Services Collaboration in Wireless Sensor and Actuator Networks: Orchestration versus Choreography

Sylvain Cherrier*, Yacine M. Ghamri-Doudane†, Stéphane Lohier*, and Gilles Roussel*
* *Université Paris-Est , Laboratoire de l'Institut Gaspard Monge (LIGM)*
77454 Marne-la-Vallée Cedex 2
† *Ecole Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise (ENSIIE)*
1 square de la résistance,91025 Evry Cedex
Email: [firstname.lastname]@univ-paris-est.fr

***Abstract*—Wireless Sensor and Actuator Networks (WSAN) and permanent connections to the Internet converge to be an emerging and promising field: Machine-To-Machine (M2M) services. To take advantages of this new field, hardware and software infrastructure compliance must be verified. Services expected by M2M alter the organization of WSAN. The software design in this area can be divided into two main categories: a centralized approach (Orchestration) where a monolithic application collects data and sends orders, and a distributed approach (Choreography) in which nodes offer and use services in a collaborative way. In this paper, we study the impact of these two architectures over WSAN. First, a mathematical analysis shows the improvement offered by choreography, thanks to the use of shorter paths between nodes. Then, an application experiments these two architectural designs to measure the impact on a real testbed. Both the theoretical mathematical analysis and the real platform experiment gives better results for the Choreography in terms of network reliability and path length. Our work quantifies the benefits obtained and provides histograms and numerical results.**

***Keywords*-Machine-To-Machine; Choreography; Orchestration; Wireless Sensor and Actuators Network; Services Oriented Architecture**

I. Introduction

The world of sensors, actuators and "smart objects" is subject to strong constraints: very limited energy, low throughput rates, restricted processing power and memory space, etc. However the small contribution provided by each node is useful, and can help dealing with user's needs. Part of this ability to process information is primarily used to organize the network. At first glance, applications using this type of "smart objects" are organized around a central node called the sink. All information is uploaded to the sink, which is the link between the inside network and the world. The sink is unique: it often has boundless energy and a much better processing power.

When merging into the wider world of M2M universal exchanges and interactions between objects, the use of WSN changes. From M2M perspective, WSAN is no more a redundant array of multiple similar sensors responsible for carrying out many measures of the same physical quantity. WSAN is rather a highly varied collection of small specialized objects, very different and complementary, responsible for coordinating actions or measures. From that point of view, the whole behaviour is modified. Data flows change from a "many-to-one" kind to a "one-to-one" node's communication, directly between sensors and actuators. Data types, message frequencies and uses become more heterogeneous. The sink role is reduced to a simple gateway to the wide Internet. Giving each component universal access from and to the Internet was the first step. The next step is the definition of new types of software designs that take into account WSAN constraints (i.e. mainly optimizing the network's lifetime) and M2M needs.

The first architectural approach inherited from WSN (*Orchestration*) is centralized: a unique application, responsible for offering access and control to users, is located somewhere outside the WSAN, behind the sink. It gathers all the data collected. After processing them, the central application deduces actions to perform. Orders are then transmitted to the actuators present in the network. This centralized approach involves a specific traffic pattern in the network. The paths from nodes to sink (and sink to each node) are massively used, leading to congestion and high energy consumption.

The other approach advocates a distributed application logic instead of the centralized one, when application characteristics allow its implementation. This approach, also known as *Choreography*, tries to process data and to give decision-making to the nodes themselves. Each node holds a part of the application, and executes some process over the data. Small interacting parts of the application are distributed over the network. Computing is performed inside the network, and communications can often be better disseminated over the network. When it is possible, decisions are made at node's level, no need to send information to any central application.

In this paper, we perform a thorough study of the impact on the network performance of these two different software designs for M2M over such a WSAN structure. Services Choreography and Orchestration solutions are compared

in terms of network traffic loads and communication path length to determine the impacts of each approach regarding a more efficient and responsive design.
The reminder is organized as follows: first, we present the related works and the background (Section II) attached to our approach. Our mathematical analysis is described in Section III, while Section IV focuses on testbed experiments and results. Finally, concluding remarks and future research directions are given.

