A Maintenance Task Scheduling System based on priority and criticality factor

Balamurugan Balusamy [#1], M S Ishwarya [*2]

[#1]Email id: balamuruganb@vit.ac.in
[*2]Email id:ishwaryaseenu@gmail.com
[#1*2]School of Information Technology and Engineering.
VIT University (www.vit.ac.in),Vellore, India

***Abstract*: It is so obvious that every site that went to critical effort of construction has to be maintained. The site can be of any place that explains earlier regardless of domain. This paper deals with a system that diagnoses maintenance problem in a site as well as, registers problems that are given by the user to it. Further, it deals with scheduling those problems for the next day maintenance schedule of the site. The complaints are forwarded department wise, each department is mailed with their problems as well as its details. Automatically, the problems are sensed through sensors while, manually, the problems are filed with custom designed applications that can be installed in the mobile devices. We propose a new automatic problem scheduling algorithm for performing the maintenance scheduling. The system automates the process of scheduling by keeping in to account several factors, got as input from sensors. And the sensor environment [1] is simulated using LABVIEW [2].**

Key Words: Scheduling, Maintenance, Site Monitor, Data Porter, Base Station

I. INTRODUCTION

Every site needs maintenance, irrespective of its size. Though the maintenance tasks are scheduled by maintenance head carefully, it is very normal to miss out or misrepresent some issues. So, we trace out these issues with sensors so there will be a clear representation of the problem producing entities. There will be some conditions that the problems escape from predetermined conditions of sensors, but still there is an inconvenience in the site. So we file user generated problems too, to schedule the maintenance tasks. Moreover, there will some zero priority jobs as well as parallel execution of jobs. We schedule some separately and integrate them with schedule in such a way that they are highlighted. The jobs that are left unclear for long time are forwarded to department head. As far as now, the schedule list or in other words, the job that needs clearing at most is the primary outcome of the base station (a component of the system).

II. METHODS

To maintain a site with methodology suggested in this paper we need some basic things to be checked. For a greater understanding we will divide the entire system into modules and each will be discussed for its importance. The entire system is divided into Site Monitor, Data Porter, Base Station, and Customer problem filing Application. In brief, site monitor will be installed in sites physically that acts as a watch for the normal as well as abnormal conditions in a site. Data Porter acts as a porter between Site Monitor and Base Station. Base Stations are the central stations that capture the issues from problems filed and schedule and forward them for clearance.

III. PROPOSEDMODULES

A. Site monitor:
This involves sensor based detection of problems in the working area. The sensor circuit is always being in touch with the site. It gives us the expected signals if there exist, a normal condition. It gives exceptional or expected signal under abnormal condition to the error monitoring server.

