Modeling for Integration of Divisible Load Theory and Markov Chains

Mequanint A. Moges,
Thomas G. Robertazzi[1]
Department of Electrical and
Computer Engineering
Stony Brook University
Stony Brook, NY 11794
e-mail:
(mmoges,tom)@ece.sunysb.edu

Abstract —

In this paper the equivalence between various divisible load-scheduling policies and continuous time Markov chains is demonstrated. The problem is to show optimal divisible load schedules for various network topologies have Markov chain analogs. This paper is a continuation of our initial short paper [1] that introduced this unification between divisible load theory and Markov chain models for the first time. A detailed analytical analysis of linear daisy chains and single and two level tree networks is presented in this paper. While most of the Markov chains are one dimensional in topology, the labeling of transitions is different from the usual practice in queueing theory.

I. Introduction

Divisible load scheduling theory (DLT) involves the study of the optimal distribution of partitionable loads among a number of processors and links [2,3,4]. A partitionable data parallel load is one that can be arbitrarily distributed among the processors and links in a system. Thus there are no precedence relations among the data. Applications include grid computing, parallel and distributed processor network scheduling, data intensive computing and metacomputing. The approach is particularly suited to the processing of very large data files as in signal processing, Kalman filtering, image processing, experimental data processing, multimedia and computer utility applications.

There has been an increasing amount of study on divisible load scheduling theory since the work of Cheng and Robertazzi [5] in 1988. Most of these studies develop an efficient allocation of load to processors over a network by forcing the processors to all stop processing at the same time. Intuitively, this is because the solution could be improved by transferring load if some processors were idle while others are still busy [6]. Optimal allocation of loads using a set of recursive equations were presented for network topologies including linear daisy chains [5], bus networks [7] and tree networks [8]. For complex networks, the concept of equivalent networks was presented in [9]. There have been further studies in terms of load distribution policies for two and three dimensional meshes [10] and hypercubes [11]. In [12] the concept of time varying processor speed and link speed are introduced. In [13] the integration of monetary cost optimization and divisible load theory is presented. Scheduling policy research includes independent task scheduling [14,15], multi-installment sequential scheduling [16], multi-round algorithms [17], fixed communication charges [18], detailed parameterization and solution reporting time optimization [19] and combinatorial optimization [20]. An important reason for using divisible load scheduling theory is its tractability, flexibility and realism for a large class of data intensive, data parallel, computational problems.

In this paper equivalent continuous time Markov chain models [21,22,23] for various network topologies and load scheduling policies currently modeled by divisible load theory are introduced. Our initial motivation for introducing this unification between divisible load theory and Markov chain models is that they have a number of commonalities between them. In their basic form the two theories are linear ones. That is, they can be solved in theory by solving the associated linear set of equations. Other common features include a schematic language, recursive or linear equation solutions, the concept of equivalent networks, the possibility for time varying modeling and solutions for infinite size homogeneous networks.

This paper presents examples of this equivalence. While most of the Markov chains are one dimensional in topology, the labeling of transitions is different from the usual practice in queueing theory. This equivalence is somewhat surprising since divisible load theory is deterministic while Markov chain models are stochastic. This new equivalence provides a novel and apparently powerful modeling tool.

The remainder of this paper is organized as follows. In section II, basic model description and notation and definitions used through out this paper are presented. Sections III, IV and V discuss models for linear daisy chains, single level tree and two level tree networks, respectively. Finally, the conclusion is contained in section VI.

II. Model Description

In this section, some assumptions for scheduling in divisible load theory are described along with some notation and definitions. As mentioned earlier, the network topologies discussed in detail in this paper include linear daisy chains, and single and two level tree networks. The models discussed in this paper account also for both homogeneous and heterogeneous processing and link speeds and various load scheduling policies.

