^{N-1} \prod_{j=0}^{i} \frac{1}{S_j}} \tag{6.10}$$

This study will be focus on the speedup metric which is defined as the ratio computation time on one processor to the computation time on the entire N children network. Specifically the speedup will be studied for a homogeneous single level tree network. So it is intended to measure the parallel processing advantage on SoCs using the speedup relationship of the conventional DLT as:

$$Speedup = \frac{T_{f0}}{T_{fN}} \tag{6.11}$$

Where T_{f0} represents the time processing the entire load α_0 equals to 1. Thus,

$$T_{f0} = \alpha_0 w_0 Tcp \tag{6.12}$$

$$T_{f0} = 1 \cdot w_0 Tcp \tag{6.13}$$

and

$$T_{fN} = \frac{1}{1 + \sum_{i=0}^{N-1} \prod_{j=0}^{i} \frac{1}{S_j}} w_0 Tcp \tag{6.14}$$

And T_{fN} represents the finish time for the load to be resolved using the divisible load scheduling mechanism on the single level tree network presented on Fig.(6.1).

As mentioned before, for the speedup for a homogeneous single level tree network, in that particular case every $S_i = S$ for i from 1=N so link speed are equal on the network and the processor speed as well. Thus equation (6.14) can be rewriting as:

$$T_{fN} = \frac{1}{1 + \frac{1}{S_0}(1 + \sum_{i=1}^{N-1} \frac{1}{S^i})} w_0 Tcp \tag{6.15}$$

And the corresponding speedup will be:

$$Speedup = 1 + \frac{1}{S_0}(1 + \sum_{i=1}^{N-1} \frac{1}{S^i}) \tag{6.16}$$

6.3.2 Simultaneous distribution, staggered start

Different than the previous protocol, the processors now simultaneously receive the data and only start to process it a soon as the processor receives it entire load assignment Fig.(6.3).

The equation that describes this model are:

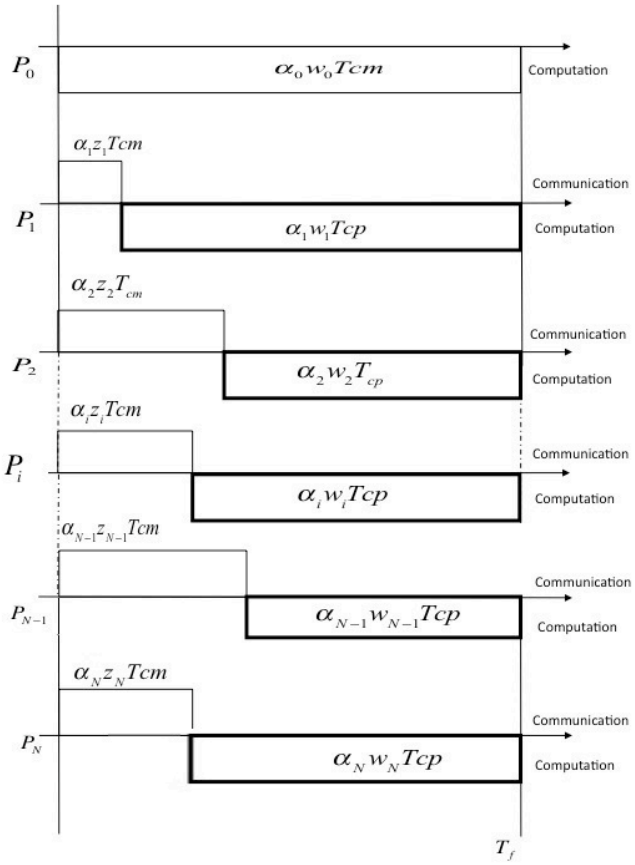

Figure 6.3: Timing diagram of single level tree with simultaneous distribution and staggered start

$$\alpha_0 w_0 T_{cp} = \alpha_1 z_1 T_{cm} + \alpha_1 w_1 T_{cp}$$

$$\alpha_1 w_1 T_{cp} + \alpha_1 z_1 T_{cm} = \alpha_2 z_2 T_{cm} + \alpha_2 w_2 T_{cp}$$

$$\alpha_i w_i T_{cp} + \alpha_i z_i T_{cm} = \alpha_{i+1} z_{i+1} T_{cm} + \alpha_{i+1} w_{i+1} T_{cp}$$

$$\alpha_{N-1} w_{N-1} T_{cp} + \alpha_{N-1} z_{N-1} T_{cm} = \alpha_N z_N T_{cm} + \alpha_N w_N T_{cp}$$

Equation (6.4) can be used as well as normalization equation.