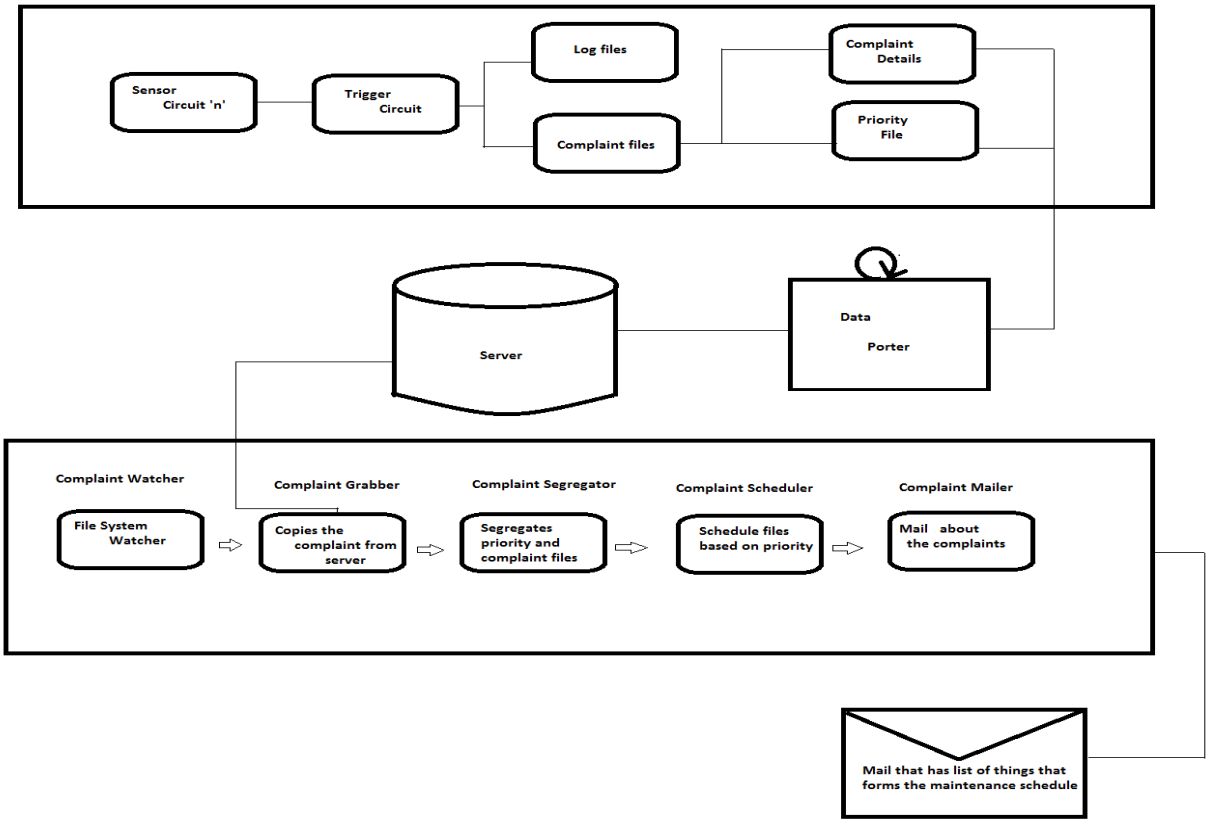
B. Data Porter:
The system is programmed in such a way that it checks for the problems in error directory in regular intervals and port them if any. If there an abnormal number of problem filing happening in a particular instance, Data Porter is demanded to port data rite at that time.

C. Base Station:
They capture the incoming data into their conscious and make them ready for feeding for scheduling algorithm. The algorithm takes them in and moulds them into a maintenance task schedule for the maintenance worker for the next shift. Each department head is forwarded with the complaints traced in that department related areas. The maintenance head is also made to know about the happenings.

D. App:

Additionally, there are tons of possibilities for a problem to escape from the threshold conditions. So, we provide an android application that allows the user either to choose from the options in it or to generate customized issues.

IV. DETAILED ARCHITECTURE



V. RELATED WORKS

The FCFS [25], the basic scheduling algorithm uses concept of least arrival time is services first. Whereas, the priority scheduling [3] uses a parameter called priority, to evaluate the job. Here, the priority expresses the criticality of the job in terms of numbers. Further, SJF [24] uses the concept that the job with least execution time is serviced first. The backfilling algorithm [4] does the job scheduling by giving the control to the jib that executes more than half of the resources. But, the Round Robin algorithm [22] gives equal chances to every job by giving control over the resources to every job on fixed time slices. Also, we have greedy algorithm [23] which follows problem solving techniques that could possibly give the best optimal solution. The multi-level queue algorithm [24] performs the task scheduling in two levels. Since, it combines two basic scheduling algorithms it gives better results than individuals.

VI. SCHEDULING ALGORITHM

Though every algorithm has its pros and cons, we suggest a new scheduling algorithm that aims at solving some unsolved problem. Our scheduling algorithm concentrates in parameter like priority, arrival time, and execution time[6]. We make use of these parameters to prepare the maintenance schedule.