As mentioned earlier, it will be assumed that the total data parallel processing load is arbitrarily divisible into fractions of loads to be assigned to each processor over a network. The root processor where the total processing load originates, keeps some processing load for itself and sends out the rest of the load to the remaining processors over the network. There are different scenarios for the processors, depending whether

[1] This work was supported by NSF Grant CCR-99-12331.

or not they can compute and communicate at the same time. In general, we will consider two cases: with front end processors and without front end processors. In the case of processors with front end processors, it is assumed that some of the processors in the network are equipped with front ends so that they are able to compute their own load fraction and communicate (if necessary) simultaneously. In the case of networks without front end processors, it is assumed that none of the processors are equipped with front ends and the processors can only compute or communicate at one time. It is assumed that solution reporting time (back to the load originating node) is negligible compared to load distribution time and so is neglected. However, solution reporting time can be naturally modeled for divisible loads when necessary.

A Notations and Definitions:

α_i : The fraction of load that is assigned to processor i by the load originating processor.

ω_i : A constant that is inversely proportional to the computation speed of processor i in the network.

z_i : A constant that is inversely proportional to the speed of link i in the network.

T_{cp} : Computation intensity constant. This is the time that it takes the i^{th} processor to process the entire load when $\omega_i = 1$. The entire load can be processed on the i^{th} processor in time $\omega_i T_{cp}$.

T_{cm} : Communication intensity constant. This is the time that it takes to transmit the entire processing load over a link when $z_i = 1$. The entire load can be transmitted over the i^{th} link in time $z_i T_{cm}$.

T_i : The total time that elapses between the beginning of the scheduling process at $t = 0$ and the time when processor i completes its computation, $i = 0, 1, ..., n$. This includes, in addition to computation time, communicating time and idle time.

T_f : processing finish time of total processing load, assuming load is delivered to originator processor by $t = 0$. Naturally,

$$T_f = \max_i T_i.$$

In all of the sections the same definitions are used for α_i, ω_i, z_i, T_{cp}, T_{cm}, T_i and T_f unless otherwise stated. Another convention that is followed in is that the load originating at the root processor is assumed to be normalized to be a unit load.

III. Linear Daisy Chain Networks

Consider a linear daisy chain network consisting of $N + 1$ processors connected via N communication links as shown in Figure 1. The root processor P_0, where the load originates keeps its own share of load α_0 and communicates the remaining load (1-α_0) to its immediate successor P_1. Similarly, the processor P_1 keeps the load α_1 and communicates the remaining load (1-α_0-α_1) to its successor P_2. This process continues until the last processor obtains its share of load α_N. Each of the $N + 1$ processors in the network are equipped with front ends. That is, each processor can compute its own fraction of load and communicates the rest of the load to its successor simultaneously. Consider the case where each processor begins to compute its fraction of load at the moment that it finishes receiving its own data. An important observation in considering linear networks is that, each processor in the network (except for the last processor) receives the fraction of load that is not only its own share of the load but also the fraction of loads that belongs to all of the rest of the processors that are beyond it.

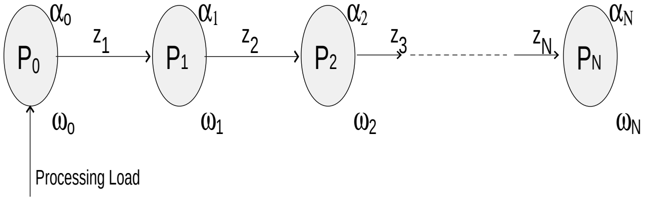


Figure 1: Linear daisy chain network with front end processors.

This process of communication and load distribution is shown through a Gantt-chart-like timing diagram in Figure 2. This protocol is referred to a “cut through switching” as the load fragment of a node’s right neighbor is retransmitted by the node to its right neighbor once it is received. That is, the node does not wait for the entire load to be received before commencing retransmission.