Expressing the previous equation system in terms of g_1 and s_i for i from 1 to N-1.

$$\alpha_0 = \alpha_1 g_{(1)} \tag{6.17}$$

$$\alpha_1 = \alpha_2 S_{(1)} \tag{6.18}$$

$$\alpha_i = \alpha_{i+1} S_{(i)} \tag{6.19}$$

$$\alpha_{N-1} = \alpha_N S_{(N-1)} \tag{6.20}$$

where

$$S_i = \frac{(z_{i+1} T_{cm} + w_{i+1} T_{cp})}{z_i T_{cm} + w_i T_{cp}} \tag{6.21}$$

and

$$g_1 = \frac{(z_1 T_{cm} + w_1 T_{cp})}{w_0 T_{cp}} \tag{6.22}$$

The correspondent fraction of load for this particular schedule protocol can be found as

$$\alpha_0 = \frac{1}{1 + \frac{1}{g_1}(1 + \sum_{i=1}^{N-1} \prod_{j=1}^{i} \frac{1}{s_j})} \tag{6.23}$$

The general expression for speedup will be,

$$Speedup = \frac{1}{\frac{1}{1+\frac{1}{g_1}(1+\sum_{i=1}^{N-1} \prod_{j=1}^{i} \frac{1}{s_j})}} \tag{6.24}$$

When the homogenous network is considered the speedup can be expressed as

$$Speedup = \frac{1}{\frac{1}{1+\frac{1}{g_1}(1+\sum_{i=1}^{N-1}\prod_{j=1}^{i}\frac{1}{1})}} \tag{6.25}$$

Simplifying the above equation,

$$Speedup = 1 + \frac{1}{g_1}(1 + N - 1) \tag{6.26}$$

The final expression will be

$$Speedup = 1 + \frac{1}{g_1}(N) \tag{6.27}$$

6.3.3 Simultaneous distribution, simultaneous start

Fig.(6.4) presents the last protocol considered here.

In this case the processors are able to process the load as soon as they receive the initial transmission. It is assume that all finish at the same time to achieve the optimal distribution of load.

$$\alpha_0 w_0 T_{cp} = \alpha_1 w_1 T_{cp}$$

$$\alpha_1 w_1 T_{cp} = \alpha_2 w_2 T_{cp}$$

$$\alpha_i w_i T_{cp} = \alpha_{i+1} w_i T_{cp}$$

$$\alpha_{N-1} w_{N-1} T_{cp} = \alpha_N w_{N-1} T_{cp}$$

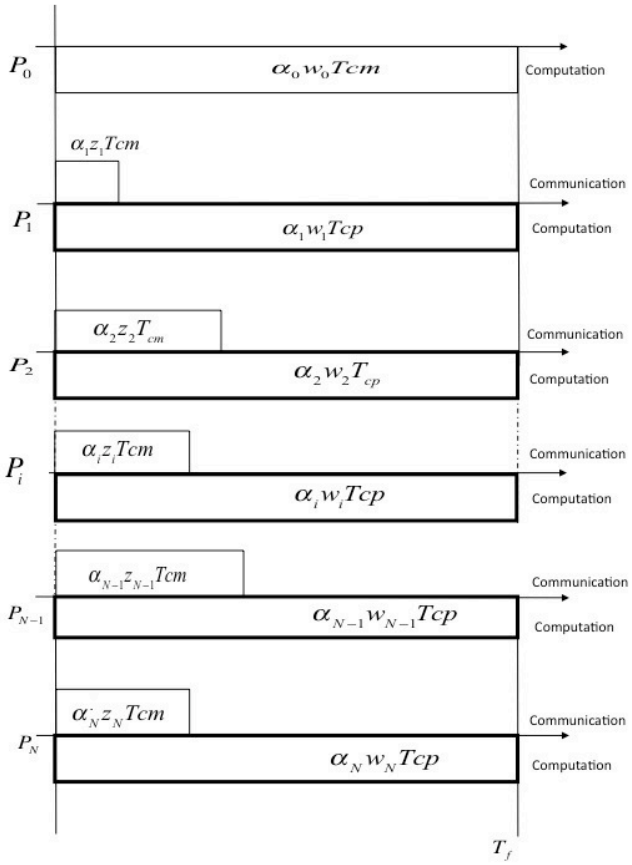

Figure 6.4: Timing diagram of single level tree with simultaneous distribution and simultaneous start