Definition 1: A real world job j={e, p, a} is characterized by three parameters- priority I, arrival time a, execution time e. These jobs are executed in a routine (r) by taking control of resources over a temporary period of time. A real world instance is a collection of jobs J={j_1, j_2, j_3}

*Definition 2: (Schedule)*For any group of unscheduled task a uniprocessor schedule S is an alternate ownership of resource by two characteristic group of jobs.

S= {j_1, j_1', j_2, j_2', j_3, j_3'}

Group 1: Tasks that has higher priority is serviced is its order, provided their priorities do not go under the threshold (half of greater priority)

S1= { $j_1(p_1)$, $j_2(p_2)$, $j_3(p_3)$ } where p_1, p_2, p_3> P

Group 2: Tasks whose priorities are lesser than the half of the greatest priority at that instant, with arrival time (least) for their extra credit to acquire temporary ownership of the resources.

S2= $\{j_1'(p_1'),\ j_2'(p_2'),\ j_3'(p_4')\}$ where $p_1',\ p_{2,}',\ p_3'$< P/2

Definition (active job):

Case 1: A job J is said to be an active job, if it has highest priority

I. p>=P
II. execution time is least

*Case 2:*A job j is said to be a active job, if it has priority less than the half of greatest priority and with least arrival time

I. p<=P/2
II. a>= a'(least arrival time)
III. execution time is least

Parameter Definition: P, dynamic variable calculated every time of occurrence of new job.

P=p_i>{p_1, p_2, p_3…. p_n}

Algorithm:

This scheduling algorithm circulates a token between the groups mentioned above. They can grab the token alternatively to acquire ownership over the resources. When group grabs the token it can perform its case characteristic way of choosing jobs that has to be executed next.

	FCFS	SJF	RR	Priority	New Approach
Arrival Time	Yes	No	No	No	Yes
Execution Time	No	Yes	No	No	Yes
Priority Index	No	No	No	Yes	Yes
Deadline	No	No	No	No	Yes
Slots	No	No	Yes	No	No

Table 1: Comparative study of related algorithms with their attribute list[7]

S. no	No. of Samples	Execution Time
1	3	11.644
2	3	26.495
3	3	11.370
4	5	11.790
5	5	25.8855
6	5	11.790
7	10	46.840
8	10	39.727
9	10	36.915
10	20	116.494
11	20	111.252
12	20	80.924
13	50	203.821
14	50	193.210
15	50	204.103
16	100	881.731
17	100	808.456
18	100	799.123

Table 2: Execution time Vs Load

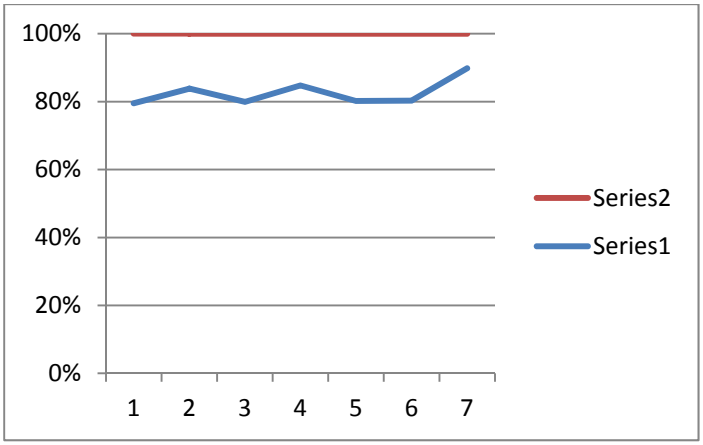


Fig 1: Graph That Shows the Execution Time with Load

The load of the algorithm is calculated with the tool developer c++ by bloodshed organization[8].

Time Complexity of the Algorithm:

Though the algorithm uses multiple strategies to elect correct process in its turn, it follows or grows linearly rather than exponentially. On calculating the time complexity it hits a minimal complexity of n.

