COORDINATED TASKS: A FRAMEWORK FOR DISTRIBUTED TASKS IN MOBILE AREA NETWORKS.

M. Palomera-Pérez[1], H. Benítez-Pérez[2], and J. Ortega-Arjona[3]

[1]Posgrado en Ciencias e Ingeniería de la Computación
Universidad Nacional Autónoma de México
miguel.palomera@gmail.com

[2]Departamento en Ingeniería en Sistemas Computacionales y Automatización
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Universidad Nacional Autónoma de México
hector@uxdea4.iimas.unam.mx

[3] Departamento de Matemáticas
Facultad de Ciencias
Universidad Nacional Autónoma de México
jloa@ciencias.unam.mx

Abstract. *A mobile sensor network is a special kind of MANET (Mobile Area Network), composed of mobile nodes with sensors. Thus, nodes are capable to sense changes in their environment. Normally, such changes are expected to trigger events in the network. Generally, every time an event occurs, a distributed task should be performed by some nodes of the network. Moreover, in general, MANET nodes are specific components, which means that their hardware and the software are designed to perform a particular set of processes.*

Coordinated Tasks *are proposed here as a model for solving reconfiguration of a network, due to events. A Coordinated Task is defined as a composition of processes that execute on different nodes, and it is only feasible when these nodes are available in the network. In this paper, it is provided a full description of the Coordinated Task execution model.*

Keywords: MANET, complex networks, distributed computing.

1. **Introduction.** The composition of several processes to perform a complex task has been previously studied, in relation with the concept of coordination [1]. However, this approach does not take into consideration neither the availability of required resources, nor changes in the environment that trigger events in the network. In this, the environment is static: every time the distributed task is executed, the same conditions should be present.

This paper focuses on a mobile sensor network, as a distributed dynamic environment, where mobile nodes can enter or leave the network at any moment. A mobile node, thus, is a module integrated by a processor with certain identity, memory, and communication capabilities in the network. Further, some nodes have sensors, meaning that they are capable of sensing changes within the mobile network.

In mobile sensor networks, the interaction of processes executing on different nodes leads to creating a special kind of distributed tasks, namely *Coordinated Tasks*. These are only schedulable and feasible when all the computing resources it requires are available. Using such a coordination model, it is possible to specify what resources (nodes-processes) are available to integrate a particular Coordinated Task, as well as to carry out the inter-processes communication.

Hence, this paper proposes a protocol to perform Coordinated Tasks, particularly in mobile sensor networks, and generally in MANET networks. The design of the protocol considers the following:

- It is possible only to use the processes already defined and implemented in the available nodes.
- Mobile nodes are not always available: since they leave the network, or because they are performing another task.
- Coordinated Tasks should be completed before a deadline, and
- Coordinated Tasks are triggered by events.

Coordinated Tasks are normaly defined by an execution script, which sets up the required processes and the order of their execution. The script also includes the name of the Coordinated Task, and the specific event that triggers its execution. It is important to note that the execution script only establishes the required processes, but does not define what node performs it. Therefore, the first step to execute a given Coordinated Task is to map its processes onto nodes of the mobile network. Furthermore, as Coordinated Tasks are executed in response to events in the mobile network, they are defined to be completed before a specific deadline. Such deadline is a time value that is also included as part of the execution script.

The key features of the results obtained using the approach presented here refer to context awareness and service management, rather than process management, which is a common approach in distributed computing, as for example, Kahn networks. There are similar approaches, like for example [12].

The present paper is structured as follows: Section 2 provides an overview of the Related Work, Section 3 defines and discusses the Coordinated Task model, Section 4 describes the algorithm for executing a Coordinated Task, Section 5 presents the results of simulating the execution algorithm, and finally, Section 6 describes some conclusions and future work.

2. **Related work.**

Several technologies have been previously combined in previous related work to develop a protocol for MANET networks, allowing it to be *context-aware*. Such a context-aware feature is what allows triggering a distributed task every time an event occurs. In all previous work, the first step to execute a distributed task is to discover which nodes can perform the processes that compose such a distributed task.