Note, that it is assume that

$$\alpha_i z_i T_{cm} < \alpha_i w_i T_{cp}$$

By expressing the previous equation system in terms of f_i the previous equation system can be written as,

$$\alpha_0 = \alpha_1 f_{(0)} \tag{6.28}$$

$$\alpha_1 = \alpha_2 f_{(1)} \tag{6.29}$$

$$\alpha_i = \alpha_{i+1} f_{(i)} \tag{6.30}$$

$$\alpha_{N-1} = \alpha_N f_{(N-1)} \tag{6.31}$$

where

$$f_i = \frac{(w_{i+1} T_{cp})}{w_i T_{cp}} \tag{6.32}$$

The optimal load fraction assigned to each the root processor is

$$\alpha_0 = \frac{1}{1 + \sum_{i=0}^{N-1} \prod_{j=0}^{i} \frac{1}{f_j}} \tag{6.33}$$

Here the speedup will be,

$$Speedup = (1 + \sum_{i=1}^{N-1} \prod_{j=1}^{i} \frac{1}{f_{(j)}}) \tag{6.34}$$

Considering a homogeneous network the equation below presented represents the speedup for the simultaneous start simultaneous processing protocol in this case it is assume that all processor on the network are similar with the exception of the root processor which is w_0. Thus the speedup is,

$$Speedup = 1 + \frac{1}{f_0}(1 + \sum_{i=1}^{N-1} \prod_{j=1}^{i} 1) \tag{6.35}$$

After simplifying the above equation,

$$Speedup = 1 + \frac{1}{f_0}(N) \tag{6.36}$$

6.4 DLT on SoC and NoC for Wireless Sensor Networks: M parallel interconnection channels

On this section a specific node from a wireless sensor array will be considered using SoC and NoC. There are two type of architectures proposed to be used on the chip. The goal here is to use conventional DLT to start to understand the interaction among cores and links on a sensor using SoC and NoC.

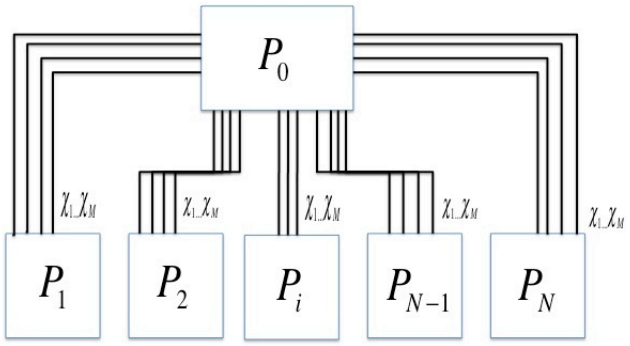

Figure 6.5: Single level tree network M parallel interconnection channels

Fig.(6.5) shows a single level tree architecture consistent on N+1

processor and M links (χ) per processor pair. On this model root core will be able to distribute the load assigned to every core in parallel fashion using M links (χ) available. Every connection pair is similar on every node, thus there always the same amount of channels available for every root core child core pair in the network.

Two new variables used on this model needs to be presented at this time;

$\pi_{i,j}$: The inverse of the computing speed of the i^{th} core on a virtual processor.

$\chi_{i,j}$: The z_j / M inverse of the link speed of the j^{th} link.

j: The represents the link number among root processor and children processor i^{th}.

A homogeneous network topology is considered during this study. Thus every all the communication channels on the network are the same. In addition all the cores used on the system are equal as well. Thus

$$\chi_{i,j} = \frac{Z_{ij}}{M} \tag{6.37}$$

where

i=$1 \cdots N$ number processors .

j=$1 \cdots M$ parallel links per processor pair.

For this particular case the mathematical expressions obtained to described the optimal load assignment on the past section can be extrapolated for the network architecture shown in Fig.(6.5) by adjusting the proper parameter in this case the link speed parameter.

6.5 Speedup for different schedule protocols on single level tree network chip architecture with M parallel links

In the past sections different speedup expressions were reviewed for different schedule protocols for the network architecture presented in Fig.(6.1). These expressions were defined in terms for dummy variables which related the link speed and the processor speed. Thus different expression for different schedule protocols for the speedup were obtained in terms of this dummy parameter which relates the speed of the processor and the link speed.