VII.CASE STUDY

The better case study for this frame work will be using it for maintenance of street. Every street has and requires continuous basic facilities like street light, drainage clearance, power supply, dustbin clearance etc., Just imagine that every attributes listed above has a corresponding sensor attached to it. So, site monitor which works on continuous feedback from these sensors also monitors them based on their feedback. If there is problem/ complaint is generated at any of this feedback points, it is then ported to the base station through the data porters. Data porters are intermediate information carrying agent network. Base stations are installed in the centers like municipality or corporation of the city. These station are responsible for receiving these complaints, see their occurrence pattern, schedule men to solve this issue with the help of the proposed algorithm. The problems, if wished, can be escalated department wise also, which increases the speed of the action.

VIII. CONCLUSION

The paper concentrates only on scheduling of maintenance task and not any other MPP (Massively Parallel Processing) jobs. The suggested algorithm holds good for scheduling the jobs as per expected.

IX. REFERENCES

[1] Kininmonth, S., "Considerations in Establishing Environmental Sensor Networks," *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on* , vol., no., pp.687,691, 3-6 Dec. 2007
[2] Bhuvaneswari, P.T.V.; Raj, G.V.A.; Balaji, R.; Kanagasabai, S., "Adaptive Traffic Signal Flow Control Using Wireless Sensor Networks," *Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on* , vol., no., pp.85,89, 3-5 Nov. 2012.
[3] Bands, J.M.; Arenas, A.; Labarta, J., "Dual priority algorithm to schedule real-time tasks in a shared memory multiprocessor," *Parallel and Distributed Processing Symposium, 2003. Proceedings. International* , vol., no., pp.8 pp.,, 22-26 April 2003.
[4] Lawson, B.G.; Smirni, E.; Puiu, D., "Self-adapting backfilling scheduling for parallel systems," *Parallel Processing, 2002. Proceedings. International Conference on* , vol., no., pp.583,592, 2002http://www.bloodshed.net/devcpp.html
[5] http://en.wikipedia.org/wiki/Municipality
[6] http://www.bluetooth.com/Pages/Bluetooth-Home.aspx
[7] [22][23][24][25]
[8] http://www.bloodshed.net/devcpp.html
[9] http://en.wikipedia.org/wiki/Maintenance_engineering#Typical_Maintenance_Engineering_Responsibilities
[10] http://www.microsoft.com/visualstudio/eng
[11] http://msdn.microsoft.com/en-us/library/system.io.filesystemwatcher.aspx
[12] http://en.wikipedia.org/wiki/Lux
[13] http://en.wikipedia.org/wiki/GUI
[14] http://www.cdagro.com/CD5846/ep1.html
[15] http://en.wikipedia.org/wiki/Pentium
[16] http://en.wikipedia.org/wiki/Piconet
[17] http://en.wikipedia.org/wiki/Pseudocode
[18] http://www.android.com/
[19] Lupetti, S.; Zagorodnov, D., "Data popularity and shortest-job-first scheduling of network transfers," *Digital Telecommunications, , 2006. ICDT '06. International Conference on* , vol., no., pp.26,26, 29-31 Aug. 2006
[20] Batcher, K.W.; Walker, R.A., "Dynamic Round-Robin Task Scheduling to Reduce Cache Misses for Embedded Systems," *Design, Automation and Test in Europe, 2008. DATE '08* , vol., no., pp.260,263, 10-14 March 2008

[21] Singh, S.; Kant, K., "Greedy grid scheduling algorithm in dynamic job submission environment," *Emerging Trends in Electrical and Computer Technology (ICETECT), 2011 International Conference on* , vol., no., pp.933,936, 23-24 March 2011
[22] http://www.sciencedirect.com/science/article/pii/0895717796000283
[23] Wei Zhao; Stankovic, J.A., "Performance analysis of FCFS and improved FCFS scheduling algorithms for dynamic real-time computer systems," *Real Time Systems Symposium, 1989., Proceedings.* , vol., no., pp.156,165, 5-7 Dec 1989