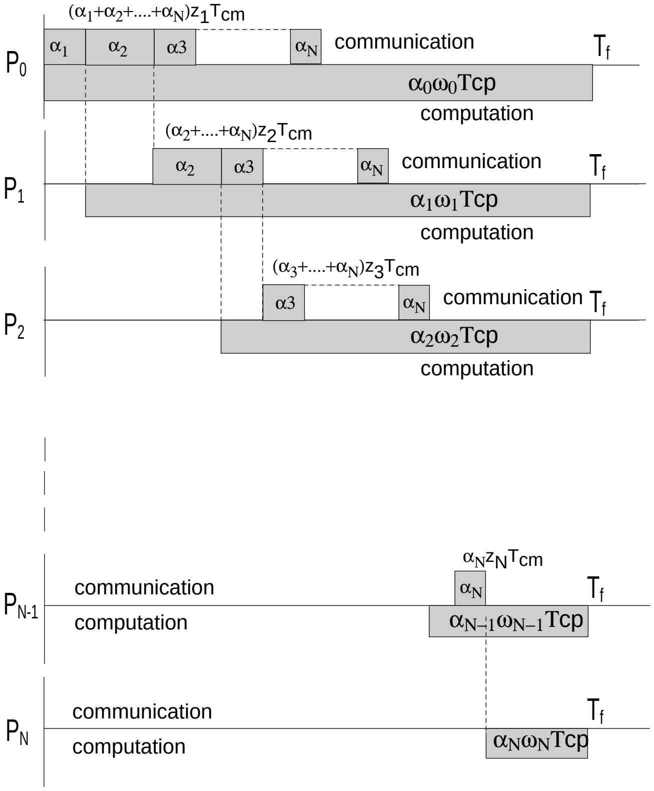


Figure 2: Timing diagram: linear daisy chain network.

As shown from the timing diagram, communication time appears above the axis and computation time appears below the axis. It is also shown that the processors have the same finishing time T_f. This corresponds to the fact that, for a minimum time solution all processors must stop computing at the same time. Indeed, otherwise some work could be transferred from a busy processor to an idle one in order to improve the solution time. Based on this result one can write the following set of equations:

$$\alpha_0\omega_0T_{cp} = \alpha_1z_1T_{cm} + \alpha_1\omega_1T_{cp} \quad (1)$$

$$\alpha_1\omega_1T_{cp} = \alpha_2(z_1+z_2)T_{cm} + \alpha_2\omega_2T_{cp} \quad (2)$$

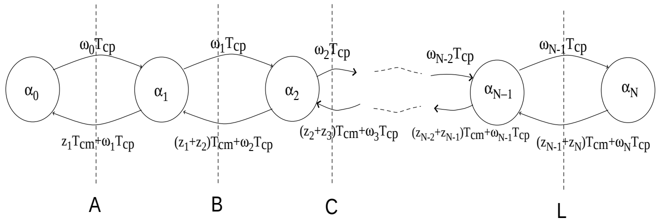
$$\alpha_2\omega_2T_{cp} = \alpha_3(z_2+z_3)T_{cm} + \alpha_3\omega_3T_{cp} \quad (3)$$

$$\alpha_{N-1}\omega_{N-1}T_{cp} = \alpha_N(z_{N-1}+z_N)T_{cm} + \alpha_N\omega_NT_{cp} \quad (4)$$

Since the load originating at the root processor is assumed to be normalized to a unit load, the fractions of the total processing load should sum to one.

$$\alpha_0 + \alpha_1 + \alpha_2 + ... + \alpha_{N-1} + \alpha_N = 1 \quad (5)$$

In this paper, the objective in presenting the above set of equations is to find a continuous time Markov chain model with α_i's being analogous to the steady state probabilities and the communication and computation time parameters being accounted for as transition rates that satisfy a set of local balance equations which corresponds to the above set of equations. In this case the Markov chain model which satisfies the requirements given above is shown in Figure 3.