By redefining the dummy parameter according to the new network topology the new speedup expressions can be found. The following sections will show the speedup for the network topology presented on Fig.(6.5)

6.5.1 Speedup for sequential distribution sequential processing with M parallel links

Equation (6.16) shows the speedup for a network topology one link to one processor. The dummy parameter defined in there is $S(i)$,

$$S_i = \frac{(z_{i+1}T_{cm} + w_{i+1}T_{cp})}{w_i T_{cp}} \tag{6.38}$$

When considering this parameter for M parallel links the new expression need to be adjusted. The parameter adjusted is the speed of the link Z_i. Using equation (6.37) B will be the new name for the dummy variable for this protocol on the new architecture.

$$B_i = \frac{(\frac{Z_{ji}}{M}T_{cm} + w_{i+1}T_{cp})}{w_i T_{cp}} \tag{6.39}$$

In the case of using an homogeneous network topology where the M links are equal and the the speed of processor is the same on all cores on the network, equation (6.40) can be used: Thus,

$$B_i = \frac{(\frac{Z}{M}T_{cm} + wT_{cp})}{wT_{cp}} \tag{6.40}$$

and

$$B_0 = \frac{(\frac{Z}{M}T_{cm} + w_1T_{cp})}{w_0T_{cp}} \tag{6.41}$$

The speedup for this found for this protocol and this architecture is:

$$Speedup = 1 + \frac{1}{B_0}(1 + \sum_{i=1}^{N-1} \frac{1}{B^i}) \tag{6.42}$$

6.5.2 Speedup for simultaneous distribution sequential processing with M parallel links

Following the same methodology as in section (6.5) the speedup expression for this protocol will be obtained from equation (6.24). The dummy variables used to related link speed and processor speed for this protocol were:

$$B_i = \frac{(\chi_{j(i+1)} T_{cm} + w_{i+1} T_{cp})}{\chi_{ji} T_{cm} + w_i T_{cp}} \tag{6.43}$$

and

$$G_1 = \frac{(\chi_{j1} T_{cm} + w_1 T_{cp})}{w_0 T_{cp}} \tag{6.44}$$

Equation (6.45) represents the speedup equation using the new dummy variables according to this network topology,

$$Speedup = \frac{1}{\frac{1}{1+\frac{1}{G_1}(1+\sum_{i=1}^{N-1}\prod_{j=1}^{i}\frac{1}{B_j})}} \tag{6.45}$$

For a homogeneous network case from equation (6.27) and substituting the parameters,

$$Speedup = 1 + \frac{1}{G_1}(N) \tag{6.46}$$

6.5.3 Speedup for simultaneous distribution simultaneous start with M parallel links

The last schedule protocol considered on the network topology presented on Fig.(6.5). Following the same methodology before proposed the dummy parameters that represents the relationship among processors and links in the case of M=1 the conventional case considered in section (6.3.3). As can be seen the dummy variable does not depend on speed of the link. Thus, the speedup for this network architecture remains the same.

6.6 DLT on SoC and NoC for Wireless Sensor Networks: M parallel cores

The network topology is considered on Fig.(6.6) consist on a root processor P_0 that distribute or assigns load to a virtual equivalent processor which is composed by G different cores. The load assignment among root processor and virtual processor is can be done using the previous schedule protocols from the classical DLT sections (6.3.1), (6.3.2) and (6.3.3). Within the virtual processor the cores are organized in a single level tree fashion, DLT is used to distribute the load among cores on every virtual processor.

It is assumed that every load assignment is arbitrarily partition able. The schedule protocol studied for this technology is the Simultaneous Distribution and Simultaneous processing.

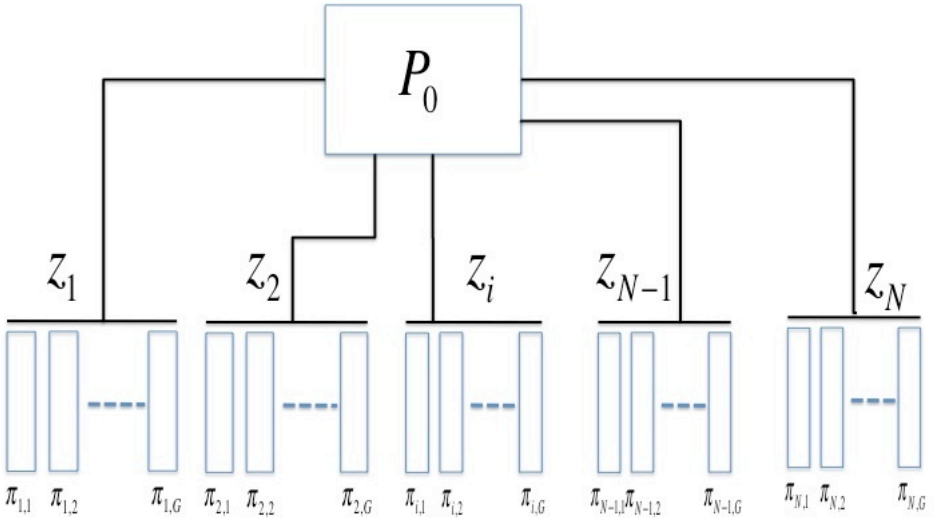

Figure 6.6: Single level tree network M parallel cores

Where,

P_0: Is the root processor with inverse of the computing speed w_0.

g: The represents the core number that belongs to a virtual processor i^{th}.

