International Journal of Mathematical Sciences
Vol. 9, No. 3-4, July-December 2010, pp. 441-452

A SYSTMATIC AND DEVELOPED ALGORITHM FOR TASK ALLOCATION SYSTEM IN COMPUTER COMMUNICATION SYSTEM

Manisha Sharma

Abstract: The Computer Communication System [CCS] has several challenging problems, in this paper we discussed and provide an optimal solution for assigning a set of "m" tasks of a program to a set of "n" processors where m > n. In Computer Communication Environment we maximize the overall throughput of the system and allocated load on all the processors should be balanced. The Inter Task Communication Time [ITCT] and Execution Time [ET] on different processors has consideration while preparing the task allocation model. The Task Allocation and Task Execution Time are presented by arrays i.e. Inter Task Communication Time Matrix [ITCTM (,)] and Execution Time Matrix [ETM (,)] respectively. This mathematical programming approach has been used to determine the optimal tasks allocation. The sets of several input data are considered to test the efficiency and complexity. It is found that the algorithm is suitable for arbitrary number of processor with the random program structure and workable in all the cases.

Keywords: Computer Communication System in Environment, Tasks Execution Time, Inter Task Communication Time, Task Allocation, Mathematical Programming

1. INTRODUCTION

The on-set of the microprocessor technology has made the Computer Communication System [CCS] economically viable and attractive for many applications of computer. However, many problem areas in CCS are still in their primitive development stages. Computer Communication System is increasingly drawing attention, yet has a meaning that is not understood.

The Computer Communication System is used to describe system terms with multiple processors. However, the term has different meanings to different systems because processors can be interconnected in many ways for various reasons. In the most general form, the word distribution implies that the processors are in geographically separate locations. Occasionally, the term is also applied to an operation using multiple mini-computers, which are not hardware, connected with each other and are connected through satellite. A user-oriented definition [1, 2] of distributed computing is "Multiple Computers, utilized cooperatively to solve problems".

Distributed processing applications range from large data base installations where processing load is distributed for organizational efficiency to high-speed signal processing systems where extremely fast processing must be performed in a real-time environment. The distributed real time environment in which, the services provided for the network reside at multiple site.

The total cost of execution of a distributed program consists of processor costs plus message transmission cost. The potential for distributed computing exists whenever there are several computers interconnected in some fashion so that a program or procedure running on one machine can transfer control to a procedure running on another. With increasing complexity of various real life problems, the demand for faster computer components is increasing.

Assigning tasks to processors is called task allocation, which involves the allocation of tasks to processors in such a way that some effectiveness measures are optimized. If the effectiveness measure can be represented as a linear function of several variables subjected to a number of linear constraints involving these variables, then the task allocation is classified as a Linear Programming Problem [LPP]. Likewise, for the processor, which can perform anyone the several tasks, possibly the difference of execution, and the effectiveness measure is the total ET to perform all tasks when one and only one task is allocated to each processor. In such cases, task allocation is classified as an assignment problem. Assigning "m" tasks to "n" processors, through exhaustive enumeration, results in n^m possible ways.

Splitting a program into small tasks and distributing them among several computing elements to minimize the overall system cost is one of the basic strategies adopted for performance enhancement of DCS. Several methods owing to Integer programming [3,4], critical delays consideration [5], branch and bound technique [6] and reliability evaluation [7] to deal with various design and allocation issues in a Distributed Processing Environment have been reported in the literature. These problems may be categorized in static (cf. Baca [8], Chu [9], PK [10], Singh [11-12], Peng *et al.* [13], Sager *et al.* [14], Singh *et al.* [15], Sirinivasan *et al.* [16], PK Av [17], Zahedi *et al.* [18]) and dynamic (cf. Bierbaum *et al.* [20] have suggested a dynamic model-based reliability analysis Bokhari[21], Casavent *et al.* [22], PK [23], Singh [25]) assignment problems. Rotithor [26] have been reported a general purpose taxonomy of dynamic task allocation in distributed computing. The present paper addressed an algorithm for systematic allocation of tasks in distributed computing environment keeping in view the allocated load on each processor should be balanced.