II. Background and related work

WSAN and M2M can be classified into multiple classes. One of them is "data-centric", which means that several sensors of the same type collect data from their environment. This is for example used in building or environment monitoring. In that case, users are mainly interested in values. Data are sent to the sink, or the sink looks for the value of a physical measurement [11]. To save energy, a cluster oriented solution can be adapted, because physically adjacent sensors detect approximately the same value, the same event, at the same time. Here, compression algorithm and data aggregation are the effective solution.
But building M2M by the integration of WSAN in a superset leads to a different view of its use. M2M is made up of interactions between an eclectic collection of multiple objects to give the user a digital vision and use of his environment. Previous views of WSN presented in [8] such as "*WSN performing a specific task*", or "*sending messages not to individual nodes but to geographical location or regions*" may not apply to WSAN when put inside the M2M realm. Multiple components with highly diversified characteristics (a mix of heterogeneous sensors and actuators) are working together. Tilting from a "sensing" to an "interacting" vision, WSAN (as part of M2M) is more "event-centric" than "data-centric".
This direct interaction between sensors and actuators is also called "coordination" [2]. In this paper, the authors described the direct "sensor-actuator" communication as "Automated Architecture", while the usual mode is called "semi-Automated Architecture" because it needs a central controller, on the sink or beyond. Owing to industrial improvements, processing capabilities of smart objects increase, and the implementation of full Internet protocols suite becomes possible [6]. Therefore, at application level, considering the whole WSAN as a Service Oriented Architecture (SOA) makes sense [10]. Hence, in a services approach, the "semi-Automated Architecture" can be seen as a *Services Orchestration*, and "Automated Architecture" is implemented by a *Services Choreography*.
The main difficulty for a real distributed application lies in the heterogeneity, especially in WSAN where hardware, Operating Systems and languages are various. Such diversity is opposed to a distributed organization of the software [3], because "*the source code of a node is tangled and tightly coupled*", "*Middleware interface and its composing components are not precisely identified*", and "*WSN codes development is done in an ad-hoc manner*".

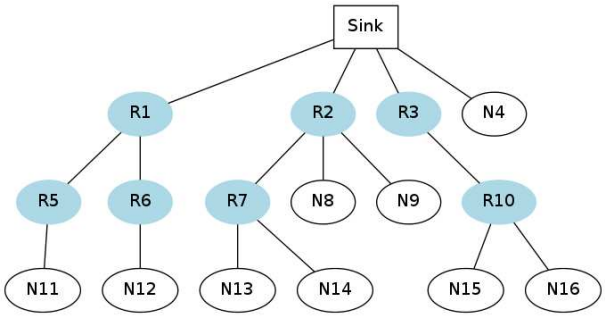

Figure 1. A network represented as a tree. For example, path length between node N9 and N13 is:
1) in a Choreography : 3 hops (R2-R7-N13)
2) in an Orchestration : 5 hops (R2-Sink-R2-R7-N13)

This orientation towards services collaboration rather than basic hardware coding releases the coupling between devices (such as in "Message-Oriented Middleware" presented in [7]). It also offers an easy way (when feasible) to design collaborative applications between nodes [4]. But this new organization forces us to think about new perspectives and issues in managing this highly constrained network. "*Many protocols and algorithms have been proposed for WSNs [but] they may not be well-suited for the unique features and application requirements of WSANs*" [2]. Furthermore, in "semi-Automated Architecture" (the traditional WSN application design, that we call in our services point of view an *Orchestration*), communication latency are not minimized and "*transmitting the sensing data to the sink usually causes fast energy depletion of nodes which are around the sink*" [9]. We are interested here in quantifying the effects of these two architectural designs.

