Journal of Applied Sciences, 2012
ISSN 1812-5654 / DOI: 10.3923/jas.2012.
© 2012 Asian Network for Scientific Information

Efficient Scheduling for Data Processing in Large-scale Sensory Environments

[1]A. Alexandrescu, [2]F. Li, [2]S. Dustdar and [1]M. Craus
[1]Department of Computer Science and Engineering, "Gheorghe Asachi" Technical University, 27 Mangeron Blvd, Iasi, 700050, Romania
[2]Distributed Systems Group (DSG), Information Systems Institute, Vienna University of Technology, Argentinierstrasse 8, A-1040 Wien, Austria

Abstract: In recent years, there has been an emergence of systems that employ large numbers of sensory devices to collect information from the real-world and to react in observed situations, e.g., smart buildings and smart grids. In such systems, massive amounts of data need to be collected and processed in near real-time. This research proposes an efficient method of processing sensory data by considering that each data transmission and processing is a task which has to be scheduled for execution to a gateway. The proposed heuristic, Weighted Minimum Completion Time (WMCT), was compared against five mapping heuristics by using five efficiency indicators-execution time, makespan, Δ time difference, solution worth and success rate. The experiments have shown that the WMCT heuristic outperforms the other methods in terms of the success rate, the solution's worth and especially, the execution time of the algorithm, while it obtained good results in regards to the other two efficiency indicators.

Key words: Deadline, large-scale sensory environment, priority, sensor data, sensor-gateway system, task mapping, task scheduling

INTRODUCTION

Large deployments of sensors are used in many systems in order to acquire information on specific environments; for example, in smart buildings (Chen *et al*., 2009) where sensors monitor building functions, security and energy consumption, or in smart grids (Farhangi, 2010) where sensors send data to grid control and schedule centers. Many similar applications can be found in study of Firner *et al.* (2011) and Bimschas *et al.* (2010). This research considers a typical sensory architecture (Gupta *et al*., 2011) where middle components, namely gateways (Pacific Controls, 2010), are used to manage sensor connections, mediate protocols and process sensory data before it is sent to applications or back end databases. These sensor-gateway systems have to be capable of collecting massive data from a large number of sensors while processing the information in near real-time.

There are some research papers on task allocation in wireless sensor networks that do not consider the task's priority and deadline (Okhovvat *et al*., 2011; Zahariev and Hristov, 2011). These mapping techniques focus on minimizing the energy consumption and reducing the tasks' completion time. Momeni *et al.* (2009) proposed a new approach to task allocation in wireless sensor actor networks that considers the periodic tasks to be real-time jobs. Edalat *et al.* (2012) proposed an auction-based strategy to maximize the lifetime of wireless sensor networks by solving the distributed task allocation problem. Other research focuses on task scheduling and data transmission in wireless video sensor networks, while in study of Huang *et al.* (2009) an approach that uses intelligent agents and intelligent algorithms is used to solve the scheduling and data transmission problems.

The goal of this research was to determine an efficient method of processing sensory data in large-scale sensory environments by mapping the data with priorities and deadlines to processing gateways. The proposed approach considers that the data transmission and processing is a task which has to be mapped to a gateway. Therefore, the efficiency of sensory data processing can be improved by applying mapping heuristics. The gateways are considered to be black-boxes with highly-computational capabilities and from the task mapping perspective, they can simply be seen as machines that are able to execute tasks.

Corresponding Author: Mitică Craus, Department of Computer Science and Engineering, "Gheorghe Asachi" Technical University, 27 Mangeron Blvd, Iasi, 700050, Romania Tel: +40-232-278680 Fax: +40-232-231343

PROBLEM STATEMENT

Sensory data processing: The sensor-gateway system is a system in which multiple sensors are connected to a gateway network. There are many different types of sensors depending on their application (e.g., acoustic, thermal, proximity, optical). Each sensor periodically sends data in different formats and sizes through various protocols and the gateways must be able to process the received data accordingly and in the shortest time possible.

Four sensor data characteristics can be distinguished: priority, volume, send rate and receiving deadline. The information received from the sensors can have different priorities depending on the context in which the sensors are used and depending on their functionality. For example, in an art gallery, during the day, a humidity sensor has a higher priority than a motion sensor because a high humidity level can damage the paintings whereas the motion sensor is quite useless because of the flow of people who visit the gallery. In a typical situation where the sensors are functioning normally, each sensor sends different volumes of data at fixed periods of time. The data size of the information sent by the sensors depends on the sensor types. Applications can also spontaneously request data from sensors; in this case the sending rate can be higher than preconfigured rates. Some data is useful if it arrives and is processed in a predetermined period of time, otherwise the data becomes stale. This study considers that the data has to be processed no matter if it is stale or not (soft-deadlines). For example, a stale temperature data is invalid to a real-time monitoring system, but it is valid if a histogram of the registered temperatures is required.

The gateways have limitations on the maximum number of active connections, the maximum bandwidth and the processing power. In order to increase the capacity and the reliability, the gateways are interconnected. The system must dynamically assign the received data to the best gateways. The sensor data represents the volume of information sent by a sensor at a fixed period of time. This problem can be modeled as a task mapping problem in which independent tasks with priorities and soft-deadlines must be mapped to gateways for execution. In this study a task is considered to be composed of sending data to a gateway and processing the data.