2. DEFINITION AND ASSUMPTIONS

Execution Time: The execution time e_{ij} ($1 \le i \le m$ & $1 \le j \le n$) of each task t_i depends on the processor p_j to which it is assigned and the work to be performed by each of tasks of that processor p_j.

Inter Tasks Communication Time: The Inter Task Communication Time c_{ik} of the interacting tasks t_i and t_k is incurred due to the data unit exchanged between them during the process of execution.

Processor Graph: Processor graph is a convenient abstraction of the processors together with interconnection. It has processors as nodes and there is a weighted edge between two nodes if the corresponding processors can communicate with each other. The weight w_{ij} on the edge between processors p_i and p_j represent the delay involved in sending or receiving massage of unit length from one to another. In order to have approximate estimate of this delay, irrespective of the two processors, we have considered the unit of the weights on all the edges in the processor graph. This is called the average unit delay.

Assumptions: Several assumptions have been made to keep the algorithm reasonable in size while designing the algorithm. The program is assumed to be the collection of "m" tasks, which are to be executed on a set of "n" processors, which have different processing capabilities. A task may be portion of an executable code or a data file. The number of tasks to be allocated is more than the number of processors (m>>n), as normally is the case in the real life distributed computing environment. It is assumed that the ET of a task on each processor is known, if a task is not executable on any of the processor due to absence of some resources, then The ET of same task on that processor is taken to be (—) infinite. We assume that once a task has completed its execution on a processor, the processor stores the output data of the task in its local memory, if the data are needed by some another task which being computed on the same processor, it reads the data from the local memory. The overhead incurred by this is negligible, so for all practical purposes we will consider it as zero. Using this fact, the algorithm tries to allocate the heavily communicating tasks to the same processor. Whenever a group of tasks is assigned to the same processor, the ITCT between them is zero.

3. PROBLEM STATEMENT

The specific problem being addressed is as follows:

Consider an application program that consists a set of **m** communicating tasks $T = \{t_1, t_2, \ldots. t_m\}$ and a DCS consisting a of set of **n** processors $P = \{p_1, p_2, \ldots. p_n\}$, interconnected by communication links, and it is assumed that "**m>>n**". The *processor graph* is a convenient abstraction of the processors together with interconnection network. It has processors as nodes and there is a weighted edge between two nodes if the corresponding processors can communicate with each other. The weight w_{ij} on the edge between processors p_i and p_j represent the *delay* involved in sending or receiving the message of unit length from one processor to another. In order to have an approximate estimate of this *delay*, irrespective of the two processors, we use the average of the weights on all the edges in the processor graph. This is called the *average unit delay*. The processing time e_{ij} of these tasks on all the processors is given in the form of Matrix ETM (,) of order m x n. The ITCT c_{ik} is taken in the form of a symmetric matrix named as ITCTM (,), which is of order m. In order to make the best use of the resources in a distributed computing system we would like to distribute the load on each processor in such a way that allocated load on the processors should be balanced.

4. PROPOSED METHOD

Since the number of task are more than the number of processors, so that it is required to form the order of the tasks for there execution by applying the equation (1) given below.

$$(RCE)_i = \frac{\sum_{j=1}^{m} c_{ij}}{\sum_{j=1}^{n} e_{ij}}, i = 1,2,.........m. \tag{1}$$

$(RCE)_i, i = 1, 2,.........m$ *is corresponding to the task* $t_1, t_2.......t_m$ //

Arrange the tasks in ascending order of their $(RCE)_i$ and store them in $T_{non-ass}\{\ \}$. Select first from task $T_{non-ass}$ { }, (say t_k) and check the minimum execution time of task t_k in ETM(,) of all the processors say $p_{i,}$ assign the task t_k to p_i and store the result in T_{ass}\{ \}. The processor position is also store in linear array Alloc{,}. The total allocated load on each processor is also stored in pload{ }. Select next task from the $T_{non-ass}$ { } say t_i, check the ITCT of task t_i with the assigned task stored in T_{ass}\{ \} say t_k also check the processor position of task t_k in Alloc{ } say p_i .If task t_i have the ITCT with t_k then assigned task t_i to that processor for which the sum of EC and ITCT and Processor load is minimum. If the task t_i has no inter task communication with the task t_k which is already assigned then assign the task t_i to that processor for which the sum of EC and processor load is minimum and then modified the pload{ }. This process is continuing until all the tasks get executed.