III. Problem Statement

A. Choreography and Orchestration

In the case of hierarchical networks of organizations, most protocols build solutions in tree form. Considering our network as a tree (see Figure 1), it is possible to evaluate the impact of application's design on this architecture. The way each node sees and uses the WSAN depends on that design. For example, consider an application build following an *Orchestrated* design. If node 9 is a sensor, and node 13 an actuator, the user may want node 13 to react to events detected by node 9. In that case, node 9 has a specific vision of the network (see Figure 2). Each message goes to node 2, then to the sink. After its trip through the outside network in direction of the central application, a second message containing the decided action comes back, relayed by the sink, going through node 2 to node 7, and then to node 13, its destination. The path from node 9 to node 13 is five

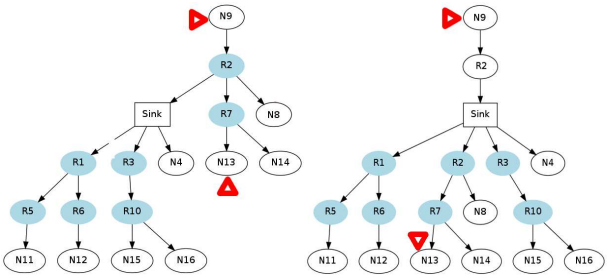

Figure 2. From application perspective, the node is seen as the root of a new tree. Here, the path from node 9 to node 13, as seen by node 9 in a Choreography (left), or in an Orchestration (right)

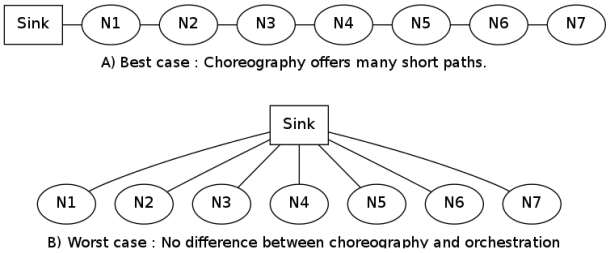

Figure 3. Extreme cases in Orchestration/Choreography comparison

hops long. Note that node 2 is solicited twice: a first time on the way up to the sink, and a second time when the action message comes back. All the network's components behaviour is driven by the central application. All data must go to the sink, to be forwarded to the application, and then an action comes back in the network.

In a *Choreography*, the global application is designed as a direct collaboration between nodes. In that architecture, the limited processing capacity of each node may be used to compute data closer to where it has been sensed, instead of being sent to a central application. Processing data consumes less energy than transmitting it (12.7 times less on a TelosB, 2.5 times less on a MicaZ [1]). Avoiding a hop and its transmission is always a good idea, especially if it is to relieve a node that is already overloaded (R2 in our example). In *Choreography*, a node sends information directly to an (or some) other node(s), and not to a central decision point. The logic of the application is distributed, and spread over the network. The decision about the action to take is made directly on nodes.

Depending on the application's design, node's view of the global network is different. In a *Choreography*, as shown in Figure 2, the path from node 13 to node 9 is made of 3 hops. The message is delivered more quickly, with less risk of getting lost. Node 2 has only one message to transmit. Collision risks decrease. In *Orchestration*, the same needs of communication between nodes involve higher number of hops (Figure 2).

B. The vision from application layer

WSAN are characterized by their strong constraints (energy, memory, throughput), but also by the advantage of having a processing power, however small it may be. The way these objects are used and interact is also a new domain. It's interesting to take advantage of the local processing power to relieve the entire network, because it results in less energy consumption and lower radio traffic.

In our study, we adopt the perspective of the programmer, and our approach is based on services. We do not evaluate here the appropriateness of the proposed network organization, but rather the consequences of the two application's design approach. We consider our network as a tree of nodes to represent the standard organization inside a WSAN. Although there may be nuances in each solutions proposed, this simplification has no impact on experiment's validity, because *Choreography* is more likely to get benefits of improvements provided by various network routing protocols. It is clear that *Choreography* uses some shorter ways in many cases (in facts, to all nodes that are on the path to the sink, or connected to that path, see Figure 2). All the other nodes are at the same distance in the two organizations. Our study focuses on qualifying and quantifying the consequences of that design's choice in terms of path length, network reliability and possible deductions we can make about network lifetime.

[1]See data sheet on MemSic WebSite (http://www.memsic.com/)

IV. Mathematical approach of path length calculation

A. Best and worst cases

Our mathematical approach is mainly based on the calculation of the path length when two nodes try to communicate. Each path in the tree presented in Figure 1 can be considered in two ways, *Choreography* or *Orchestration*, changing the vision from each node (Figure 2). This characterization depends on the software design running in the node.