In task mapping, the tasks are scheduled to run on machines and, usually, the mapping algorithm has an Expected Time to Compute (ETC) matrix which contains estimations of the time it would take to execute a task on a specific machine. This study considers that the expected time to compute can be obtained by estimating the execution time of the data processing algorithm located on each gateway depending on the available resources. Usually, the gateways are considered to be inter-connected via a high speed network.

In order to obtain an efficient sensory data processing method, the mapping heuristic must consider the particularities of the sensory environment. Previous mapping heuristics cannot be directly applied to the problem considered in this study because they are general approaches to task mapping that do not take into account the sensor data characteristics and do not focus on improving all the efficiency indicators. Only some mapping heuristics consider the task's priority and deadline, but they do not scale well because, in order to map a task, all the other unmapped tasks are evaluated. This leads to a significant increase of the execution time especially for a large number of tasks which is a problem in a large sensory environment.

Efficiency indicators: The mapping heuristics must efficiently assign and schedule n tasks to m gateways. Each task t_i ($i = 1 \ldots n$) has a corresponding priority p_i, deadline d_i and, also, an expected time to compute on each gateway (etc_{ij}). A gateway is usually a single-core industrial-grade computer that can execute only one task at a time, therefore a task will execute on its assigned gateway only after all the other previous scheduled tasks have finished their execution on that gateway. The gateway availability time (mat_j, where $j = 1 \ldots m$) is the time it takes for all the tasks assigned to that gateway to finish their execution. For a task that is going to be assigned to a gateway, the task completion time (tct_i) is the sum between the gateway availability time and the expected time to compute on that gateway. In order to evaluate the algorithms, the deadline factor (df_i) (Braun *et al.*, 2008; Kim *et al.*, 2007) was adapted to suit the considered situation:

$$df_i = \begin{cases} 1.00, & \text{if } tct_i \le d_i, \\ 0.50, & \text{if } tct_i \le d_i, \\ 0.25, & \text{if } tct_i \le d_i, \\ 0.05, & \text{otherwise.} \end{cases} \tag{1}$$

Also, Braun *et al.* (2008) defined the worth of a task as the priority multiplied by the deadline factor:

$$Worth_i = p_i \times df_i \tag{2}$$

Five efficiency indicators were used to compare the performance of the tested mapping heuristics:

- The most common method of evaluating the solutions of mapping heuristics is the makespan which is defined as the time it takes for all the tasks to finish their execution on their assigned gateways. This efficiency indicator is used in problems involving static mapping of independent tasks. The drawback of considering only the makespan is that it does not take into account the tasks' priorities and deadlines
- A mapping heuristic must assign the tasks so that the load is balanced across gateways, therefore the second efficiency indicator is the Δ time difference, defined as the difference between the time in which the first and the last gateway finish running their assigned tasks (Alexandrescu *et al*., 2011). If Δ is zero then the load is perfectly balanced across gateways
- The third indicator is the execution time of the mapping algorithm. The tasks have to be mapped as fast as possible, especially in dynamic task mapping situations. If, for example, a genetic algorithm is used to map the tasks, then some tasks might miss their deadline because of the time it takes to do the mapping
- The fourth indicator is the solution's worth which is an adaptation of the evaluation value from (Braun *et al*., 2008) and is defined as the sum of the tasks' worth. This indicator introduces penalties to the task's priority based on the task's completion time compared to its deadline.
- The considered problem requires that the maximum number of tasks with the highest sum of priorities complete their execution before their deadline. Once a task has missed its deadline it matters less when it finishes its execution as long as it is done in a reasonable amount of time. The last efficiency indicator is the success rate which represents the weighted percentage of tasks that finished before their deadline, i.e., the ratio between the sum of priorities of the tasks that finished before their deadline and the sum of all the task priorities

An ideal mapping heuristic must obtain, in the shortest time, the solution with the lowest makespan, the lowest Δ time difference and with the highest solution worth and success rate.

WEIGHTED MINIMUM COMPLETION TIME

The proposed mapping heuristic, Weighted Minimum Completion Time (WMCT), uses the Minimum Completion Time (MCT) method to map the tasks to gateways. The MCT algorithm has the advantage of a short execution time, but it has several disadvantages: the makespan and Δ time difference are higher compared to other mapping methods and the priority and deadline of a task are not considered in the mapping process.

To overcome this problem, all the tasks are sorted in descending order based on their weight:

$$\text{weight}_i = \frac{p_i}{e^{d_i}} \tag{3}$$

Then, the tasks are mapped in the order of their weight, while also factoring whether the tasks' deadline has been exceeded or not. The tasks with a higher weight will be more likely to be mapped before tasks with a lower weight. The weight of a task is higher for high priority tasks which must complete their execution sooner rather than later. The use of the exponential function results in a greater impact of the deadline on the weight, especially for large deadline values, i.e., the tasks with large deadline values are mapped later and also the priority has a smaller impact on the weight.