Calculate the exaction time and inter tasks communication time of each processor and store the result in a linear array pet(j) and pitct(j) respectively where j= 1,2,…n.

$$pet(\mathrm{j}) = \sum_{\mathrm{i}=1}^{\mathrm{m}} e_{ij}\ \mathrm{x}_{\mathrm{ij}},\ \mathrm{j} = 1,2,....\mathrm{n}$$

$$\left\{ \begin{array}{c} \text{Where } x_{ij} = 1, \text{ if } t_i \text{ and } t_j \text{ are on the same processor.} \\ 0, \text{ otherwise} \end{array} \right\}$$

And

$$pitct(\mathrm{j}) = \sum_{\mathrm{i=1}}^{\mathrm{m}} TTCT\ (\mathrm{i})\ \mathrm{x_i},\ j = 1,2,......n$$

$$\left\{ \begin{array}{c} \text{Where } x_i = 1, \text{ if } t_i \text{ is on the } j^{th} \text{ processor.} \\ 0, \text{ otherwise} \end{array} \right\}$$

Finally, sum up the value of pct(j) and pitct(j), (j=1,….,n) and store the result the tbtp(j) and pickup the maximum value of tbtp(j) i.e. tost called as total system optimal time.

The Mean Service Rate [MSR] of the processors in terms of T_{ass}(j) is then calculated by using the equation (2) and store the results in MSR(j) (where j = 1,2,…,n).

$$MSR(j) = \frac{1}{pet(j)}\ (where\ \mathrm{j} = 12,...n) \tag{2}$$

The overall mean service time and throughput of the processors are calculated by using the equation (3) and (4) respectively. Store the results of mean service time and throughout in the linear arrays MST (j) and TRP (j), where j=1, 2…….,n respectively.

$$MST(j) = \frac{1}{MSR(j)}\ (where\ \mathrm{j} = 12,...n) \tag{3}$$

$$TRP(j) = \frac{TTASK(J)}{PET(j)}\ (where\ \mathrm{j} = 12,...n) \tag{4}$$

5. IMPLEMENTATION OF THE METHOD

To justify the application and usefulness of the present method an example of a DCS is considered which is consisting of a set of “n = 3” processors P = {p_1, p_2, p_3} connected by an arbitrary network. The processors only have local memory and do not share any global memory. The processor connections graph is a depicted in figure-1 and tasks execution graph also pictorially depicted in figure 2. A set of “m = 8” executable tasks T = {t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8,} which may be portion of an executable code or a data file. The Inter tasks communication graph is depicted in figure 3.

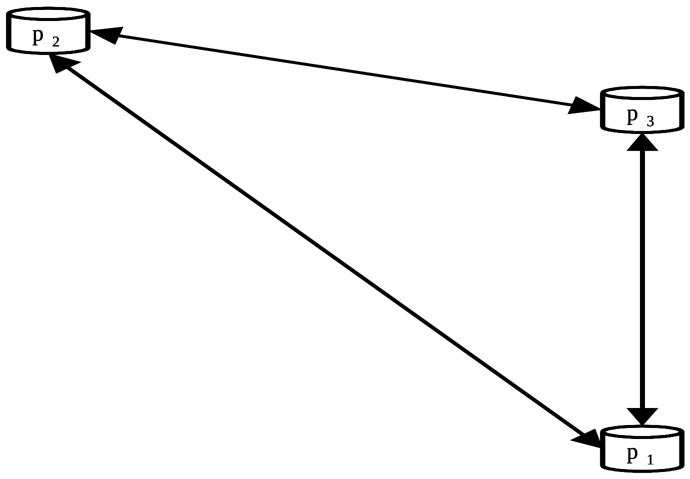
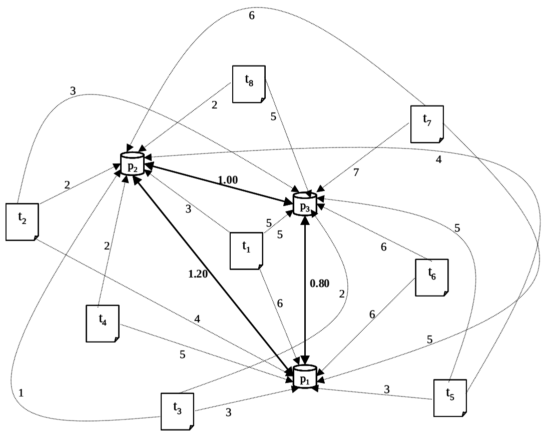