These service (or process) discovery regarding a mobile network has been an important research topic during the last few years. Examples of these are the SLPManet [2] protocol, which modifies the Service Location Protocol [3] to work in MANET environments. Basically, this protocol establishes that service request packages are broadcasted, while service replies packages are cached by every node. However, due to this protocol is developed at the level of the application layer, it simply cannot access any routing information. Another problem is that cache entries may be false when the network topology changes.

In this paper, it is proposed a discovery-less architecture, similar to [4]. Nevertheless, instead of AODV, DSR [5] is used here. Moreover, the protocols proposed by some related work only one service is discovered per message. Here, in order to execute a coordinated task, several processes should be found, and hence, the protocol proposed uses multiple service requests, encapsulated into one DSR route request.

Other approaches have been developed around the study of MANET networks and mobile networks, related to the topic here, such as those presented in [12, 13, 14, 15]. These types of study have allowed us to focus the research in context awareness instead of process management.

Further, the Coordinated Task protocol attempts to keep a low network load by merging all service discovery replies that belong to a particular Coordinated Task. Here, service discovery replies are piggy-backed on DSR route reply packets.

After process discovery, the related Coordinated Task should be executed. However, executing a distributed task on MANET is challenging, since during execution,the availability of nodes may change: some mobile nodes become unavailable because they leave the network, or fail. Thus, they have to be replaced by other available nodes.

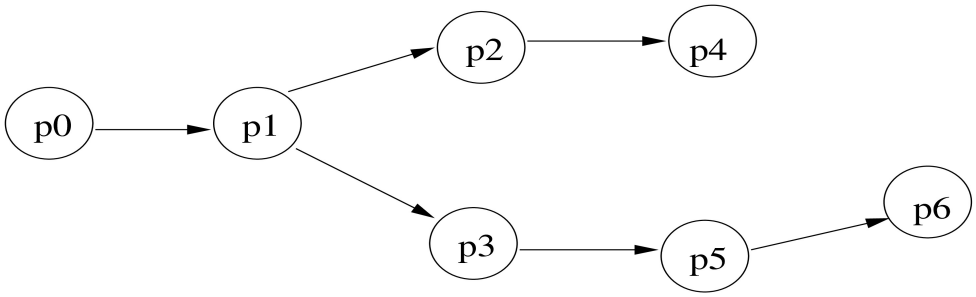


FIGURE 1. Example of execution flow for a Coordinated Task of seven-processes.

In [6], a distributed task is modelled as a Task Graph. An on-line algorithm is used to replace the unavailable nodes. However, this does not take into consideration neither the perfoemance (in terms of execution time) nor the network load, which makes this approach not suitable for the problem posed here.

Here, a similar Task Graph approach to [6] is used, but only to represent the data flow among of the processes. However, the network load is considered in order to map processes to nodes. Moreover, local cache information is used for replacing unavailable nodes, instead of performing new service requests.

3. **Coordinated Tasks.** A *Coordinated Task* is specified by using a control-driven coordination model [7]. In this model, every process of the task has two ports: input and output. These ports are the only means of interaction between a process and its environment.

Input and output ports from different processes are linked to form a *communication* flow. This communication model guarantees that the activity of each process is decoupled from the activity of any other process: each process is only in charge of performing a sequence of instructions that consumes and produces data. Data transmission between two nodes is responsibility of the operating system on those nodes.

Executing a Coordinated Task creates a data flow in the network. This flow, called CT execution flow, is modeled using a Task Graph. The CT execution flow has a root node (represented as p_0 in Figure 1). This node contains the process in charge to detect the event that triggers the execution of the Coordinated Task.

The CT execution flow is implemented by using an adjacency list. Entries of such a list are tuples, composed of two process ids: one for the source process of the flow, and the other for the destination process. Tuples in the list are ordered according to the flow direction. So, every process may appear as source or as destination. For the CT execution flow in Figure 1, the correspondent adjacency list is as follows:

$$\{(p_0, p_1), (p_1, p_2), (p_1, p_3), (p_2, p_4), (p_3, p_5), (p_5, p_6)\}$$

Finally, the Coordination Task protocol is in charge of mapping processes of the Coordinated Task onto available nodes. This protocol also links the input ports of each process to the corresponding output ports. For the actual purposes of this paper, nodes with sensors are specifically used to start a Coordinated Task, and thus, they have a CT execution script per each event. Hence, every Coordinated Task has an associated execution script.