$\pi_{i,g}$: The inverse of the computing speed of the g^{th} core on a virtual processor.

z_c: The inverse of the link speed of the g^{th} link.

6.6.1 Speedup for sequential distribution sequential processing with M parallel cores

Using the same methodology presented previously the speedup for the virtual processors can be obtain as:

$$Speedup = 1 + \frac{1}{1 + \sum_{i=0}^{N-1} \prod_{j=0}^{i} \frac{1}{BP_j}} \tag{6.47}$$

where

$$BP_0 = \frac{(z_1 T_{cm} + weq_1 T_{cp})}{w_0} \tag{6.48}$$

and for $i = 1$ to N virtual processors

$$BP_i = \frac{(z_{i+1} T_{cm} + weq_{i+1} T_{cp})}{weq_i} \tag{6.49}$$

The single level tree core network on the the virtual processor can be collapsed into one single node this will allowed to find the equivalent computation speed weq,

$$weq_i = \frac{1}{1 + \sum_{i=0}^{G-1} \prod_{j=0}^{i} \frac{1}{sc_j}} \tag{6.50}$$

G is the number of cores on the every virtual processor and SC is the constant that related interconnection links and cores on every virtual processor.

$$SC_g = \frac{(z_{c_{g+1}} T_{cm} + \pi_{g+1} T_{cp})}{\pi_g T_{cp}} \tag{6.51}$$

The sequential distribution sequential processing is used as well on the virtual processor to distribute load among cores processors.

Contrary as previously shown in equation (6.50) if the equivalent computation speed weq in SoC virtual processor can be described as:

$$weq = \frac{w}{G} \tag{6.52}$$

A possible theoretical limit can be found for this network topology with M parallel cores per virtual processor.

6.6.2 Speedup for simultaneous distribution simultaneous processing with M parallel cores

Another schedule protocol to assign load to the rest of the virtual processors is simultaneous distribution simultaneous start. Equation (6.34), (6.35) represent the speedup expressions for this protocol. When the incoming load on every virtual processor arrives the load is partitioned and assigned

using the simultaneous distribution and simultaneous start presented on section (6.3.3). For this case the weq will be expressed as:

$$weq_i = \frac{1}{1 + \sum_{i=0}^{G-1} \prod_{j=0}^{i} \frac{1}{fc_j}} \tag{6.53}$$

Where,

$$fc_j = \frac{(\pi_{j+1} T_{cp})}{\pi_j T_{cp}} \tag{6.54}$$

Considering a homogeneous network the equation below presented represents equivalent computation speed weq for the simultaneous start simultaneous processing protocol in this case it is assume that all cores on the network are similar with the exception of the root processor which is π_0. Thus weq_i is,

$$weq_i = \frac{1}{1 + \frac{1}{f_{co}}(1 + \sum_{i=1}^{G-1} \prod_{j=1}^{i} 1)} \tag{6.55}$$

After simplifying the above equation,

$$weq_i = \frac{1}{1 + \frac{1}{f_{co}} G} \tag{6.56}$$

To find the general speedup of the entire network topology the equations (6.32), (6.34), (6.35) and (6.36) are used and adjusted to this network topology using the equations above presented. Thus,

$$speedup = 1 + w_0 \frac{(f_{co} + G)}{f_{co}} N \tag{6.57}$$

Equation (6.57) can be expressed in terms of cores π and root processor w_0 as :

$$speedup = 1 + w_0 N \frac{(\pi_1 + \pi_0 G)}{\pi_1} \tag{6.58}$$

6.7 Results and Commentaries

The schedule protocols, Sequential distribution Staggered Start and Simultaneous distribution Staggered Start was simulated for a singe level tree network with M parallel links was simulated with the following parameters:

$w_i = 100$ for i=1 to 10, 20 40.

$w_0 = 90$.

$z_i = 100$ for i=1 to 20.

$\pi_0 = 95$.

$\pi_i = 100$ for i=1 to 10.

$T_{cp} = 1$

T_{cm}= 1

In Fig.(6.7) are shown results for the Sequential distribution and Staggered start for single level tree M parallel links network topology. Three different networks size were simulated composed of 10, 20 and 40 processors. On the vertical axis the value of speedup is presented. On the other hand the on the horizontal axis presents the number of parallel links.