ω_0T_{cp}
ω_1T_{cp}
ω_2T_{cp}
$\omega_{N-2}T_{cp}$
$\omega_{N-1}T_{cp}$
α_0
α_1
α_2
α_{N-1}
α_N
$z_1T_{cm}+\omega_1T_{cp}$
$(z_1+z_2)T_{cm}+\omega_2T_{cp}$
$(z_2+z_3)T_{cm}+\omega_3T_{cp}$
$(z_{N-2}+z_{N-1})T_{cm}+\omega_{N-1}T_{cp}$
$(z_{N-1}+z_N)T_{cm}+\omega_NT_{cp}$
A
B
C
L

Figure 3: Markov chain model for linear network with cut through switching and front end processors and heterogeneous links.

As shown in the Markov chain, using the local boundaries, one can write the following set of local balance equations as in the following manner. Using a balance equation at boundary A one can write:

$$\alpha_0\omega_0T_{cp} = \alpha_1z_1T_{cm} + \alpha_1\omega_1T_{cp} \quad (6)$$

At boundary B also one can write:

$$\alpha_1\omega_1T_{cp} = \alpha_2(z_1+z_2)T_{cm} + \alpha_2\omega_2T_{cp} \quad (7)$$

Similarly, for boundary L one can have the following equation:

$$\alpha_{N-1}\omega_{N-1}T_{cp} = \alpha_N(z_{N-1}+z_N)T_{cm} + \alpha_N\omega_NT_{cp} \quad (8)$$

This set of equations directly correspond to the set of equations that are derived from the Gantt-chart-like timing diagram. This new modeling tool combines both the equations and diagram into a Markov chain which is simple and compact. Note that the pattern of transition rate labeling is unusual for a Markov chain model in telecommunications and networking applications.

IV. Single level tree networks

A Single level tree without front end processors

A single level tree network of $N+1$ processors and N links is shown in Figure 4. The root processor P_0, where the load originates keeps its fraction of the total load for itself to compute and distributes sequentially the remaining load to its child processors at the lower level. Each processor in the network is assumed to have no front end processor. That is, the root processor will first finish communicating all of the load to be transmitted to the lower level before it starts computing its own fraction of load. The terminal processors start computing only after completely receiving their respective fraction of loads (known as staggered start). Note that if all z_i's have the same numerical value, one has a bus network model. The timing diagram showing the process of load distribution for a single level tree network with out front end processors is shown Figure 4.

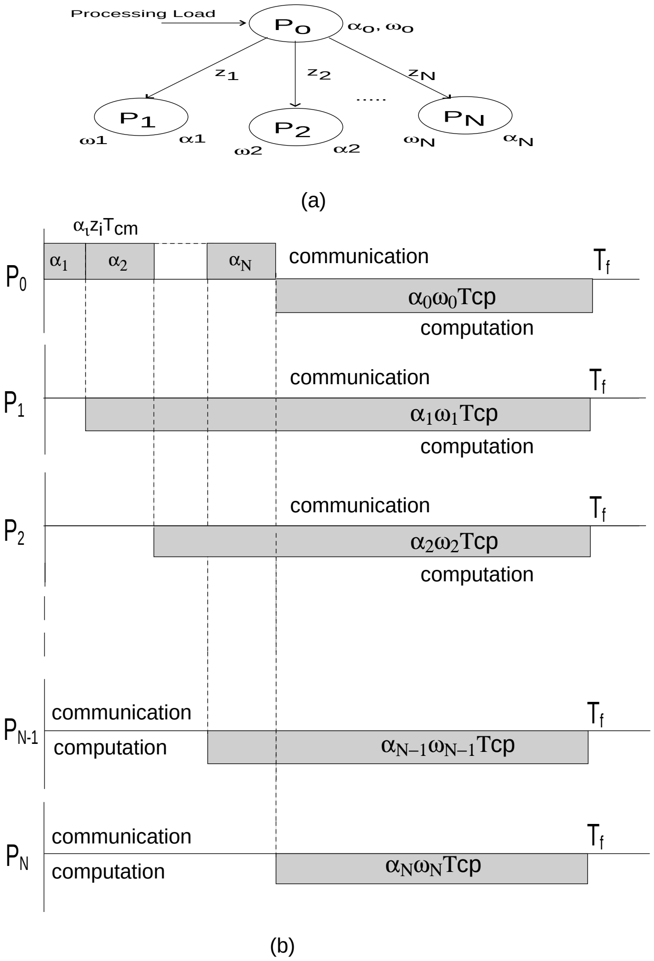


Figure 4: Single level tree network - (a) network topology (b)Timing diagram without front end at nodes.