Figure 1: Processors Graphs



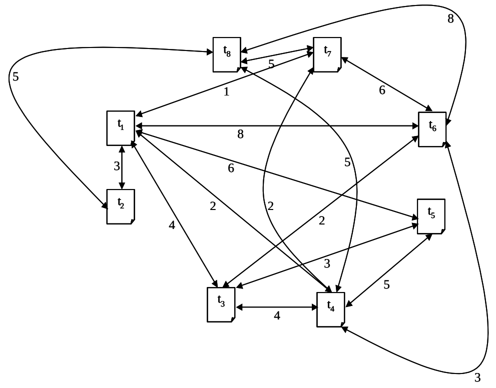


Figure 2: Tasks Execution Graph **Figure 3: Inter Tasks Communications Time Graphs**

Input of the Algorithm: Data required by the Algorithm is given below: Number of processors available in the system (n) = 3

Number of tasks to be executed (m)= 8

$$ECM(,) = \begin{array}{cccc} & p_1 & p_2 & p_3 \\ t_1 & 6 & 3 & 5 \\ t_2 & 4 & 2 & 3 \\ t_3 & 3 & 1 & 2 \\ t_4 & 5 & 2 & — \\ t_5 & 3 & 4 & 2 \\ t_6 & 6 & — & 6 \\ t_7 & 5 & 6 & 7 \\ t_8 & — & 2 & 5 \end{array}$$

$$ITCCM(,) = \begin{array}{ccccccccc} & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 \\ t_1 & 0 & 3 & 4 & 2 & 6 & 8 & 1 & 0 \\ t_2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ t_3 & 4 & 0 & 0 & 4 & 3 & 2 & 0 & 0 \\ t_4 & 2 & 0 & 4 & 0 & 5 & 3 & 2 & 5 \\ t_5 & 6 & 0 & 3 & 5 & 0 & 0 & 0 & 0 \\ t_6 & 8 & 0 & 2 & 3 & 0 & 0 & 6 & 8 \\ t_7 & 1 & 0 & 0 & 2 & 0 & 6 & 0 & 5 \\ t_8 & 0 & 5 & 0 & 5 & 0 & 8 & 5 & 0 \end{array}$$

The mathematical programming approach has been used to determine the optimal allocation of tasks. The optimization results from the algorithm ensure overall system cost as well as load on the processors are optimally balanced. Table-1 and figure – 4 are shows the optimal assignment of tasks to the processors. Table 2 shows the Results of the algorithm.

Table 1
Results of the algorithm

Tasks	*Processor*
t_3	p_1
t_7	p_1
t_2	p_2
t_4	p_2
t_8	p_2
t_1	p_3
t_5	p_3
t_6	p_3

