ISSN No. 0976-5697

Volume 2, No. 3, May-June 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Remote Monitoring and Controlling Distributed Real-Time Systems using Multi-Agents

Anusha Kannan*
School of Information Technology and Engineering
VIT University,
Vellore, India
anusha.k@vit.ac.in

Usha Devi.G
School of Information Technology and Engineering
VIT University,
Vellore, India
ushadevi.g@vit.ac.in

Abstract: In recent years, research on software agents has gained tremendous amount of attention. To date, very little of the research has involved agents that operate within real time systems. An agent system would have certain unique characteristics versus traditional scheduling algorithms. This stems directly from agents ability to schedule execution times for a given task. The distributed systems are used in various application areas and their working environment is very complex due to which they face concurrency and synchronization problems. To cope up with these two problems in this paper we present a novel approach to enhance the efficiency of the distributed systems by implementing agent architecture and RMA scheduling algorithm. The main work of this article includes designing agent architecture for typical concurrency embedded system applications, and adapting classic RMA for scheduling tasks which have soft real time requirements.

Keywords: Distributed Real-Time Systems, Agents, RMA scheduling algorithm, Architecture Design.

INTRODUCTION

A distributed system is a collection of independent computers that appear to the users of the system as a single coherent system. There is always a need for effective monitoring, controlling and tracking of system components in a distributed real-time environment, as the clients are geographically distributed and are connected to the server through network. To give priorities to server response for various client requests in distributed systems agents are used.

Software agent [1] can be defined as a component of software and/or hardware which is capable of acting exactingly in order to accomplish tasks on behalf of its user [2] [3]. An agent is a component of hardware or software which is capable of acting exactingly in order to accomplish tasks on behalf of its user. In this issue, agent is deputy for the tasks to concurrency and synchronization. It rearranges the multi-access sequence for supporting as many parallel processing requests,[4] [5] and with specific scheduling algorithm [6] [7] such as Rate Monotonic Scheduling Algorithm.

Scheduling (RMS) algorithm [8] [9] [10] for satisfying time constraints of tasks can be used to solve multiple task scheduling. The indirect coupling architecture of agent can well adapt to meeting the requirements of parallel processing for better performance, and it has good extensibility for future multi-agent applications.

These agents act as an intermediate between clients and server. The concurrent and synchronized execution of multi-agents reduces the complexity of accuracy, time delay and synchronization between the components. Agents are also helpful in managing the complexity of the system, as by using them it is easy to define a system in terms of agent-mediated processes. Multi-agent technology offers a number of characteristics that make it well suited for distributed process monitoring and fault diagnosis tasks.

Scheduling is defined as the way processes are assigned to run on the CPU. To give priorities to the tasks in the real-time environment the basic scheduling algorithms like first come first serve (FCFS), shortest job first (SJF), round robin(RR) etc. are not efficient, So we haven choosen rate monotonic scheduling algorithm(RMS) over other scheduling algorithms. Rate Monotonic Scheduling presents one approach to addressing the problem of Distributed real-time systems. Rate Monotonic Scheduling (RMS) can be accomplished based upon rate monotonic principles. Rate Monotonic refers to assigning priorities as a monotonic function of the rate (frequency of occurrence) of those processes.

II. RMA SCHEDULING ALGORITHM

Rate Monotonic Scheduling presents one approach to addressing the problem of Distributed real-time systems. Rate Monotonic Scheduling (RMS) can be accomplished based upon rate monotonic principles. Rate Monotonic refers to assigning priorities as a monotonic function of the rate (frequency of occurrence) of those processes. In RMA scheduling algorithm, task execution is always consistent with rate monotonic priority: a lower priority task never executes when a higher priority task is ready i.e tasks with shorter periods are assigned higher priorities; no other criteria are considered for priority assignment. RMA is helpful for tasks that occur at a periodic rate even the task parameters can be changed quickly and easily to modify the system and the design of the system has become much easier by using it. Comparing to other scheduling algorithms RMA scheduling algorithm has less disadvantages so, it is very efficient for distributed real-time systems.

III. AGENT ARCHITECTURE FOR DISTRIBUTED REAL-TIME SYSTEMS

In distributed real-time environment clients are geographically distributed and connected to the server through network. The clients give the requests to the server through agents, then the server processes these requests and send back the results to the agent. The agent prioritize these results by using Rate Monotonic Scheduling Algorithm(RMS) and passes it to the clients.