The steps of the WMCT algorithm are as follows. Firstly, all the tasks are sorted in descending order based on their weight. Next, the ordered task list is processed and each task is assigned to its minimum completion time gateway, only if the task finishes its execution before the deadline. If there are unassigned tasks, step two is repeated but only the tasks that finish before twice their deadline are mapped. If there are still unassigned tasks, step two is repeated but this time only the tasks that finish before four times their deadline are mapped. All the remaining tasks are mapped to their minimum completion time gateway. Lastly, all the tasks that finish after four times their deadline are reassigned. The reassignment consists of identifying the gateway that finishes its assigned tasks the last (g_p) and the one that finishes the earliest (g_q). The reassignment then finds the task from gateway g_p with the largest difference between the availability time of g_p and the availability time of g_q plus the expected time to compute of the task on gateway g_q and if the time difference is greater than zero, it moves the chosen task from g_p to g_q and tries to make another reassignment. The algorithm stops when no more reassignments can be made.

The time complexity for mapping n tasks to m gateways is O(n log n + nm), where the complexity for sorting the tasks is O(n log n) and the complexity for each of the other five steps of the algorithm is O(nm). The execution time of the algorithm can be further improved if the tasks are sorted by their weight as they arrive in the mapping queue, before the mapping algorithm runs.

The Weighted Minimum Completion Time heuristic is designed to obtain very good results for the five efficiency indicators. Efficiency Indicators. Because the algorithm considers only one task at a time in the mapping process, the execution time of the algorithm is very short compared to other algorithms, especially for a high number of tasks. A low value for the makespan and the Δ time difference is obtained by assigning the tasks to their minimum completion time gateway. Also, by first sorting the tasks based on their weight, the tasks with the highest priority and the tightest deadline are mapped first. This particularity, added to the fact that the tasks are mapped depending on how much they exceed their deadline, ensures a high solution's worth and a high success rate.

Step 6 reassigns only the tasks that missed their deadline, therefore, this step does not affect the solution's worth and the success rate efficiency indicators. The goal of the reassignment is to minimize the makespan and the Δ time difference by moving tasks from the gateways which finish the assigned tasks the last to the less loaded gateways. This last step is efficient if there are enough tasks that finish after four times their deadline.

EXPERIMENTAL RESULTS

Simulation setup: At fixed periods of time, each sensor sends a volume of data to be processed. This is translated into a task with priorities and deadlines that has to be assigned to a gateway and then executed. Because of the large number of sensors in the system there will most likely be moments when multiple tasks are waiting to be processed. This study addresses this issue and simulates such a situation, showing how efficient each tested mapping algorithm is in assigning the tasks that need to be processed at a point in time. Some tasks can originate from the same sensor, but, because of the environment's characteristics, most of the tasks come from different sensors in the sensor-gateway system.

The simulator uses a set of configurable parameters to generate the input data for the tested algorithms. The performance of each algorithm is computed by running the algorithm for a specified number of generated input data sets (1000 in this study) and by computing the mean of each efficiency indicator. The input data (i.e., the Expected Time to Compute matrix, the deadlines and the priorities of the tasks) are generated from a set of parameters which describe the sensors' behavior. The values for these parameters were chosen to best simulate a generic multi-sensor heterogeneous environment. In this study, all the random values were generated using a uniform random distribution.

The first two parameters are the number of sensors and the number of gateways to which the tasks will be assigned. For the tests in this study, between 100 and 1000 tasks (with increments of 100 tasks) are assigned to 10 gateways. Each sensor can send a random number of values to be processed (in this study, between 1 and 10) and each value can be processed in a time between 0.01 and 0.10 time units (t.u). The time it takes to process the information received from a sensor is calculated as the number of values being sent multiplied by the time in which a value is processed. The resulted time is an estimation of the execution time of a task on a gateway (processing time). A line in the ETC matrix represents the estimated execution times of a task on each gateway. Therefore, these times are calculated by multiplying the processing time by a random processing time variation factor between 1.0 and 1.5 which is a measure of the system's heterogeneity.

As is the case in a real sensor environment, in this study, some sensors send each time a fixed number of values to be processed while other sensors send a variable number of values. The interval for processing a value was chosen so that the values for the efficiency indicators are in an acceptable range and the differences between the tested algorithms can be easily seen. This study uses generic time units which can be seconds, tenths of a second or any time unit, depending on how much processing is done at the gateways for each piece of data received from the sensors. For example, at one time a sensor can send four values and each value can be processed in 0.7 t.u, resulting in an estimation of the execution time for the considered task of 2.8 t.u which can mean that the four values will be processed at the earliest in 2.8 sec.

When computing the deadline, the simulator considers the task's processing time and the elapsed time from the task's arrival. The deadline for each task is calculated as the sum between the task's processing time multiplied by a processing coefficient and a deadline constant. The processing coefficient is a random number between 1 and 10, because the deadline must be greater than and depend on the processing time, whereas the deadline constant is a random value between 0 and 10 t.u. This method of computing the deadline was chosen because in a real sensor environment, the tasks have to be processed in near real-time but also a specific time period can pass before the data becomes outdated. There is an acceptable fixed period of time in which the data has to be processed (in this study, the deadline constant) but the deadline also depends on the sensor that sends the data and on the volume of data that needs to be processed (the task's processing time multiplied by a processing coefficient).

In order to simulate the degree of importance of the sensor data, each task was assigned, with the same probability, a high, medium or low priority level as in (Kim *et al*., 2007). For the tests in this study the priority level was respectively represented by the weights 4, 2 and 1.