A mathematical analysis gives *the best* and *the worst* case shown in Figure 3. All nodes are numbered from the sink (0) to the last node (n). Path length is the distance between a node i and a node j. In the first place, let's study the linear model of Figure 3(A), where node 7 is only accessible through node 6, which is accessible from node 5, etc. By adding the length of all possible paths (i.e. for all i and j), and dividing by the number of all possible couples (i, j), we get the average path length $\mu(n)$ for n nodes (we divide by two not to count the path from i to j plus the path from j to i).

$$\mu(n) = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} distance_{i,j}$$

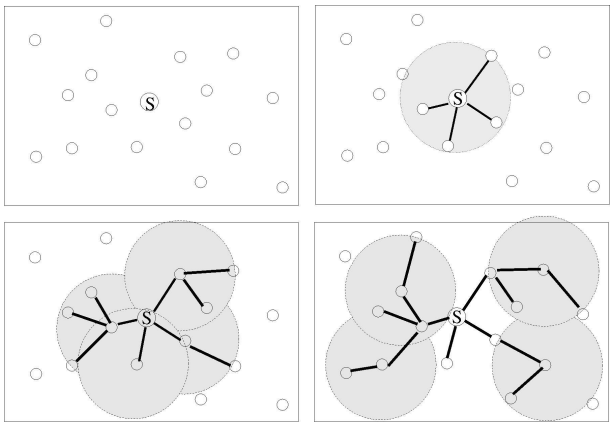

Figure 4. Algorithm used to build the tree. The sink (S) searches its neighbours, which in turn look for their neighbours, and so on.

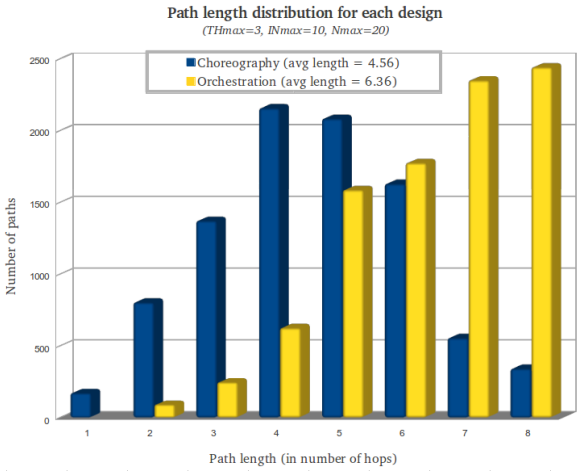

Figure 5. Comparison between Choregraphy and Orchestration for a reduced height and very wide tree (depth 3).

This formula can be simplified in

$$\mu_(n) = \frac{2}{|P|} \sum_{P} distance_{i,j}$$

$$\text{with } P = \text{ set of possible couples of } n \text{ nodes}$$

In the case of *Orchestration*, the path goes from node *i* to the sink (node 0), and then back from node 0 to *j*. The distance between two nodes *i* and *j* is equal to $(i + j)$.

$$\mu_o(n) = \frac{2}{|P|} \sum_{P} (i + j) = n + 1 \quad (1)$$

Considering *Choreography*, the distance between two nodes *i* and *j* is the shortest path in the tree, and equals $(j - i)$.

$$\mu_c(n) = \frac{2}{|P|} \sum_{P} (j - i) = \frac{1}{3}(n + 1) \quad (2)$$

After simplification, the average path length between two nodes in a linear structure of *n* nodes is $n+1$ in the case of *Orchestration*. For a *Choreography*, the average path length is $1/3(n+1)$. In the best case, a *Choreographed* architecture reduces the average path length by 3.

Figure 3 (B) shows a one hop tree. In that case, the path length between two nodes is always 2 hops. There is no difference between *Choreography* and *Orchestration*. This is the worst case, in which *Choreography* is strictly equal to *Orchestration*, and brings no improvement.

So, the best case is encountered when the tree is very thin, and each node has no sibling. On the contrary, a one-hop tree is the worst case, giving a strict equality between *Choreography* and *Orchestration*. However, *Choreography* always offers shorter average path length, except in the worst case scenario where both designs are equivalent.