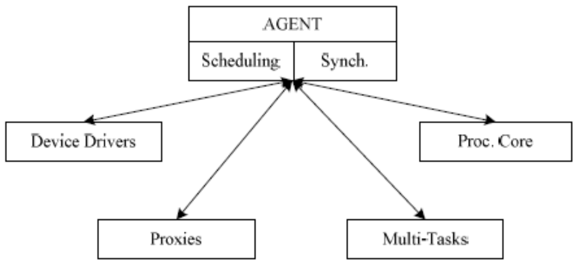

Figure 1 Agent Architecture in Distributed System

Software agent has evolved from multiagent systems. It disposes issues such as interaction and communication between processes, the scheduling and allocation of resources, negotiation and cooperation. Although multi-thread programming tool kits can implement such synchronization or concurrency even with priority support, it is still a tough work to fulfill parallel tasks spontaneously but at the same time guaranteeing meeting individual deadline constraints and integral optimal performance.

Software agent is a more prominent solution for such problems. According to the research of Steven [4], the directional kernel communication can be redesigned for an intermediate scheduling layer for better performance. While considering the time constraints of parallel processing tasks, the agent contains a task scheduler for deciding and arranging tasks activities sequence. Figure 2 illustrates the agent architecture design, and the agent is responsible for communication with different parallelizable tasks, allocation resources and collaboration between tasks.

Agent design is in software level, and would not change the hardware structure of embedded system. The agent schedules and synchronizes components that in charge of the concurrent tasks. Concurrent tasks with synchronization or real time constraints should register in the agent firstly. Then, after agent figuring out reasonable execution sequence, it synchronizes all tasks and schedules them in turn. Further the agent can avoid the heavy and complicated synchronization work amongst the processes and lowers shared resources access collision. The agent needs not to be persistent for economic resources usage, and it can be triggered by concurrent tasks request dynamically. When receiving new service requests, the agent will re-computes possible response time and decides whether or not they can be merged into current task group for scheduling.

The proposed system is a desktop search engine which searches web pages for a given keyword on user's PC .When clients send the requests all these requests are received by the agent, which then calculates the arrival time of each request(task).Now, these tasks are checked by the agent whether they are schedulable or not. If at the same time many tasks arrive then the agent schedules them by calculating the priority, execution time. If in between any new task arrives then it calculates the priority and reschedules the task accordingly. The agent sends all these reschedule tasks as input to search engine. Now, the search engine searches for files stored in the hard disk and it fetches the path name of the files in which the keyword is found in the form of links. These links can be clicked to open the files and view it. Also the results include additional information like the total number of times the keyword has occurred in that particular file. After every search if the keyword is found in any of the file in the specified location then the absolute path name of that file is saved in the database. In this way we create a search engine index which will be highly useful in optimizing the speed of the search performed. When a particular keyword is requested for search for the first time using this search engine then the algorithm searches the location of the files containing that particular keyword from the hard disk, next time when the same keyword is given the results are fetched from the index and thus saves time. Also any change in the hard disk file is reflected in the index also.

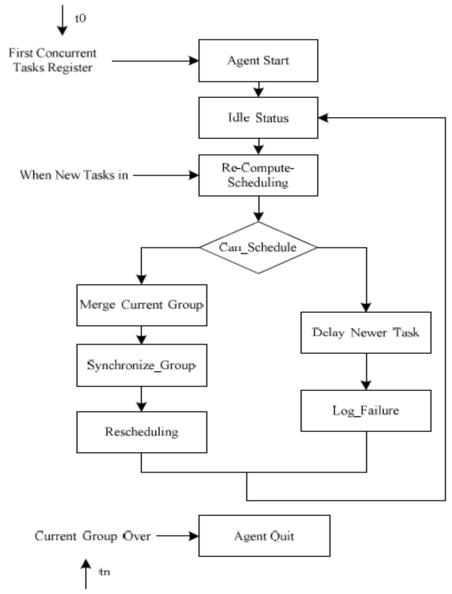

Figure 2 Architecture of Desktop Search Engine using Agents.