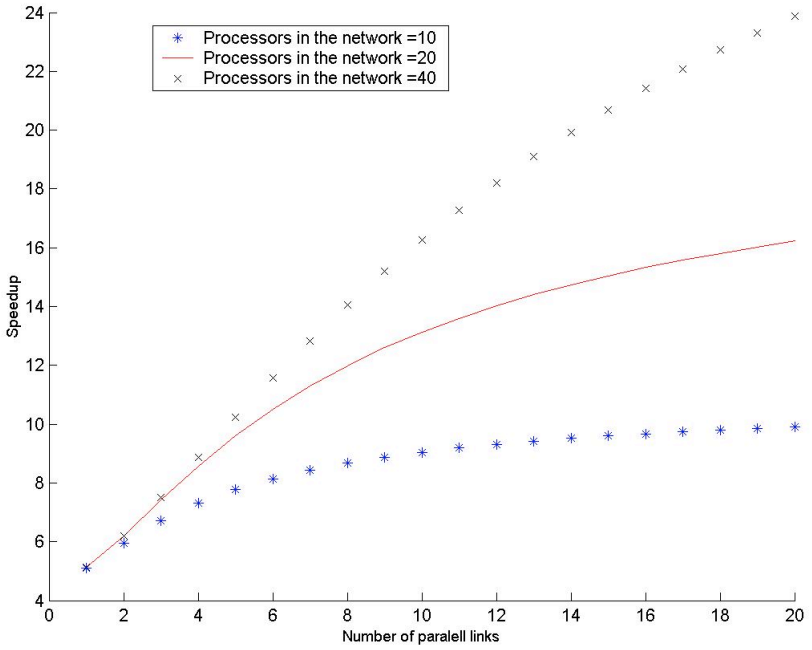

Figure 6.7: Speedup for a single level tree M parallel links with sequential distribution and sequential start

It can be seen that the speedup achieved after 5 parallel links is higher for the network simulated. After this point the value for speedup for the network with 10 processor has a slower increase compared to the rests of the network simulated.

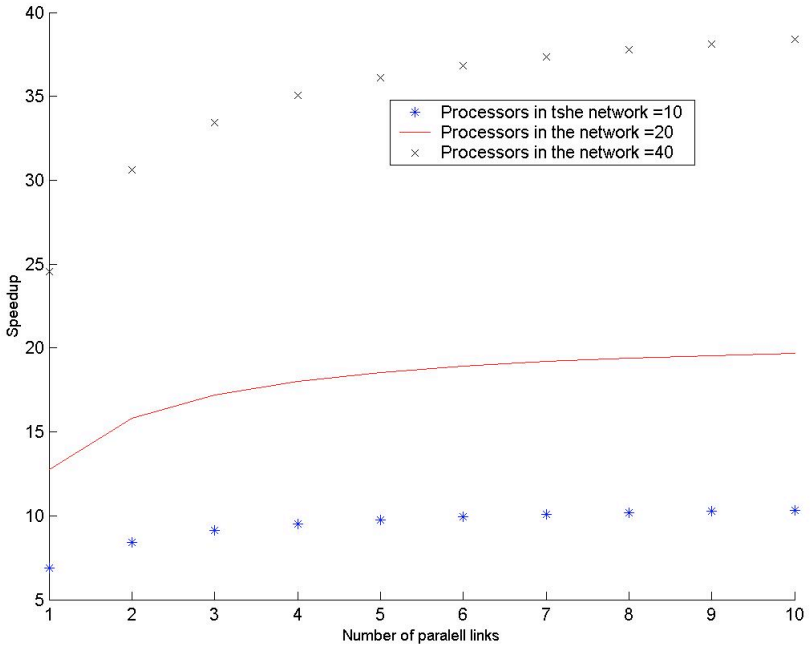

Figure 6.8: Speedup for a single level tree with M parallel links simultaneous distribution and staggered start

Fig.(6.8) presents the simulations results for the protocol with Simultaneous Distribution and Staggered Start. The speedup plot for the network of 10 processor rapidly saturates to 10 after 5 parallel links. After this point adding extra links will not gain and eventually would increase the energy used for transmission among nodes and root processor. In addition the speedup is scaled by the network size, as shown on different plots for the network simulated.