B. Our probabilistic model for the general case

To study the general case, we have randomly positioned nodes on a square (Figure 4) with the sink (the root) in the centre. The radio range is set through a radius parameter. We use Unit Disk Graph (UDG) to simulate the accessibility of each node. Any node at a distance less than this value is considered accessible, otherwise inaccessible. The sink starts to find reachable nodes. These nodes try to reach other nodes, and so on (Figure 4). The resulted tree is analysed when used by *Orchestration* and by *Choreography*. The simulation gives the distribution of numbers of paths by length for each design.

Figures 5, 6 and 7 show our results when comparing these two designs, depending on nodes density, number of nodes, and the three following parameters : Tree Maximum Height (*THmax*), Internal Nodes Maximum number (*INmax*) and Nodes Maximum number (*Nmax*). These parameters are used to build different types of trees. They correspond for example to the parameters *Lmax, Rmax and Cmax* used in ZigBee [1].

All simulations are based on an area containing 100 nodes. For each graph, we vary the three parameters (*THmax, INmax and Nmax*) to build the structure of the tree. Then, we count the length of each path from each node to all other nodes. This is done according to *Orchestrated* architecture (via the sink) and then to *Choreographed* architecture (using the shortest path in the tree). Each analysis is the averaged distribution of a set of 1000 tests. The average of 10 analyses is plotted on Figures 5, 6 and 7.

C. Results given by the probabilistic model

Our first result (Figure 5) represents the distribution of paths length for a reduced height and wide tree (a 3 maximum height tree, with 20 children maximum by node, of which 6 are internal nodes maximum). Even with these values that do not favour the *Choreography*, path length is generally and significantly shorter than in *Orchestration*. It results in a shorter response time, a reduction of total energy

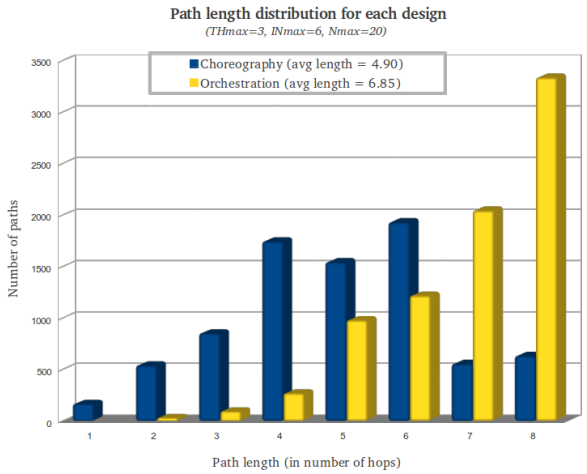

Figure 6. Comparison between Choreography and Orchestration for usual values given in ZigBee presentation.

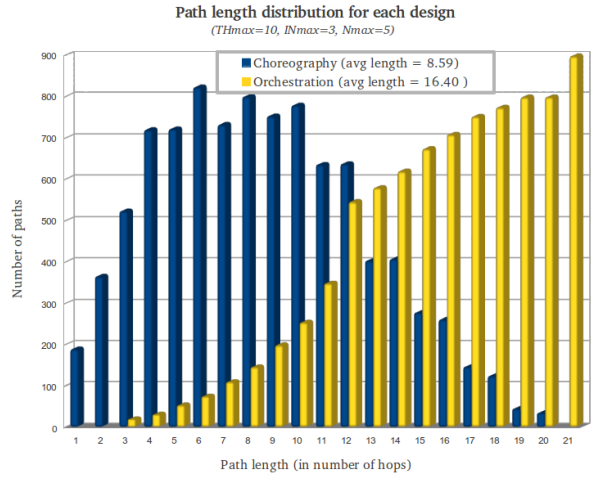

Figure 7. A favorable case for Choreography.

consumption, and a smaller risk of information loss for a non-reliable network such as WSAN.

Our second plot (Figure 6) is the path length distribution calculated using usual parameters values than can be found in the literature [1] (*THmax=3, INmax=6, Nmax=20*). The resulting tree is less compacted than the first one, because each level has fewer internal nodes. Our calculations show a significant increase in the number of paths of maximum length in the *Orchestration* case.