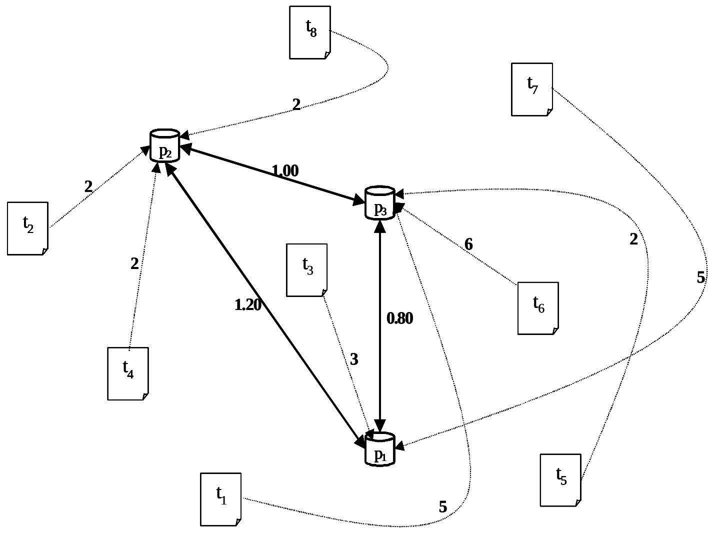


Figure 4: Optimal Assignment Graph

Table 2
Processors wise EC and ITCC and total of EC and ITCC

P	*EC*	*ITCC*	*MSR*	*TPP*	*MSR*	*T*
1	2	3	4	5	6	(3+6)
p_1	7	22	0.143	0.286	6.993	28.993
p_2	6	32	0.167	0.501	5.988	37.988
p_3	13	37	0.077	0.231	12.987	49.987

P–Processor, MSR–Mean service rate, TPP-Throughput of the processors, T-Total

The mean service rate and throughput of the processors are given in form of graph in figure – 5. The Maximum busy time of the system is 49.987, which is related to processor p_3 depicted in figure 6.

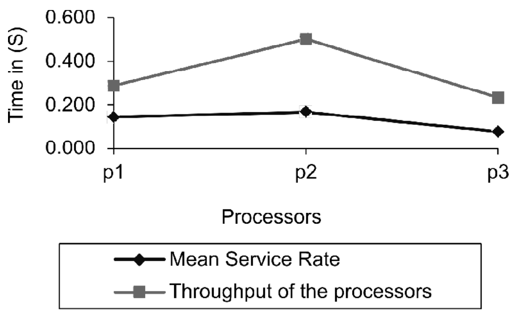


Figure 5: Mean service rate and throughput of the processors

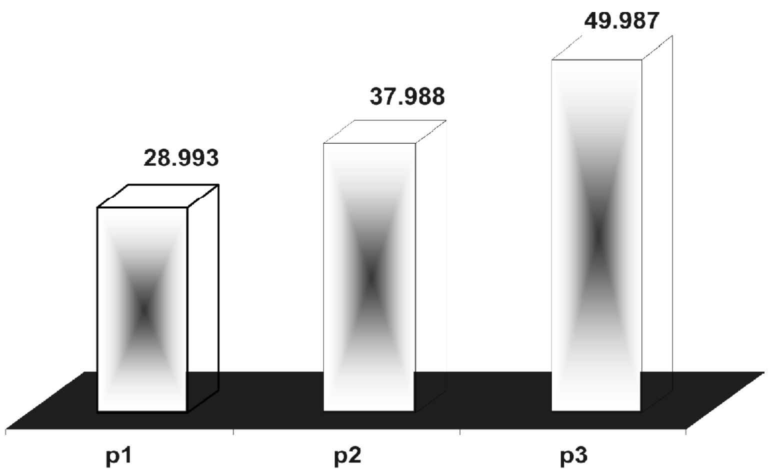


Figure 6: Maximum busy time of the system

6. CONCLUSION

The present paper deals with a simple yet efficient mathematical and computational algorithm to identify the *Systematic Allocation* of tasks for evaluation of performance of the Distributed Processing Systems number. A simple procedure has been developed to determine the following:

1. Systematic Allocation of tasks in DPS
2. Mean service rate,
3. Mean service time
4. Throughput of the processors

Table 1 shows that 2 tasks are executing on processor $p_{1,}$ 3 tasks are executing on p_3 and 2 tasks are executing on p_3. Table 2 shows that results of the algorithm form the table it is concluded that maximum busy time of the systems as 49.987 which is related to processor p_3. Therefore, the optimal time of the DPS is 49.987. Throughput of the processors is 0.286, 0.501 and 0.231. The average throughput of the DPS is 0.339.

The Performance of the algorithm is compared with [13]. The algorithm suggested in [13] is not considered the criteria of load balancing and proper utilization of each processor whereas our model considered both the issues. The run time complexity of the algorithm suggested by R.Y. Richard *et al.* [22] is o (n^m) which to high and the show the problem is NP-Hard. The algorithm suggested by G. Sagar *et al.* [13] runs o (m^2n). The run complexity of the algorithm presented in this paper is o $[1/2(5m^2+2mn)]$, which is much less then that of [13]. Table 3 and figure 7 represents the complexity comparisons of the algorithms.