Fig.(6.9) shows the simulation for a single level tree network with M parallel cores. The network topology simulated consisted on a single level

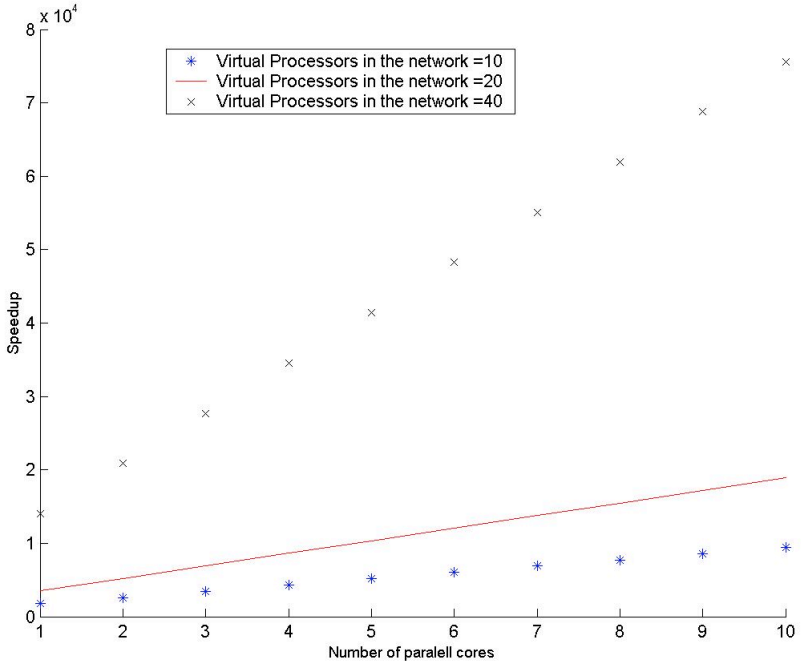

Figure 6.9: Speedup for a single level tree with M parallel cores simultaneous distribution and simultaneous start

tree network that uses the a simultaneous distribution and simultaneous start to assigned load from root to virtual. This protocol is used to distribute load among cores on every virtual processor. On the vertical axis is shown the metric for speedup on the horizontal axis are the number of cores simulated per virtual processor basis. Notice the high values of the speedup for different network sizes.

6.8 Conclusion

A novel network topology using DLT on NOC and SoC was presented. By using the schedule protocol of the classical DLT and extrapolating them to the new single level tree network topology with N parallel different links and M different processors expressions for speedup were found. Simulation showed that the Simultaneous Distribution and Staggered Start outperform the rest of protocol simulated. Thus for the same network size and the same amount of parallel links a grater speedup value could be obtained. In addition simulation showed possible saturation points for the number of parallel links, thus there is not a proportional increase on the speedup while increasing the number of links on every protocol. This finding will help future research on energy minimization for data transmission since could avoid to deployed unnecessary links and therefore reduce energy. In addition by simulating speedup of a network topology with M parallel cores per processor (SoC) for a specific DLT protocol high speedup where found.

Chapter 7

Conclusion

On this research project different studies were presented taking into account processing time and monetary cost for a single level tree networks. Chapter 2 presented an analytical close solution for optimum finish, reporting and pre-processing time for a single level tree sensor network. Simulations for each of the protocols were presented as well. Chapter 3 considered the total monetary cost function for a single level tree network under changes on the link speed and processor speed. Simulation results were provided for a specific set of networks parameters that showed a complex relationship between the total monetary cost and the specific network parameters. In chapter 4 such a relationship is explored, proposing a strategy to study the monetary total cost in a single level tree star network as a function of a non linear load parameter simulation results were presented. Chapter 5 a different heuristics methodologies to adapt the data acquisition protocols were developed for a single level tree network. A robust mechanism was developed using DLT when a sensor has a partial failure when processing

its load job assigned. On chapter 6 a for the first time on the DLT field the possibility to use DLT on SoC and NoC for wireless sensor networks was presented. Different classical DLT schedule protocol were adapted to this new emerging technology. Simulations for speedups demonstrate wireless sensor networks using DLT could take advantage of this new SoC and NoC technology. The importance of this finding is that will encourage more studies of DLT and SoC and NoC.

It has been found that DLT is a feasible mathematical tool to study scheduling protocols for wireless sensor networks.

Bibliography

[1] M. Moges and T.G. Robertazzi, *Wireless Sensor Networks: Scheduling for Measurement and Data Reporting* , IEEE Transactions on Aerospace and Electronic Systems, vol. 42, no. 1, pp. 327-340, Jan. 2006.

[2] Y.C. Cheng and T.G. Robertazzi, *Distributed computation with communication delays*, IEEE Transactions on Aerospace and Electronic Systems, Vol. 22, pp. 60-79, 1988.

[3] T.G. Robertazzi, *Ten reasons to use divisible load theory*, Computer, Vol. 36, pp. 63-68, 2003.

[4] Bharadwaj, V., Ghose, D., and Mani, V., *A Study of Optimality Conditions for Load Distribution in Tree Networks with Communication Delays*, Dept. of Aerospace Engineering, Indian Institute of Science, Bangalore, India, Technical Report 423/GI/02-92, Dec. 1992.