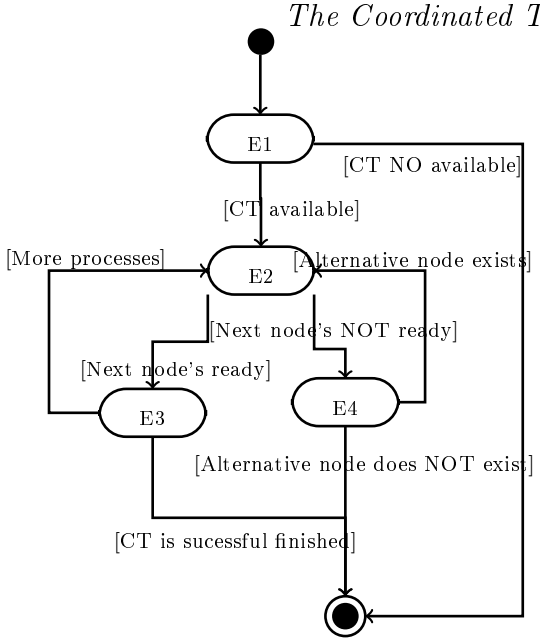
Commonly, the CT execution script includes the following information:

Coordinated Task Name: a unique name that identifies a Coordinated Task.

Trigger Event Identifier: a unique id that represents an event.

Processes Identifiers: a field used to identify a process in a Coordinated Task. Depending on the application, the process id could be represented as simple as integers, or as complex as full XML descriptions, like WSDL [8]

CT execution algorithm. *Given an event, execute ist associated Coordinated Task. The Coordinated Task should met its deadline.*



E1**:** *[Discover] Execute the processes discovery algorithm.*
E2**:** *[Send] Send a message to the next node in the CT execution flow.*
E3**:** *[Execute] Execute the related process.*
E4**:** *[Search]* Search for alternative nodes.

FIGURE 2. Algorithm used to execute a Coordinated Task.

Processes Adjacency List: the adjacency list, describing the CT execution flow.
Coordinated Task Soft Deadline: every Coordinated Task should be completed before a deadline, and thus, this field represents such a deadline as a timeout value.
Coordinated Task Maximum Communication Requirement: a list indicating how many data is transmitted by every process for each node. If a node does not produce data, its entry is the size of a CT Message.

4. **Coordinated Task Execution Algorithm.** Figure 2 shows the algorithm used to execute a Coordinated Tasks. Step E1 uses a reactive process discovery algorithm [9], based on encapsulating a discovery message into the DSR routing protocol [5]. Responses to the discovery messages include the process worst execution time and the amount of data produced. These values are used in Equation 1 to obtain the coordinated task execution time E_{CT}.

$$E_{CT} = T_D + T_T + T_E \tag{1}$$

T_D is the time used by the process discovery procedure, and it is obtained by the node which detects the event. This node is also in charge of starting the process discovery algorithm.

On the other hand, T_T is the time needed to transmitted the data. To determine T_T , it is required to find out the actual network capacity. However, how to do this is still an open issue. For this paper, an approach similar to that in [10] is followed. So, using the results in [10], the transmission time (T_T) is calculated by using Equation 2. Notice that, Equation 2 considers that no message is sent in parallel.

$$T_T = \sum_{i=1}^{k} \left(\frac{\sqrt{n}}{W} L_i \right) \tag{2}$$

where:

$i..k$	are the processes in the coordinated task.
n	is the number of nodes available in the network.
W	is the nominal network bandwidth, in kbytes/second.

TABLE 1. Simulation parameters

Parameter	Value	Parameter	Value
Mac Protocol	IEEE802_11	Maximum speed	1.999 m/s
Cover radio	50 m	Number of processes	2
Area size	300 x 300 m	Event frequency	10 seconds
Simulation time	1000s	UDP message size	500 bytes
Number of nodes	50	CT deadline	1000s
Minimum speed	1.0 m/s	Multi-hop size	3

L_i is the message length in kbytes. This value is obtained from the response to the discovery message.