Now one can write the following set of equations for solving for the optimal solution time as:

$$\alpha_0\omega_0 T_{cp} = \alpha_N\omega_N T_{cp} \tag{9}$$

$$\alpha_1\omega_1 T_{cp} = \alpha_2 z_2 T_{cm} + \alpha_2\omega_2 T_{cp} \tag{10}$$

$$\alpha_i\omega_i T_{cp} = \alpha_{i+1} z_{i+1} T_{cm} + \alpha_{i+1}\omega_{i+1} T_{cp} \tag{11}$$

$$\alpha_{N-2}\omega_{N-2} T_{cp} = \alpha_{N-1} z_{N-1} T_{cm} + \alpha_{N-1}\omega_{N-1} T_{cp} \tag{12}$$

$$\alpha_{N-1}\omega_{N-1} T_{cp} = \alpha_N z_N T_{cm} + \alpha_N\omega_N T_{cp} \tag{13}$$

Since the load originating at the root processor is assumed to be normalized to a unit load, the fractions of the total processing load should sum to one.

$$\alpha_0 + \alpha_1 + \alpha_2 + ... + \alpha_{N-1} + \alpha_N = 1 \tag{14}$$

The corresponding Markov chain model which has the same set of local balance equations as the above set of equations is shown in Figure 5. Note that state 0 is a neighbor of state N, which is unusual for a typical telecommunications Markov chain model.

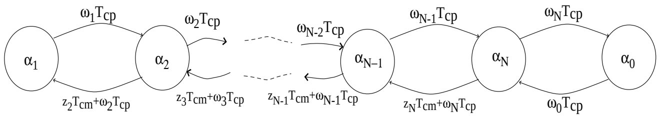


Figure 5: Markov chain model for single level tree network without front end processors.

B Single level tree networks with front end processors

A single level tree network of $N+1$ processors and N links is shown in Figure 6. The root processor P_0, where the load originates keeps its fraction of the total load for itself to compute and distributes the remaining load to its child processors at the next lower level sequentially. The root processor in the network is equipped with a front end. That is, the root can compute its own fraction of load and communicates the rest of the load to each of its children simultaneously. In this case each processor begins to compute its fraction of load at the moment that it finishes receiving its data. The timing diagram of the process of load distribution for a single level tree network with front end processors is shown Figure 6.

The set of equations for solving for the minimum finish time can be written as:

$$\alpha_0\omega_0 T_{cp} = \alpha_1 z_1 T_{cm} + \alpha_1\omega_1 T_{cp} \tag{15}$$

$$\alpha_1\omega_1 T_{cp} = \alpha_2 z_2 T_{cm} + \alpha_2\omega_2 T_{cp} \tag{16}$$

$$\alpha_i\omega_i T_{cp} = \alpha_{i+1} z_{i+1} T_{cm} + \alpha_{i+1}\omega_{i+1} T_{cp} \tag{17}$$

$$\alpha_{N-2}\omega_{N-2} T_{cp} = \alpha_{N-1} z_{N-1} T_{cm} + \alpha_{N-1}\omega_{N-1} T_{cp} \tag{18}$$

$$\alpha_{N-1}\omega_{N-1} T_{cp} = \alpha_N z_N T_{cm} + \alpha_N\omega_N T_{cp} \tag{19}$$