The last graph (Figure 7) shows the differences when the network configuration parameters are favourable to *Choreography*: important tree height, few internal nodes and leaves at each level. Once more, the *Choreography* is a more efficient organization.

Finally, the 10 analyses we conducted allowed us to obtain the average path length standard deviation of our various simulations. In facts, these standard deviation values are so low that we could not even plot them on the graphs. They mostly represent less than 1% of the number of paths, showing that there is very little variation between different estimates.

Table I
THEORETICAL AVERAGE PATH LENGTHS ACCORDING TO THE APPLICATION DESIGN

A n nodes network	Average path length		
	Orch.	Chor.	Ratio C./O.
Worst Case (THmax=0)	2	2	100%
THmax=3 INmax=3 Nmax=10	6.96	5.67	81 %
THmax=3 INmax=6 Nmax=20	6.85	4.90	72 %
THmax=3 INmax=10 Nmax=20	6.36	4.53	71 %
THmax=10 INmax=3 Nmax=5	16.40	8.59	52 %
Best case (THmax=n)	(n+1)	1/3(n+1)	33 %

Finally, Table I gives the compatibility of our results while comparing the two models: the “best and worst cases" vs “general case". Results given by the general case are positioned between the extreme cases. As the tree height increases, the path length gains from the *Choreography* are becoming more pronounced. They are accentuated by a small number of internal nodes, reaching a reduction by nearly 2 in the case presented on Figure 7.

To sum up with the mathematical analysis, we have shown that for any kind of tree, the *Choreography* is always a better choice. Compared to an *Orchestrated* architecture, the gain in terms of path length vary from 1 to 3 times better. The improvement depends on the characteristics of the given network topology and the nodes involved in the communication.

V. TESTBED EXPERIMENTS AND RESULTS

A. TestBed description

To experiment these two architectures with a real implementation, we developed specific software on Contiki [5]. Contiki is a Free Software operating system for different sensors hardware, such as TelosB, MicaZ[2] or Sensinode[3] devices. Contiki comes with Cooja, a network simulator connecting emulating TelosB devices running Contiki. The choice of Cooja instead of other network simulators, such as NS-2 or NS-3, is motivated by the fact that we wanted to see the real effects of an architectural design over a running implementation. Launching our tests over a testbed gives a better view for evaluating the real consequences of a design.

Contiki uses an implementation of 6LowPan, and a routing protocol called RPL, in charge of building the routing paths (i.e. the tree). The programmer does not provide any parameter for that tree, and has no control over it (see Figure 8 and 9). In addition, the tree may be rebuilt dynamically if needed. The retrieved data from Contiki are: number of *sent*, *received* and *forwarded* IP packets. These data show the

[2]From Memsic http://www.memsic.com/
[3]From Sensinode http://www.sensinode.com/

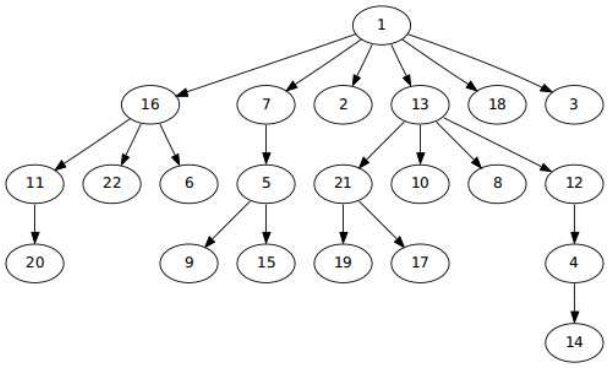

Figure 8. Tree given by Contiki during Orchestration experiment #50. High level routers are 16, 7 and 13.

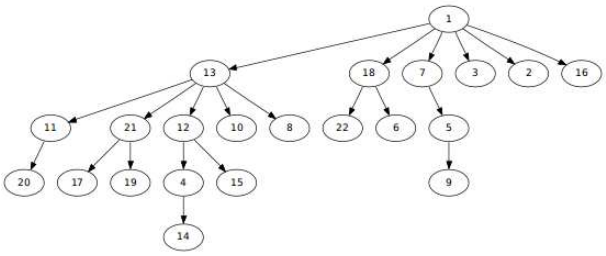

Figure 9. Orchestration experiment #60: node 16 is no more a router. Node 18 is now in charge of node 22 and 6. 13 becomes more important.

entire network activity (including both our application and network inner requirements). Our application enables us to obtain the number of messages that have been successfully received.