Other mapping heuristics: In order to evaluate and compare the proposed solution five existing mapping heuristics were applied to the problem considered in this study. These heuristics were designed to improve the solution's worth efficiency indicator. These methods were adapted to achieve better results regarding the other four efficiency indicators for a more relevant comparison between these heuristics and the proposed WMCT method. Next are presented the changes that were made to the five mapping heuristics.

The Two-Phase Fitness (TPF) heuristic described by Braun *et al*. (2008) was slightly modified because, in this study's context, the tasks do not have dependencies, hard deadlines and multiple versions. The proposed adaptation of the algorithm uses the task's worth from Eq. 2 to determine the best gateway. This allows the tasks to be mapped even after their deadline has been missed, but they will incur a penalty given by the deadline factor df_i.

The other four considered heuristics, i.e., Rescheduling Min-Min, Relative Cost, Slack Sufferage and Max-Max, are described by Kim *et al*. (2007). Optimizations were made to improve the execution time of the algorithms in order to have a fairer comparison with the proposed approach. Also, in this study the Relative Cost heuristic maps only the task with the lowest relative cost, among all the highest worth tasks, to its minimum completion time gateway. This was done to obtain a better makespan and Δ time difference, a higher solution's worth and a higher success rate at the expense of a slightly higher execution time of the mapping heuristic.

The Max-Max heuristic assigns tasks to the gateway with the maximum fitness which was defined in this study as:

$$\text{fitness}_i = \begin{cases} \text{worth}_i / \text{etc}_{ij}, \text{if } df_i \geq 0.25, \\ \text{worth}_i / \text{mat}_{ij}, \text{otherwise} \end{cases} \quad (4)$$

The proposed approach to this heuristic maps the tasks that finish before four times their deadline to the gateway with the shortest execution time and the tasks that finish after that to the gateway with the shortest completion time. This method of computing the fitness improves the makespan and the Δ time difference while it does not influence the other efficiency indicators.

RESULTS

The performance of the proposed mapping heuristic, Weighted Minimum Completion Time, was compared against five other heuristics, i.e., Two-Phase Fitness, Rescheduling Min-Min, Relative Cost, Slack Sufferage and Max-Max. The results obtained based on the five efficiency indicators. Efficiency Indicators-namely, the execution time of the algorithm, the makespan, the Δ time difference, the solution's worth and the success rate-for each of the six heuristics are presented below.

The means of the execution times for each of the six tested algorithms are presented in Table 1. Regardless of the number of tasks that had to be mapped to the ten gateways, the proposed algorithm, WMCT, mapped the tasks the fastest. For 1000 tasks the proposed algorithm did the mapping in 4.39 sec which was more than nine times faster than the second fastest algorithm, Max-Max which mapped the tasks in 40.68 sec. The least effective of the six algorithms were Two-Phase Fitness and Slack Sufferage which were 32 times slower than WMCT (more than 142 sec).

The second efficiency indicator used to evaluate the results was the makespan. Table 2 shows the mean of the values obtained for the makespan over 1000 iterations. One of the goals of the algorithms was to obtain a low makespan value. The Relative Cost heuristic produced the solutions with the best makespan for all the tested number of tasks. On the other hand, the worst makespan was

Table 1: The execution time (sec) for six mapping heuristics each running for 1000 ETC matrices and mapping 100 to 1000 tasks to ten gateways

	# tasks									
Algorithm	100	200	300	400	500	600	700	800	900	1000
Two-Phase Fitness	1.33	5.42	12.25	22.15	34.53	49.71	67.90	89.19	113.00	143.48
Rescheduling Min-Min	0.49	1.85	4.11	7.33	11.47	16.28	22.33	29.17	37.27	46.73
Relative Cost	0.69	2.75	5.98	10.91	16.78	23.82	32.22	43.69	54.71	69.35
Slack Sufferage	1.36	5.49	12.04	22.05	33.83	48.42	65.81	89.15	112.29	142.19
Max-Max	0.43	1.57	3.35	6.13	9.40	13.36	18.16	25.31	32.07	40.68
WMCT	**0.21**	**0.41**	**0.65**	**0.97**	**1.36**	**1.81**	**2.30**	**2.90**	**3.55**	**4.39**

ETC: Expected time to compute, WMCT: Weighted minimum completion time, Underline row is proposed method, Bold values are best values

Table 2: The makespan (time units) for six mapping heuristics each running for 1000 ETC matrices and mapping 100 to 1000 tasks to ten gateways

	# tasks									
Algorithm	100	200	300	400	500	600	700	800	900	1000
Two-Phase Fitness	3.83	7.04	10.37	13.69	17.05	20.32	23.62	26.84	30.05	33.32
Rescheduling Min-Min	3.85	7.03	10.24	13.40	16.61	19.76	22.97	26.12	29.26	32.46
Relative Cost	**3.57**	**6.71**	**9.88**	**13.00**	**16.13**	**19.24**	**22.44**	**25.57**	**28.71**	**31.91**
Slack Sufferage	3.64	6.77	10.01	13.25	16.53	19.74	23.01	26.22	29.41	32.69
Max-Max	4.93	8.01	10.98	13.96	16.98	19.94	23.09	26.22	29.41	32.66
WMCT	3.95	7.40	10.87	14.30	17.73	21.01	24.41	27.80	31.20	34.69

ETC: Expected time to compute, WMCT: Weighted minimum completion time, Underline row is proposed method, Bold values are best values