Finally, T_E is the time used by the nodes to execute its assigned processes. It is calculated by simply adding all the worst execution time of each process. These worst execution values are transmitted as a field of the response to the discovery message. Each nodes obtains these values as presented in [11].

A Coordinated Task is considered feasible if its execution time (E_{CT}) satisfies Equation (3), where D_{CT} represents the CT deadline.

$$E_{CT} \leqq D_{CT} \tag{3}$$

The following step in the execution flow is started by sending a message to the node (step E2). If there is no MAC ACK message from this node, it is considered that the node is not available, and a search for an alternative node is started (step E4). On the contrary, if the node is available, it executes its processes (step E3). This procedure repeats until the Coordinated Task finishes, or until it is stopped due to the unavailability of any of the nodes required to perform the processes of this particular Coordinated Task.

5. **Preliminary results.** A ns-2 simulator is used here to test the performance of the Coordinated Task protocol, according to the following features:

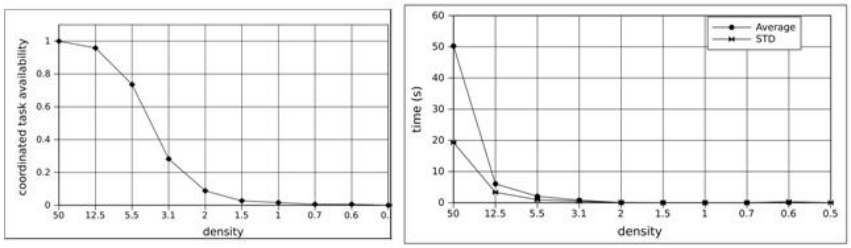
Availability: is the ratio of the number of successful Coordinated Task executions over the number of events during a simulation. A Coordinated Task execution (CT) is considered successful if it satisfies that: *(a)* all its processes execute, *(b)* all the nodes receive an UDP message, giving the CT data flow in time. Notice that the processes execution time is not taken into account in this simulation, due to the goal of the experiments is to model the behavior of the mobile network.

Execution Time: is the time in seconds between the request of CT execution and when the last nodeof the CT data flow gets its message.

Table 1 shows the parameters used during the experiments. Notice that each simulation is performed 500 times.

At the beginning of every simulation, nodes are set at random positions inside a two-dimensional area. During all the simulation time, nodes are kept moving though random paths inside such an area. However, the speed of each node is kept constant during all the simulation. Such an speed is randomly selected for each node at the start of the simulation, from a speed range (Table 1).

During the experiment, availability and execution times are measured depending on the density of nodes. Such a density is modified regarding the size of the area, which changes from 100×100 to 1000×1000 in steps of 100 units. This approach is preferred instead of directly increasing the number of the nodes, which could compromise the performance of the simulator. Figure 3 shows the density of nodes, given in units of nodes per area of 100×100 units.



a) Coordinated Task availability **b) Coordinated Task execution time**

FIGURE 3. Simulation results. The density of nodes is given as number of nodes per area of 100×100 units.

The availabilty of the Coordinated Task depending on the density of nodes is presented in Figure 3a. From this, it is noticeable that the density of nodes has a higher impact on the number of successfully performed Coordinated Tasks. For a larger density of nodes, there is a larger number of successful Coordinated Tasks. However, increasing the density of nodes has a negative impact on the execution time (Figure 3b): execution time tends to increase with the density of nodes. Moreover, its STD also increases, confirming that the network latency tends to be worse for larger density of nodes.

6. **Conclusions and Future Work.** This paper presents the Coordinated Task, as a protocol to perform a distributed task on MANET networks. The distributed task is executed as response to events in the network. An event is any variable sensed by any node of the network. Particularly, the protocol discussed here is suitable when a temporal new behavior is required due to a specific event in environment.