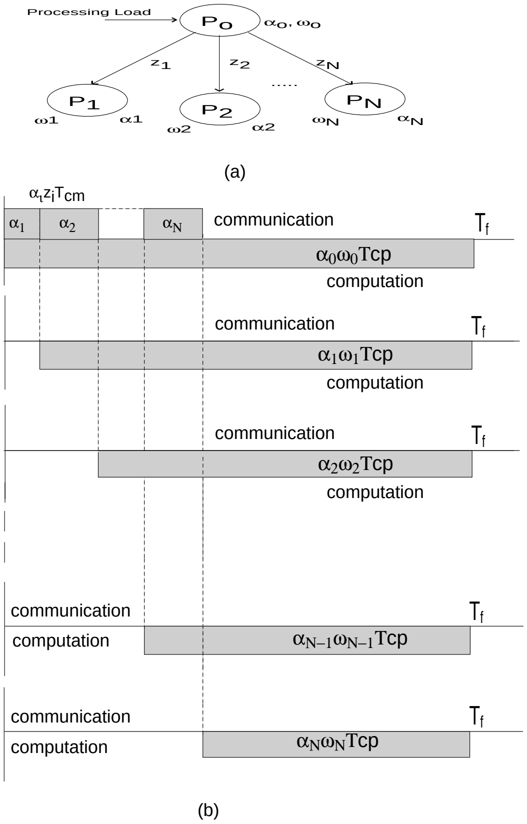


Figure 6: Single level tree network - (a) network topology (b)Timing diagram for front end case.

The fractions of the total processing load should sum to one.

$$\alpha_0 + \alpha_1 + \alpha_2 + ... + \alpha_{N-1} + \alpha_N = 1 \tag{20}$$

The corresponding Markov chain model which has the same set of local balance equations as the above set of equations is shown in Figure 7.

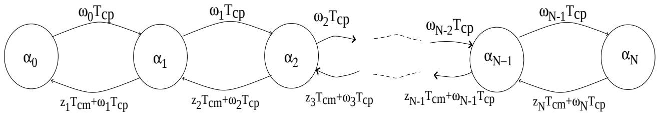


Figure 7: Markov chain model for single level tree network with front end processor at root.

V. Two level tree networks

Now consider a two level binary tree network of communicating processors as depicted in Figure 8. As can be seen from the figure, there are three types of nodes: *root*, *intermediate* and *terminal* nodes. The root is the node where the processing load originates. Then there are intermediate nodes which can be viewed as parents of the lower level nodes with which they have direct communication. The terminal nodes are nodes that have no child processors and hence can only be children nodes. In this section, we consider the case where the communication between the root processor and the intermediate nodes is concurrent and the communication between the intermediate nodes and the terminal nodes is sequential. As in section III, "cut through switching" is used at the intermediate nodes. These assumptions are made to show the ubiquity of the divisible load schedule and Markov chain equivalence. This process of load distribution and communication is shown in Figure 9.

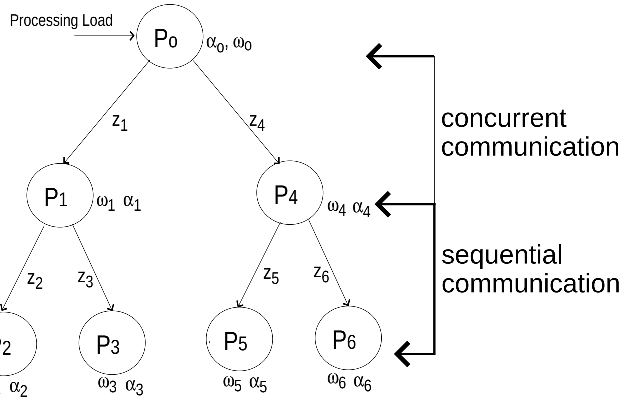


Figure 8: Two level tree network with front end processors.