Table 3: The Δ time difference (time units) for six mapping heuristics each running for 1000 ETC matrices and mapping 100 to 1000 tasks to ten gateways

	# tasks									
Algorithm	100	200	300	400	500	600	700	800	900	1000
Two-Phase Fitness	0.74	0.57	0.60	0.70	0.81	0.89	0.93	0.95	0.99	1.00
Rescheduling Min-Min	1.03	1.07	1.09	1.10	1.11	1.11	1.12	1.11	1.12	1.12
Relative Cost	**0.50**	**0.44**	**0.39**	**0.30**	**0.18**	**0.12**	**0.09**	**0.07**	**0.06**	**0.05**
Slack Sufferage	0.74	0.63	0.67	0.71	0.75	0.76	0.75	0.76	0.77	0.78
Max-Max	3.23	2.82	2.15	1.71	1.39	1.11	1.00	0.95	0.98	1.01
WMCT	0.90	0.93	0.95	0.87	0.76	0.45	0.26	0.19	0.15	0.11

ETC: Expected time to compute, WMCT: Weighted minimum completion time, Underline row is proposed method, Bold values are best values

Table 4: The solution's worth for six mapping heuristics each running for 1000 ETC matrices and mapping 100 to 1000 tasks to ten gateways

	# tasks									
Algorithm	100	200	300	400	500	600	700	800	900	1000
Two-Phase Fitness	228.9	437.8	626.1	795.1	951.4	1090.8	1221.3	1342.4	1459.5	1567.9
Rescheduling Min-Min	228.4	437.6	624.8	790.9	941.3	1075.2	1200.7	1316.2	1425.4	1525.1
Relative Cost	228.1	430.2	606.4	756.7	886.4	996.4	1096.2	1181.5	1260.3	1330.0
Slack Sufferage	230.4	444.6	640.0	817.8	982.6	1131.4	1274.2	1408.3	1537.7	1662.9
Max-Max	232.6	456.9	663.8	851.6	1025.6	1183.4	1331.9	1470.0	1602.9	1725.8
WMCT	**233.3**	**466.4**	**699.4**	**918.5**	**1107.5**	**1272.2**	**1426.0**	**1565.6**	**1693.3**	**1806.0**

ETC: Expected time to compute, WMCT: Weighted minimum completion time, Underline row is proposed method, Bold values are best values

obtained by the Max-Max heuristic for less than 400 tasks and by the WMCT heuristic for more than 400 tasks. For 100 tasks, the best makespan was 3.57 t.u (Relative Cost) and the worst was 4.93 t.u (Max-Max), while, for 1000 tasks, the best makespan value was 31.91 t.u (Relative Cost) and the worst was 34.69 t.u (WMCT).

The algorithms were also evaluated on how well the load was distributed across gateways. The load is well balanced if the Δ time difference is minim. As it was the case with the makespan, the Relative Cost algorithm obtained the lowest Δ time difference as can be seen in Table 3. The Δ time difference obtained by the WMCT algorithm for 1000 tasks was twice the value obtained by the Relative Cost heuristic (0.11 t.u compared 0.05 t.u), but it was also seven times better than the third best algorithm, Slack Sufferage which was 0.77 t.u and more than ten times better than the worst algorithm, Rescheduling Min-Min which was 1.12 t.u. For 100 tasks, the best Δ time difference was 0.50 t.u (Relative Cost) and the worst 3.23 t.u (Max-Max).

The results regarding the fourth efficiency indicator, the solution's worth, are shown in Table 4. Higher values signify a better solution. For all the tested situations, the proposed algorithm, WMCT, was the best, followed by Max-Max, while the Relative Cost algorithm was last. For 100 tasks the best worth was a value of 228.1 (Relative Cost), while the best was 233.3 (WMCT). For 1000 tasks, the WMCT solution's worth had a value of 1806 which was 4% better than the solution's worth of the Max-Max heuristic (with a worth of 1726) and 26% better than the solution's worth of the worst algorithm, Relative Cost (with a worth of 1330).

The last efficiency indicator is a measure of how many tasks finished before their deadline based on their priority. The values obtained for the success rate are presented in Table 5. The best values were obtained by the WMCT algorithm, followed by Max-Max, Slack Sufferage, Two-Phase Fitness, Rescheduling Min-Min and, lastly, Relative Cost; this order was maintained regardless of the number of tasks to be assigned. As the number of tasks increased, the success rate decreased because there were more tasks that needed to be mapped to the same number of gateways. The WMCT algorithm obtained an almost 100% success rate for less than 300 tasks and had a 33% decrease in the success rate for 1000 tasks (67.9%). The worst success rate was obtained by the Relative Cost algorithm, 96.7% for 100 tasks and 43.8% for 1000 tasks.