In order to measure the performance of the Coordinated Task protocol, it is simulated using the ns-2 simulator. The simulation results show that the density of nodes has a higher impact on the execution time. For a larger density of nodes, there is longer execution time. Moreover, the latency of the execution time also increases with the density of nodes. According to this, it is suggested that the effects of the density of nodes requires further study. How many nodes should execute the same process, and how many nodes are required in total to guarantee the execution, are left as future work.

Another issue regarding the use of this protocol is how to deal with missing nodes, due to the nodes leaving the network. For this, the protocol proposes looking at the local cache for alternative nodes. However, the results show a low rate of successfully executed Coordinated Tasks, while the number of processes increases. Therefore, a better mechanism to find alternative nodes is needed. An alternative to explore is to allow the nodes to search in the cache of their neighbors.

Acknowledgements. This is a work in progress as part of proyects PAPIIT-UNAM IN103310 and ICyTDF PICCO 10-53.

REFERENCES

[1] Theophilos A. Limniotes and George A. Papadopoulos. Real-Time Coordination in Distributed Multimedia Systems. *IPDPS '00: Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing. 2000*

[2] El Saoud Mohamed Abou , Kunz Thomas and Mahmoud, Samy. SLPManet: service location protocol for MANET. *IWCMC '06: Proceedings of the 2006 international conference on Wireless communications and mobile computing 2006.*

[3] Guttman, E. Service location protocol: automatic discovery of IP network services. *Internet Computing, IEEE 1999*

[4] PaaL E. Engelstadl ; Yan Zheng ; Rajeev Koodkli and Charles E. Perkins. Service Discovery Architectures for On-Demand Ad Hoc Networks. *Ad Hoc & Sensor Wireless Networks 2006.*

[5] D. Johnson, Y. Hu, and D. Maltz. The dynamic source routing protocol (dsr) for mobile and hoc networks for ivv4. *Thechnical report, Internet-Draft. 2003*

[6] Prithwish Basu, Wang Kem, Thomas D.C. Little. A Novel Approach for Execution of Distributed Tasks on Mobile Ad Hoc Networks. *IEEE 2002.*

[7] George A. Papadopoulos and Farhad Arbab. Coordination models and languages. *Centrum voor Wiskunde en Informatica (CWI) 1998.*

[8] Christensen Erik , Curbera Francisco , Meredith Greg and Weerawarana Sanjiva. Web Service Definition Language (WSDL). *http://www.w3.org/TR/wsdl*

[9] P.E. Engelstad and Y. Zheng. Evaluation of service discovery architectures for mobile ad hoc networks. *Proceedings of the Second Annual Conference on Wireless On-demand Network Systems and Services, pages 2-15. 2005*

[10] P. Gupta and P. R. Kumar. March 2000). The capacity of wireless nwtworks. *IEE Tansactions on Information Theory, 46(2): 388-404.*

[11] Palomera-Pérez Miguel and Benítez-Pérez Héctor. Scheduling Coordinated Tasks. *In proceedings of the 5th IEEE Conference on Industrial Electronics and Applications. X'ian, China. 2009*

[12] Nakano, H., Utani, A., Miyauchi, A., and Yamamoto, H., Grouping of Mobile Nodes in MANET based on Location and Mobility Information using an ART Network. *International Journal of Innovative Computing, Information and Control. Volume 5, Number 11(B), pp. 4357-4365, 2009*

[13] Chiang M. and Wang, S., Hybrid Consensus Agreement on CDS-based Mobile Ad-Hoc Network. *International Journal of Innovative Computing, Information and Control. Volume 5, Number 8, pp. 2291-2309, 2009*

[14] Wang, S., Wang, S. S., and Yan, K.Q., The New Territory of Generalized Byzantine Agreement in a Virtual Subnet of Mobile Ad-Hoc Network. *International Journal of Innovative Computing, Information and Control. Volume 4, Number 8, pp. 2097-2112, 2008*

[15] Lai, W. K., Weng, M., Lo, S., and Shieh, C. KAdHoc: A DHT Substrate for MANET Based on the XOR Metric. *ICIC Express Letters. Volume 3, Issue 4(A), pp. 909-914, 2009*