The set of equations based on the minimum finish time can be written as:

For branch 1 (linking P_0 to P_1), one can write:

$$\alpha_0\omega_0 T_{cp} = \alpha_1 z_1 T_{cm} + \alpha_1\omega_1 T_{cp} \quad (21)$$

$$\alpha_1\omega_1 T_{cp} = \alpha_2(z_1+z_2)T_{cm} + \alpha_2\omega_2 T_{cp} \quad (22)$$

$$\alpha_2 z_2 T_{cm} + \alpha_2\omega_2 T_{cp} = \alpha_3(z_1+z_3)T_{cm} + \alpha_3\omega_3 T_{cp} \quad (23)$$

Similarly, for branch 4 (linking P_0 to P_4):

$$\alpha_0\omega_0 T_{cp} = \alpha_4 z_4 T_{cm} + \alpha_4\omega_4 T_{cp} \quad (24)$$

$$\alpha_4\omega_4 T_{cp} = \alpha_5(z_4+z_5)T_{cm} + \alpha_5\omega_5 T_{cp} \quad (25)$$

$$\alpha_5\omega_5 T_{cp} = \alpha_6 z_6 T_{cm} + \alpha_6\omega_6 T_{cp} \quad (26)$$

Note that the two branches 1 and 4 have, as is possible, two structurally different local balance equations based on the sequence of their fraction of loads. In the first case the fraction of loads are arranged in such a way that load fractions to children are in increasing order of load fragment size. Whereas in the second branch the load fractions to children are arranged in decreasing order of load fragment size.

The fractions of the total processing load should sum to one.

$$\alpha_0 + \alpha_1 + \alpha_2 + ... + \alpha_{N-1} + \alpha_N = 1 \quad (27)$$

The corresponding Markov chain model which has the same set of local balance equations as the above set of equations is shown in Figure 10. Note that state 0 is at the center of the Markov chain and states α_1, α_2, and α_3 which correspond to branch 1 are the left side neighbors of state 0. On the other hand states α_4, α_5, and α_6 which correspond to branch 4 are shown to be the right side neighbors of state 0. This is, again, unusual for the typical telecommunications Markov chain model.

VI. Conclusion

In this paper an alternative model for performance evaluation of divisible load schedules using Markov chain models is

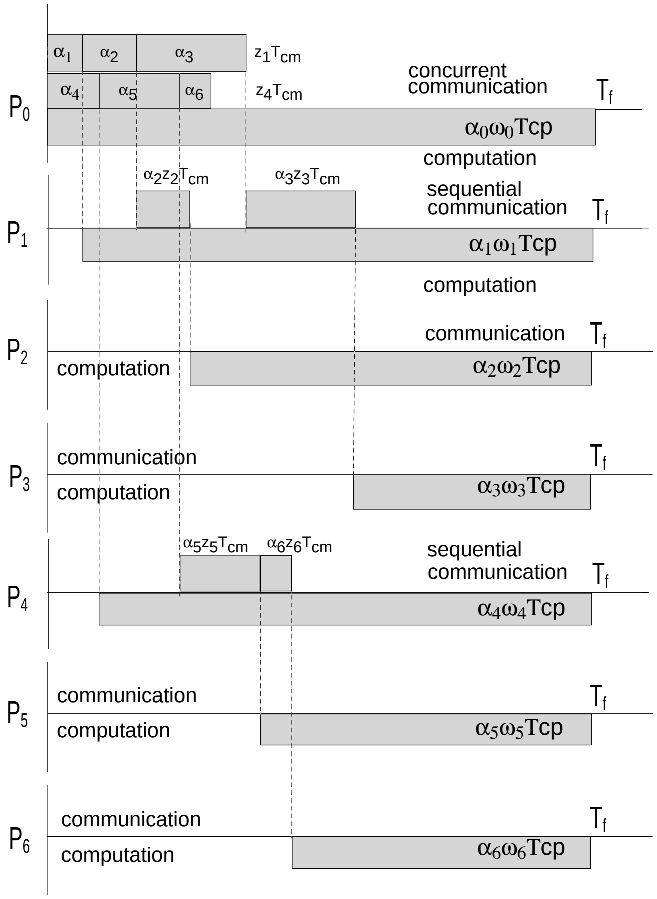


Figure 9: Timing diagram of a two level tree network with front end processors.