Table 4
Results of run time complexity of the algorithms

M	*N*	*Run time complexity of the algorithms*	
		G. Sagar et al. [13]	*Present Model*
5	3	75.0	78.0
6	3	108.0	108.0
7	4	196.0	151.0
8	4	256.0	192.0
9	5	405.0	248.0
10	5	500.0	300.0
11	6	726.0	396.0
12	6	864.0	432.0
13	7	1183.0	514.0
14	7	1372.0	588.0
15	8	1800.0	683.0
16	8	2048.0	768.0

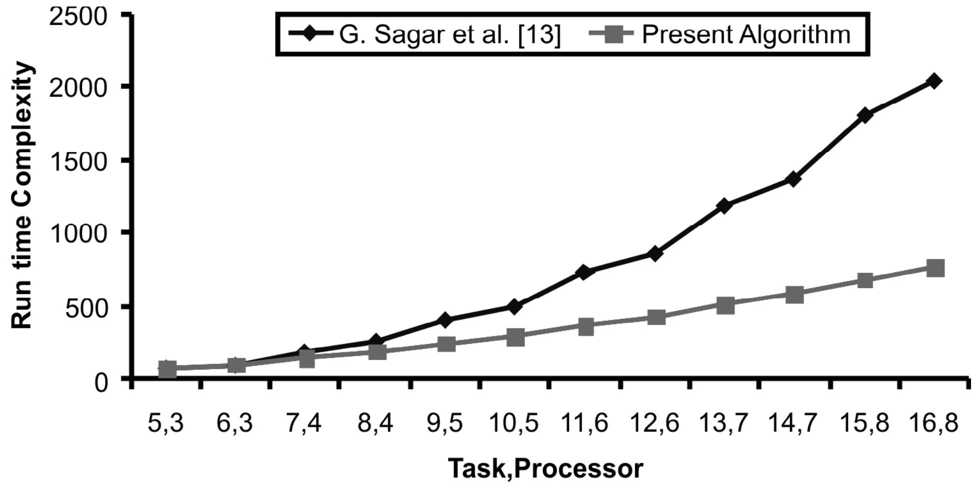


Figure 7: Comparisons of the complexity of the algorithms

It concluded that algorithm is general and can accommodate a large number of tasks to be assigned on any number of processors. To tests the generality of our algorithm the several sets of input data are considered and it is found that the algorithm is suitable for arbitrary number of processor with the random program structure and workable in all the cases.

REFERENCES

[1] Bhutani K.K., “Distributed Computing”, The Indian Journal of Telecommunication, pp. 41-44, 1994.

[2] Sitaram B.R., “Distributed Computing – A User’s View Point”, CSI Communications, Vol.-18 No. 10, pp.26,28, 1965.

[3] Chu W.W., “Optimal File Allocation in a Multiple Computing System”, IEEE Trans. On Computer, Vol.C-18 pp.885-889, 1969.

[4] Dessoukiu-EI O.I. and Huna W.H., “ Distributed Enumeration on Network Computers,” IEEE Trans. On Computer, Vol.C-29 pp.818-825, 1980.

[5] J.B. Sinclayer, “ Optimal Assignment in Broadcast Network” IEEE Trans. On Computer, Vol.37 (5), pp.521-351, 1988.

[6] Richard R.Y., Lee E.Y.S. and Tsuchiya M., “A Task Allocation Model for Distributed Computer System”, IEEE Tran. On Comp, Vol.C-31 pp.41-47, 1982.

[7] Min-Sheng Lin, , “A Linear-time Algorithm for Computing K-terminal Reliability on Proper Interval Graphs”, IEEE Trans. Reliability, vol. 51, pp. 58-62, 2002.

[8] Baca, D.F., “Allocation Modules to Processor in a Distributed System”, IEEE Transactions on Software Engineering, vol. 15, pp. 1427-1436, 1989.

[9] Chu, W.W., “Optimal File Allocation in a Multiple Computing System”, IEEE Transactions on Computer, vol. 18, pp. 885-889,1969.

[10] P K, Avanish, “An Algorithm for Optimal Index to Tasks Allocation Based on Reliability and cost”, published to the proceedings of International Conference on Mathematical Modeling held at Roorkee, pp. 150-155, 2001.