Table 5: The success rate for six mapping heuristics each running for 1000 ETC matrices and mapping 100 to 1000 tasks to ten gateways

	# tasks									
Algorithm	100	200	300	400	500	600	700	800	900	1000
Two-Phase Fitness	97.3	91.4	85.4	79.7	74.6	69.9	65.7	62.0	59.1	56.5
Rescheduling Min-Min	96.9	91.0	84.7	78.5	72.8	67.9	63.7	60.0	57.0	54.2
Relative Cost	96.7	88.6	80.9	73.4	66.5	60.6	55.6	50.9	47.1	43.8
Slack Sufferage	98.3	93.7	88.5	83.3	78.3	73.5	69.4	65.8	62.7	60.3
Max-Max	99.5	96.9	92.5	87.5	82.8	78.5	74.5	70.9	67.8	65.0
WMCT	**100.0**	**100.0**	**100.0**	**97.2**	**91.1**	**85.3**	**80.3**	**75.8**	**71.8**	**67.9**

ETC: Expected time to compute, WMCT: Weighted minimum completion time, Underline row is proposed method, Bold values are best values

Table 6: The mean percentages of tasks that finished before their deadline based on their priorities (low, medium and high) for six mapping heuristics each running for 1000 ETC matrices, for 400, 700 and 1000 tasks

	400 tasks			700 tasks			1000 tasks		
Algorithm	low	medium	high	low	medium	high	low	medium	high
Two-Phase Fitness	12.6	21.8	32.4	3.1	13.5	30.8	0.2	7.9	28.9
Rescheduling Min-Min	11.0	21.5	32.3	1.5	12.3	30.6	0.2	5.7	28.7
Relative Cost	9.7	18.1	31.3	2.4	9.3	27.1	0.9	4.6	23.0
Slack Sufferage	13.0	24.1	**33.2**	2.4	13.2	**33.3**	0.9	3.2	**33.3**
Max-Max	21.1	27.7	31.9	**12.7**	21.8	29.4	**8.7**	**17.7**	26.9
WMCT	**27.2**	**33.2**	**33.2**	5.3	**25.0**	33.0	1.4	14.7	31.9

ETC: Expected time to compute, WMCT: Weighted minimum completion time, Underline row is proposed method, Bold values are best values

Another interesting aspect is how each algorithm mapped the tasks based on their priority. Table 6 shows the percentages of tasks that finished before their deadline for 400, 700 and 1000 tasks; for less than 300 tasks the results are not that relevant because most of the tasks finished before their deadline. From the total number of tasks to be mapped, a third had a high priority, a third had a medium priority and a third had a low priority. Regardless of the number of tasks, the Slack Suffrage algorithm mapped the most high priority tasks (almost all of them), followed closely by the WMCT algorithm. Max-Max mapped the most medium priority tasks for 1000 tasks (17.7%), whereas WMCT mapped the most for 400 and 700 tasks (27.7% and, respectively, 29.4%). The number of low priority tasks that finished before their deadline was higher for fewer tasks because most of the higher priority tasks were also mapped. WMCT mapped the most low priority tasks for 400 tasks (21.1%), but the Max-Max mapped the most for 700 and 1000 tasks (12.7% and, respectively, 8.7%). For a high number of tasks there were fewer tasks that could finish before their deadline, therefore significantly more high priority tasks were mapped compared to the number of low priority tasks. From 1000 tasks, a total of 48.0% of tasks finished before their deadline by mapping with the WMCT algorithm, whereas, by mapping the tasks with the Max-Max algorithm, a total of 53.3% of tasks finished before their deadline.

DISCUSSION

The task mapping problem has been widely studied in different scenarios. The basic approach is to statically map a set of independent tasks to heterogeneous machines. Heuristics like Opportunistic Load Balancing, Minimum Execution Time, Minimum Completion Time, K-percent Best, Min-Min, Max-Min or Suffrage are the most well known methods of mapping independent tasks. A comparison of these heuristics was performed by Braun *et al.* (2001) and Maheswaran *et al.* (1999) or Alexandrescu and Craus (2010). Alexandrescu *et al.* (2011) proposed a genetic algorithm to improve the makespan and the Δ time difference at the expense of a higher execution time. The mutation operator described in the aforementioned research was the basis for the last step of the mapping heuristic presented in this study. The main disadvantage of these methods is that they do not consider the task's priority and deadline.

Some approaches to the task mapping problem start from a candidate solution (which can be generated by another mapping heuristic) and try to improve it. Such a heuristic is presented by Attiya and Hamam (2006), where a simulated annealing approach is used to map tasks in order to maximize the reliability of a distributed system. The difference, compared to this research, is that the simulated annealing heuristic considers inter-dependent tasks and it does not take into account the deadline for processing a task.

Braun *et al.* (2008) proposed three algorithms for static resource allocation of tasks having dependencies, priorities, deadlines and multiple versions. Two of the used heuristics are a genetic algorithm and a GENITOR-style algorithm, but these take longer to run and are not suitable in a dynamic mapping environment. Also, a two-phase fitness algorithm which is based on the Min-Min heuristic, is presented. The main difference compared to this study is that the context does not involve dependencies, multiple versions and hard deadlines.

The Two-Phase Fitness (TPF) algorithm used in this study is an adaptation of the algorithm from Braun *et al.* (2008). The proposed WMCT heuristic outperformed the TPF method regarding the execution time of the algorithm. WMCT was more than six times faster than TPF for 100 tasks and more than 32 times faster for 1000 tasks; this was due to the much greater scalability of the WMCT heuristic. Regardless of the number of tasks that needed to be mapped, the only efficiency indicator at which WMCT obtained poorer results compared to TPF was the makespan. TPF produced solutions with a 3% to 4% lower makespan than WMCT. Regarding the Δ time difference, the TPF algorithm was better for less than 500 tasks, whereas the proposed WMCT obtained lower time differences for 500 or more tasks. With the increase in the number of tasks that need to be mapped, the WMCT heuristic showed a descending trend for Δ time difference value, as opposed to the TPF heuristic which obtained increasingly poorer results. The values for the solution's worth and the success rate were in favor of the WMCT heuristic regardless of the number of tasks. The WMCT heuristic obtained a 15% better solution's worth compared to the TPF heuristic for 1000 tasks and an 11% higher success rate.