IV. SOFT REAL TIME SCHEDULING

As described above, the software agent should properly group the peripheral requests and enforce specific scheduling algorithm for concurrent tasks. Rate Monotonic Scheduling Algorithm was proposed by Liu [9] is a highly effective and widely used algorithm for real time application. According to Liu, supposing system consists of n tasks $S = \{t1, t2, \ldots, tn\}$. Also these tasks should be independent of each other. If this group of tasks is schedulable, it can satisfy the next equation,

$$\sum_{i=1}^{n} C_i / T_i \leq U(n) = n(2^{1/n} - 1) \qquad (1)$$

where *Ci* is the worst time cost and *Ti* is the period of task *i*. Tasks with shorter periods are assigned higher priorities with RMS algorithm.

Usually, the tasks are not completely independent because there is synchronization or cooperation. Then, RMS relaxes the previous constraint and gets the next equation.

$$\sum_{i=1}^{n} (C_i + B_i) / T_i \leq U(n) = n(2^{1/n} - 1) \qquad (2)$$

where *Ci* and *Ti* are the same as that of (1). *Bi* is the time that task *i* blocked by lower priority tasks. Case that lower priority processes block higher ones is said to priority inverse. Sprunt and Lehoczky [10] advanced this research to support non-periodical tasks scheduling. Considering two types of non-periodical tasks *TE* and *TR*, *TE* is a temporal emergent task, and its deadline is *DE*. TR is also non-periodical task but with little real time constraint. *CR* is the worst execution time for non -periodical tasks. In fact, *TR* is equivalence period for non-periodical cycle. Then, average response time consists of the average waiting time *Wq* and average execution time *We*

$$W = W_q + W_e \qquad (3)$$

$$\rho = D_q / I \qquad (4)$$

Dq is the time interval between continuous tasks quitting from queue. ρ is the tasks occurrence frequency which is used for evaluating the average CPU occupation. *I* denotes average tasks occurrence interval.

$$W_q = (\rho * D_q) / (2(1\text{-}\rho)) \qquad (5)$$

V. CONCLUSION

In summary, this paper has presented software agent architecture managing multi-tasks in typical embedded systems by designing a software agent architecture it was applied in internet environment for distribution tasks which will improve the concurrency and synchronization between the agents and also maintains the accuracy of data between the components of distributed embedded system components. The proposed method gives the best software agent which give good tracking and controlling of distributed embedded system components. The future work will focus on individual real time visualization reports which will be produced dynamically. In future agents are extended to support different inputs like Bluetooth, IR etc.

VI. REFERENCES

[1] Hu Jin1, Liang-Yin Chen2, Nian-Wei Chen1, Yang Lei1,"Software Agent Design with Real Time Scheduling for Embedded Systems", Chengdu Univ. of Information Technology, Chengdu 610041, Sichuan, China, 2009

[2] Nwana, H. S., Wooldridge, M., "Software Agents: An Overview", Knowledge Engineering Review, 1996, pp. 16-22.

[3] T.Abdelzaher, J.Stankovic, C.Lu, R.Zhang , and Y.Lu. " Feedback Performnace control in software services". IEEE Control Systems, 23(3), June 2003.

[4] Edward A. L. Stephen, N., "Actor-oriented Design of Embedded Hardware and Software Systems", Journal of Circuits, Systems, and Computers, Vol.2, No.3 , 2003, pp. 231-260.

[5] Robert P. Dick, Niraj K., "CORDS: hardware-software co-synthesis of reconfigurable real-time distributed embedded systems", IEEE/ACM International Conference on Computer Aided Design, 1998, pp. 62-68.

[6] James D. Monte and Krishna R., "Scheduling Parallelizable Tasks to Minimize Make-Span and Weighted Response Time", IEEE Trans. on Systems, Man, and Cybernetics – Part A: Ststemns and Humans, 2002, Vol. 32, No.3, pp. 335-345.

[7] Vercauteren, S. Lin, B. De Man, H., "A Strategy for Real-Time Kernel Support in Application-Specific HW/SW Embedded Architectures", in Proc. of the 33rd International Conference on Design Automation, Las Vegas, NV, USA, 1996, pp. 678-683.

[8] Fowler, P. Levine, L., "Technology Transition Push: A Case Study of Rate Monotonic Analysis (Part 1)", Technical Report CMU/SEI-93-TR-29 ESC-TR-93-203, 1993.

[9] Liu, C.L James W., "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment",Journal of the ACM, 1973, 20(1), pp. 46-61

[10] Lehoczky, J. Sha, L and Ye, D., "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior", In Proc. IEEE 10th Real-Time Systems

.