[11] Singh, P.K., “An Efficient Algorithm for Allocating Tasks to Processors in a Distributed System”, Proc. of the 19th National system conference, SSI, held at Coimbatore, India pp. 82-87, 1995.

[12] Singh, P.K., “A Fast Algorithm for Allocating Tasks in Distributed Processing System”, Proc. of the 30th Annual Convention of CSI, held at Hyderabad, India pp. 347-358,1995.

[13] Peng, Dar-Tezen, Shin, K. G. and Abdel, Zoher T. F., “Assignment Scheduling Communication Periodic Tasks in Distributed real time System”, IEEE Transactions on software Engg. vol. SE-13, pp. 745-757,1997.

[14] Sagar, G., and Sarje, A.K., “Task Allocation Model for Distributed System”, Int. J. System Science, vol. 22, pp. 1671-1678,1991.

[15] Singh, Kumar, A., “An Efficient Algorithm for Optimizing Reliability Index in Tasks-Allocation”, Acta Ciencia Indica, vol. xxv m, pp. 437-444,1999.

[16] Srinivasan, Santhanam and Jha. K. Niraj, “Safety and Reliability Driven Task Allocation in Distributed System”, IEEE Transactions on Parallel and Distributed Systems, vol. 10, pp. 238-250,1999.

[17] P. K. and Avanish, “An Efficient Static Approach for Allocation through Reliability Optimization in Distributed Systems”, presented at the International conference on Operations Research for Development (ICORD2002) held at Chennai, 2002.

[18] Zahedi, E., and Ashrafi, N., “Software Reliability Allocation based on Structure, Utility, Price and Cost”, IEEE Transactions on Software Engineering, vol. -17, pp. 345-356, 1991.

[19] V. Nefedova, R. Jacob, I. Foster, Z. Liu, Y. Liu, E. Deelman, G Mehta, M. Su, K. Vahi.”Automating Climate Science: Large Ensemble Simulations on the TeraGrid with the GriPhyN Virtual Data System”. *Presented at the eScience Conference in Amsterdam*, December, 2006.

[20] Bierbaum, Rene L., Brown, Thomas D. and Kerschen, Thomas J., “Model-Based Reliability Analysis”, IEEE Trans. on Reliability, vol. 51, pp. 133-140, 2002.

[21] Bokhari, S.H., “Dual Processor Scheduling with Dynamic Re-Assignment”, IEEE Transactions on Software Engineering, vol. 5, pp. 341-349, 1979.

[22] Casavent, T.L. and Kuhl, J. G., “A Taxonomy of Scheduling in General Purpose Distributed Computing System”, IEEE Transactions on Software Engineering, vol. 14, pp. 141-154, 1988.

[23] P K, Avanish, “Optimizing for the Dynamic Task Allocation”, published to the proceedings of the III Conference of the International Academy of Physical Sciences held at Allahabad, pp. 281-294, 1999.

[24] Gaurav Khanna, Umit Catalyurek, Tahsin Kurc, Rajkumar Kettimuthu, P. Sadayappan, Joel Saltz and Ian Foster “Multi-Hop Path Splitting and Multi-Pathing Optimizations for Data Transfer over Shared Wide-Area Networks using GridFTP” *Proceedings of the 17th IEEE International Symposium on High-Performance Distributed Computing (HPDC 2008)*, June 2008.

[25] Singh, P.K., “An Efficient Algorithm for Multi-processor Scheduling with Dynamic Reassignment”, Proc. of the 6th National seminar on theoretical Computer Science, held at Banasthally Vidyapeeth, India pp. 105-118, 1996.

[26] Rotithor, H.G., “Taxonomy of Dynamic Task Scheduling in Distributed Computing Systems”, IEEE Proc. Computer Digit Tech., vol. 14, pp. 1-10,1994.

[27] John Bresnahan, Rajkumar Kettimuthu, Michael Link and Ian Foster. “Harnessing Multicore Processors for High Speed Secure Transfer”. *Proceedings of the 26th IEEE Infocom’s High-Speed Networks Workshop*, May, 2007.

Dr. Manisha Sharma
Astt. Prof. / Punjab University
E-mail: manishatewaripu@gmail.com