The work from Kim *et al.* (2007) presents and compares eight dynamic mapping heuristics which use independent tasks with priorities and multiple soft-deadlines. The dynamical aspect of the mapping heuristic is obtained by using mapping events which are triggered when a new task arrives and there is no mapping process currently running and then applying a static heuristic for the tasks waiting to be executed. The same principle can be applied to the method used in this study in order to dynamically process the data using the proposed heuristic.

A significant difference, compared to this study, is that their research uses a different method of generating the expected time to compute matrix and another way of generating the deadline for each task which does not take into account the characteristics of a sensor-gateway environment. In this study, the proposed method of generating these values is much better suited for a sensory environment. Sensory Data Processing. The methods of computing the matrix and deadline have an important impact on the performance of the mapping algorithms. Another difference is that in this study uses the makespan, the Δ time difference, the execution time of the algorithm and the success rate, besides a variation of the evaluation method from Kim *et al.* (2007) to more accurately determine the efficiency of a mapping heuristic.

Four adaptations from the aforementioned research were used for comparison against the proposed WMCT heuristic-i.e., Rescheduling Min-Min (RMM), Relative Cost (RC), Slack Suffrage (SS) and Max-Max. The results obtained in this study regarding these algorithms are similar to the results from the aforementioned research. The execution times of each of the four algorithms were poorer than the execution time of the WMCT algorithm, regardless of the number of tasks that needed to be mapped. This was due to a lower time complexity compared to the other heuristics; also, the WMCT algorithm scaled significantly better. As the number of tasks that needed to be mapped increased, each algorithm followed its increasingly higher trend and the hierarchy of the algorithms regarding the execution time was the same throughout the simulation. For 1000 tasks, the WMCT algorithm mapped all the tasks in 4.39 sec, whereas the second best algorithm, Max-Max, did the mapping it a time that was almost ten time slower (40.68 sec).

In regards to the makespan, the WMCT heuristic outperformed the Max-Max heuristic for 100, 200 and 300 tasks. Apart from these cases, WMCT performed poorer than the tested algorithms. WMCT obtained a 10% higher makespan mean than the best heuristic (Relative Cost) for 100 tasks and 8.7% higher for 1000 tasks. The reason for the poorer makespan values is that the WMCT heuristic is based on the Minimum Completion Time heuristic which does not take into consideration all the remaining tasks when assigning a task to a gateway. The final step of the WMCT algorithm improves the makespan by reassigning the tasks from the most loaded gateways, but it is not enough to outperform the other tested algorithms. For less than 500 tasks, the Δ time difference obtained by WMCT was worse than Slack Suffrage, Two-Phase Fitness and Relative Cost, but for more than 500 tasks WMCT was outperformed only by Relative Cost which obtained the lowest time difference for all the tested number of tasks. For 1000 tasks, the WMCT heuristic obtained a Δ time difference of 0.11 t.u, whereas Relative Cost obtained a value 0.05 t.u and the third best algorithm, Slack Suffrage, obtained a value of 0.78 t.u, while the worst values were obtained by Rescheduling Min-Min (1.12 t.u). Regarding the Δ time difference, the Relative Cost, Max-Max and WMCT algorithms showed a descending trend with the increase of the number of tasks, while the other three algorithms, Two-Phase

Fitness, Rescheduling Min-Min and Slack Suffrage obtained slightly higher time differences as the number of tasks that had to be mapped increased. The influence of the final step of the WMCT algorithm which tries to level the load across gateways, was more noticeable for over 500 tasks and its efficiency increased proportionally to the number of tasks that had to be mapped. The Relative Cost and the WMCT were the only algorithms that systematically improved their Δ time difference for over 200 tasks.

Although the other heuristics focus solely on improving the solution's worth, the proposed WMCT algorithm outperformed those heuristics regardless of the number of tasks. Also, WMCT obtained the highest success rate among all the tested algorithms. The success rate difference between WMCT and the second best algorithm, Max-Max, increased until 400 tasks to a difference of 9%, but then started to decrease to only 3% for 1000 tasks. This suggests that, for more than 1000 of tasks, the Max-Max algorithm might outperform WMCT regarding the success rate. The other algorithms obtained lower success rates, the Relative Cost algorithm being the worst with only 43.8% success rate, compared to WMCT's 67.9% for 1000 tasks. The optimal performance of the WMCT heuristic in terms of the solution's worth and the success rate is due to the fact that the proposed algorithm maps the tasks depending on how likely they are to finish before their deadline.

Overall, the WMCT algorithm was by far the best regarding the execution time of the algorithm, the Relative Cost algorithm was best at obtaining a low makespan and a low Δ time difference and, even though it mapped fewer tasks that finish before their deadline compared to the Max-Max, the WMCT algorithm also obtained the best solution's worth and the highest success rate.