[5] J. Sohn and T.G. Robertazzi, *Optimal divisible load sharing for bus networks*, IEEE Transactions on Aerospace and Electronic Systems, Vol. 32, pp. 34-40, 1996.

[6] 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.

[7] T.G. Robertazzi, *Processor equivalence for a linear daisy chain of load sharing processors*, IEEE Transactions on Aerospace and Electronic Systems, Vol. 29, pp. 1216-1221, 1993.

[8] J.F. Kurose and R. Simha, *A microeconomic approach to optimal resource allocation in distributed computer systems*, IEEE Trans. Networking, Vol. 38, no. 5, pp. 705-717, May 1989.

[9] D. Menasce and V. Almeida, *Cost-performance analysis of heterogeneity in supercomputer architectures*, Proc. IEEE ACM Supercomputing '90, pp.169-177, 1990.

[10] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang, *Pricing in computer networks: motivation, formulation and example*, Trans. Networking, Vol. 1, no 6, pp.657-627, Dec. 1993.

[11] J. Sohn, T.G. Robertazzi and S. Luryi, *Optimizing Computing Costs Using Divisible Load Analysis*, IEEE Transactions on Parallel and Distributed Systems, Vol. 9, pp. 225-234, 1998.

[12] J. Blazewicz and M. Drozdowski, *The performance limits of a two dimensional network of load sharing processors*, Foundations of Computing and Decision Sciences, Vol. 21, pp. 3-15, 1996.

[13] M.Moges, L.A. Ramirez, C. Gamboa and T,G. Robertazzi, *Monetary Cost and Energy Use Optimization in Divisible Load Processing*, Proc. of the 2004 Conference on Information Sciences and Systems, Princeton University, March 2004.

[14] C.F. Gamboa and T.G. Robertazzi, *Optimizing a Divisible Load Nonlinear Cost Function*, 2005 Conference on Information Sciences and Systems, The Johns Hopkins University, Baltimore, Maryland, March 2005.

[15] S. Charcranoon, T.G. Robertazzi and S. Luryi, *Cost Efficient Processor Arrangement in Single Level Tree Network*, State Univ. of New York at Stony Brook College of Eng. and Applied Science Technical Report 757, 30 Mar. 1998,available from T. Robertazzi. IEEE Trans. on Para. and Dist. Sys. 9 (1998), 225-234. Also related: US Patent 5,889,989, J. Sohn, T.G. Robertazzi and S. Luryi, Load sharing controller for optimizing monetary cost, March 30, 1999.

[16] A. Cerpa et al., *Habitat monitoring: Application driver for wireless communications technology*, 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean, Costa Rica, April 2001.

[17] G. J. Pottie, W. J. Kaiser, *Wireless Integrated Network Sensors*, Communications of ACM, vol. 43, no. 5, pp. 551-558, May 2000.

[18] V. Bharadwaj, D. Ghose, V. Mani, *Design and Analysis of Load Distribution Strategies for Infinitely Divisible Loads in Distributed Processing Networks with Communication Delays*, Dept. of Aerospace Engineering, Indian Institute of Science, Bangalore India Technical Report 422/GC/01-92, Oct. 1992.

[19] H.M. Wong, D. Yu, B. Veeravalli 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 (PDCS 2003), Nov. 2003.

[20] V. Bharawaj, D. Ghose and T.G. Robertazzi, *Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems*, in the special issue of Cluster Computing on Divisible Load Scheduling (D. Ghose and T. Robertazzi, editors), spring 2003.

[21] S. Charcranoon, T.G. Robertazzi, and S. Luryi, *Load Sequencing for a Parallel Processing Utility* Journal of Parallel and Distributed Computing, vol. 64, 2004, pp. 29-35.

[22] J.T. Hung and T.G. Robertazzi, *Distributed Scheduling of Nonlinear Computational Loads* Proceedings of the 2003 Conference on Information Sciences and Systems, The Johns Hopkins University, Baltimore, MD, March 2003.

[23] J. D.Ownes, W. J. Daily, R. Ho, D.N. Jayasimha, S. W. Keckler,

L. Peh, *Research Challenges for ON-CHIP Interconnections Networks* IEEE Computer Society, 0272-1732, 2007.

[24] T.G. Robertazzi, *Networks and Grids Technology and Theory* Stony Brook, NY US. Springer ISBN 13-978-0-387-36758-3, 2007.

[25] Bharadwaj, V., Li, H.F. and Radhakrishnan, T., *Scheduling Divisible Loads in Bus Networks with Arbitrary Processor Release Times* Computers and Mathematics with Applications, Pergamon Press, Vol. 32, No. 7, 1996, pp. 57-77.