In this paper, the word *packets* is used to mean the *whole network activity* whereas *messages* refers to *application* layer.

B. Description of the experiment

Our mathematical analysis attempted to exhaustively enumerate all possible paths in the tree according to the two architectures in order to deduce profiles of path lengths. Theses analyses examines the impact when we vary the tree shape.

But on a testbed, the programmer has no control over the building of the tree. In addition, replicating the exhaustive communication from node to node of our mathematical study is not representative of a real M2M application. To test the impact of the application design on any wireless network, we performed the following program: each node chooses only one recipient to simulate the interaction between a sensor and an actuator. Once the actuator chosen, each node sends 30 messages. Our test consists in counting the number of messages actually arrived on the destination (*an application layer view*), and the number of IP packets that were sent or routed by each router at each level of the tree *(a network view*). Results are achieved using both architectures: either by sending directly to the recipient (*Choreography*) or through the sink (*Orchestration*). The operating mode of this test provides a view of the benefits of the *Choreography* without having particularly promoted it. Indeed, the random choice of the actuator and the dynamic changes of the tree in 6LoWPAN (see Figure 8 and 9) give an overview of the design effects.

We developed 3 softwares in Contiky 2.5:

- For each node
  - A *Choreography* client/server: While listening to others, this program sends data directly to only one node.
  - An *Orchestration* client/server: Only listening and sending to the sink. Messages contain the id of the real destination's node.
- On the sink
  - A normal sink: we use code given in Contiki's rpl-udp example (being the tree root).
  - An *Orchestrated* sink: this sink analyses each received message, and relays it to its final destination. It plays two roles: the sink, and the central application.

We use 21 nodes plus a sink. According to our preliminary tests, this seems to be enough to have significant results. Nodes are positioned randomly. Experiments are run 100 times for each architecture. The only differences is the way of transmitting messages: through the sink for *Orchestration*, and using the shortest path for *Choreography*.

C. Testbed's first result : nodes activity by level

We made numerous tests to eliminate insignificant results. The design of our test is to regularly change the randomized couple sensor-actuator (in order to respect the diversity of realities, there is no evidence that a sensor and an actuator are neighbours in the tree). We eliminated the first results of the list because they were likely to be distorted by the construction of the tree. We kept about thirty results in the middle of the experiment. To identify the major trends, we grouped these measures into three classes corresponding to loading rates of top level routers (i.e. nodes 16, 7 and 3 in Figure 8). Classes are: *low*, *average* and *high* top level activity. On the graph, we also give the overall reliability of the network observed for each class.

Our first graph (Figure 10) shows the *Orchestration* results. Each bar represents the average number of IP packets forwarded by all nodes at a given level. This chart shows activity at each level, and how data traffic is mainly concentrated on high level nodes. In comparison, the deeper in the tree a node is, the less it is requested. From level 3, traffic becomes insignificant.

This graph gives another interesting value: the percentage of application messages that reached their destination. The reliability of the application increases when network activity

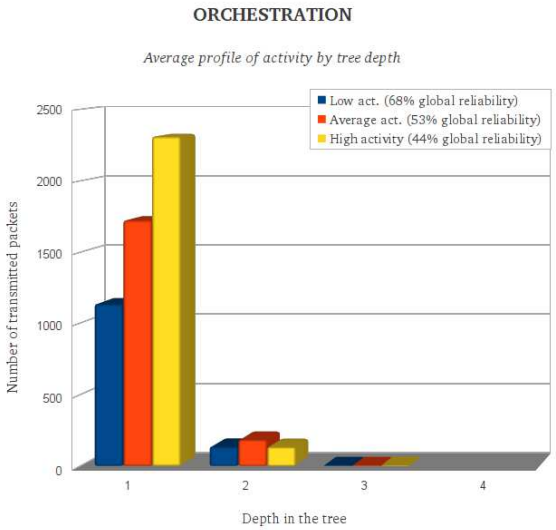

Figure 10. In an Orchestration, top level routers are heavily used because all data must travel to the sink, and come back to actuators.