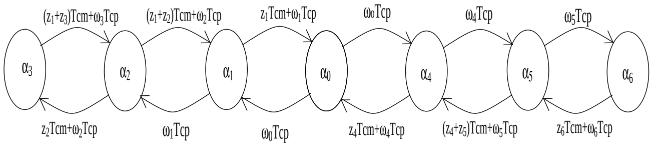


Figure 10: Markov chain model for two level tree network with front end processors.

presented. The model is based on the principle of local balance equations. We have provided Markov chain models for linear daisy chains, single level trees (with and without front end processors) and two level tree networks. It was found that these models are relatively simple and compact. Further areas of research would include multi level tree networks. Moreover it would be interesting to explore how to extend this result to networks that are more complex including hypercubes and two dimensional meshes.

References

[1] M. Moges and T.G. Robertazzi, "Optimal divisible load scheduling and Markov chain models," *Proceedings of the 2003 Conference on Information Sciences and Systems*, The Johns Hopkins University, Baltimore, MD, USA, 2003.

[2] 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.

[3] V. Bharadwaj, D. Ghose, T.G. Robertazzi, "Divisible load theory: A new paradigm for load scheduling in distributed systems," *Cluster Computing*, vol. 6, pp. 7–18, 2003.

[4] T.G. Robertazzi, "Ten reasons to use divisible load theory," *Computer*, vol. 36, pp. 63–68, 2003.

[5] 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.

[6] 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.

[7] S. Bataineh and T.G. Robertazzi, "Bus oriented load sharing for a network of sensor driven processors," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, pp. 1202–1205, 1991.

[8] Y.C. Cheng and T.G. Robertazzi, "Distributed computation for a tree network with communication delays," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 26, pp. 511–516, 1990.

[9] 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.

[10] 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.

[11] J. Blazewicz and M. Drozdowski, "Scheduling divisible jobs on hypercubes," *Parallel Computing*, vol. 21, pp. 1945–1956, 1996.

[12] J. Sohn and T.G. Robertazzi, "Optimal time-varying load sharing for divisible loads," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, pp. 907–923, 1998.

[13] 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.

[14] O. Beaumont, L. Carter, J. Ferrante, A. Legrand, and Y. Robert, "Bandwidth-centric allocation of independent tasks on heterogeneous platforms," *In International Parallel and Distributed Processing Symposium IPDPS'2002*, IEEE Computer Society Press, 2002.

[15] O. Beaumont, A. Legrand, and Y. Robert, "Optimal algorithms for scheduling divisible workloads on heterogeneous systems," *12th Heterogeneous Computing Workshops*, HCW'2003, 2003.

[16] V. Bharadwaj, D. Ghose, V. Mani, "Multi-installment load distribution in tree networks with delays," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 31, pp. 555–567, 1995.

[17] Y. Yang and H. Casanova, "UMR: A multi-round algorithm for scheduling divisible workloads," *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)*, Nice, France, 2003.

[18] J. Blazewicz and M. Drozdowski, "Distributed processing of distributed jobs with communication startup costs," *Discrete Applied Mathematics*, vol. 76, pp. 21–41, 1997.

[19] A.L. Rosenberg, "Sharing partitionable workloads in heterogeneous NOWs: greedier is not better. In D.S. Katz, T. Sterling, M. Baker, L. Bergman, M. Paprzycki, and R. Buyya, editors," *Cluster Computing*, pp. 124–131, 2001.

[20] P.F. Dutot, "Divisible load on heterogeneous linear array," *Proceeding of the International Parallel and Distributed Processing Symposium (IPDPS'03)*, Nice, France, 2003.

[21] W.J. Gordon and G.F. Newell, "Closed queueing systems with exponential servers," *Operations Research*, vol. 15, pp. 254–265, 1967.

[22] J.R. Jackson, "Networks of waiting lines," *Operations Research*, vol. 5, pp. 518–521, 1957.

[23] T.G. Robertazzi, "Computer networks and systems: Queueing theory and performance evaluation," *3rd edition*, Springer-Verlag, 2000.