CONCLUSIONS

An important aspect of sensor-gateway systems is efficiently assigning sensory data to processing gateways. In this study, the data transmission and processing are seen as a task with priorities and soft-deadlines which has to be mapped to a gateway. The efficiency of an algorithm for mapping independent tasks with priorities and deadlines depends on the number of tasks and gateways, the method of generating the expected time to compute matrix and, especially, the tasks' deadline.

This study proposes a mapping heuristic called Weighted Minimum Completion Time which was compared to five adaptations of the best task mapping heuristics. There are several advantages of using the WMCT heuristic to dynamically map tasks with priorities and soft-deadlines. The execution time scales significantly better with regard to the number of tasks compared to the other tested algorithms, and, even though it obtains a poorer makespan, the load becomes more balanced with the increase of the number of tasks that have to be mapped. The WMCT heuristic does not always have the most tasks that finish before their deadline but it obtains the best solution's worth and the highest success rate.

The proposed heuristic is efficient in the sense that it can handle much better the arrival of a high number of tasks in the system (task spikes), compared to the other tested heuristics, due to its greater scalability and because it obtains the highest success rate in the shortest time. Taking into consideration all the above, the WMCT heuristic is an efficient method for processing sensory data in large-scale sensory environments.

ACKNOWLEDGMENTS

This study was supported by the EURODOC "Doctoral Scholarships for research performance at European level" project, financed by the European Social Found and the Romanian Government and the Pacific Controls Cloud Computing Lab (PCCCL) at Vienna University of Technology.

REFERENCES

Alexandrescu, A. and M. Craus, 2010. Improving mapping heuristics in heterogeneous computing. Proceedings of the 6th European Conference on Intelligent Systems and Technologies, October 7-9, 2010, Iasi, Romania, pp: 1-12.

Alexandrescu, A., I. Agavriloaei and M. Craus, 2011. A genetic algorithm for mapping tasks in heterogeneous computing systems. Proceedings of the 15th International Conference on System Theory, Control and Computing, October 14-16, 2011, Sinaia, Romania, pp: 1-6.

Attiya, G. and Y. Hamam, 2006. Task allocation for maximizing reliability of distributed systems: A simulated annealing approach. J. Parallel Distrib. Comput., 66: 1259-1266.

Bimschas, D., H. Hellbruck, R. Mietz, D. Pfisterer, K. Romer and T. Teubler, 2010. Middleware for smart gateways connecting sensornets to the internet. Proceedings of the 5th International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks, November 29-December 3, 2010, Bangalore, India, pp: 8-14.

Braun, T.D., H.J. Siegel, N. Beck, L.L. Boloni and M. Maheswaran *et al.*, 2001. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distrib. Comput., 61: 810-837.

Braun, T.D., J.S. Howard, A.M. Anthony and Y. Hong, 2008. Static resource allocation for heterogeneous computing environments with tasks having dependencies priorities deadlines and multiple versions. J. Parallel Distrib. Comput., 68: 1504-1516.

Chen, H., P. Chou, S. Duri, H. Lei and J. Reason, 2009. The design and implementation of a smart building control system. Proceedings of the 2009 IEEE International Conference on e-Business Engineering, October 21-23, 2009, Macau, pp: 255-262.

Edalat, N., C.K. Tham and W. Xiao, 2012. An auction-based strategy for distributed task allocation in wireless sensor networks. Comp. Commun., 35: 916-928.

Farhangi, H., 2010. The path of the smart grid. Power Energy Magza., 8: 18-28.

Firner, B., R.S. Moore, R. Howard, R.P. Martin and Y. Zhang, 2011. Smart buildings, sensor networks and the internet of things. Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems, November 2-4, 2011, Seattle, WA, USA., pp: 337-338.

Gupta, V., R. Goldman and P. Udupi, 2011. A network architecture for the web of things. Proceedings of the 2nd International Workshop on Web of Things, June 16, 2011, San Francisco, CA, USA., pp: 1-3.

Huang, H.P., R.C. Wang, L.J. Sun, H.Y. Wang and F. Xiao, 2009. Research on tasks schedule and data transmission of video sensor networks based on intelligent agents and intelligent algorithms. J. China Univ. Posts Telecommun., 16: 84-91.

Kim, J.K., S. Shivle, H.J. Siegel, A.A. Maciejewski and T.D. Braun *et al.*, 2007. Dynamically mapping tasks with priorities and multiple deadlines in a heterogeneous environment. J. Parallel Distrib. Comput., 67: 154-169.

Maheswaran, M., S. Ali, H.J. Seigel, D. Hensgen and R. Freund, 1999. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. J. Parallel Dist. Comput., 59: 107-131.

Momeni, H., M. Sharifi and S. Sedighian, 2009. A new approach to task allocation in wireless sensor actor networks. Proceedings of the 2009 First International Conference on Computational Intelligence, Communication Systems and Networks, July 23-25, 2009, Indore pp: 73-78.

Okhovvat, M., M. Sharifi and H. Momeni, 2011. Task allocation to actors in wireless sensor actor networks: An energy and time aware technique. Proc. Comput. Sci., 3: 484-490.

Pacific Controls, 2010. Bot gateway. Technology for Sustainable Development. http://pacificcontrols.net/products/bot-gateway.html

Zahariev, P.Z. and G.V. Hristov, 2011. Performance evaluation of data delivery approaches for wireless sensor networks. Proc. Comput. Sci., 3: 714-720.