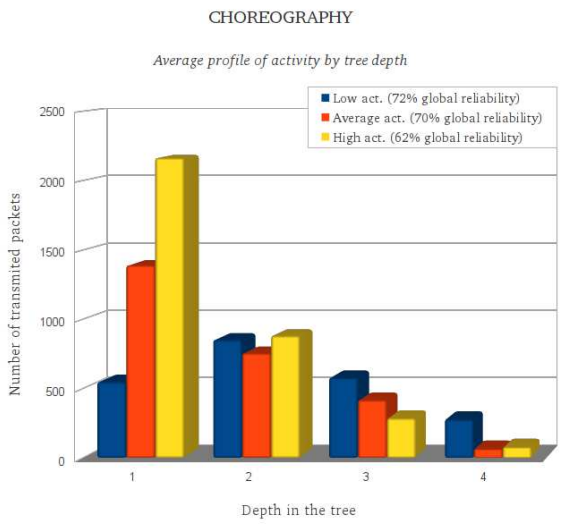

Figure 11. In a Choreography, there is non negligible chance that a message reaches its destination without crossing all tree levels.

decreases on top level nodes. Because all application messages cross the network towards the sink in the *Orchestration* design, the whole network reliability is highly correlated with the high load of these nodes. The more top level nodes are loaded, the less reliable the network is. Furthermore, even if the saturation level is not reached, we find that the design of *Orchestration* implies high activity mainly on these top level elements, consuming quickly their energy. There is little network activity at level 2, and almost not deeper.
A *Choreography* shows a different profile (see Figure 11). With the same 3 classes of average network activity at top level nodes, the forwarding effort is spread over different tree levels. Low level nodes are more used. Figure 11 shows that levels 2, 3 and 4 handle more packets, while traffic on top level decreases. So we have a better activity distribution among the network, and reliability is obviously improved compared to the *Orchestration*.

D. Testbed's global view: activity and network reliability

Reading raw results (before average) shows significant differences on the percentage of application messages reaching their destination. We focus on any possible link between the first level transmissions and the overall reliability of the network. The way of designing applications has a strong impact on network reliability and activity distribution.
Figures 12 and 13 plot the percentage of packets that reached their destination compared to the activity of the top level nodes. The network is stressed with important data rates, leading to significant packet losses. In the case of *Orchestration*, top level nodes activity is strongly linked to network reliability, and their overload leads to a degradation of the entire network accessibility. A global trend can be

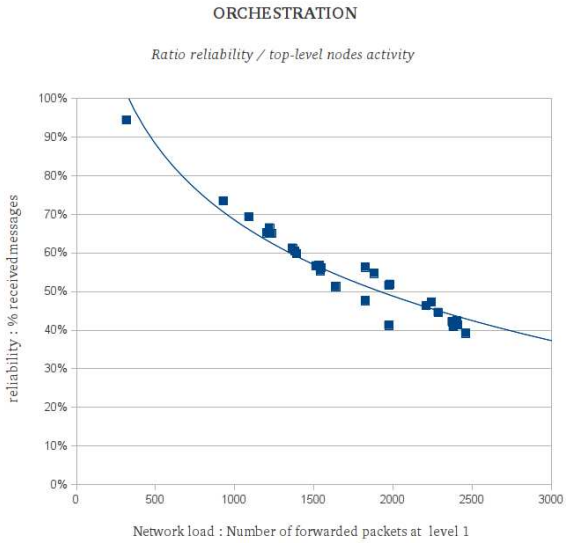

Figure 12. The more first level is overload, the less reliable is the network. Orchestration saturates top level nodes.

deduced from Figure 12. The little variations are due to few tree modification during the experiment (Figures 8 and 9). On the contrary, in the *Choreography*, the correlation does not appear so clearly (Figure 13). Traffic is better distributed within the network. The top level nodes are not involved in all communications. Network reliability reaches higher values, and is less dependent on top level nodes activity.
These two graphs show clearly how application's design impacts the network. For the same purpose, choosing to collect information and to take decision outside the network

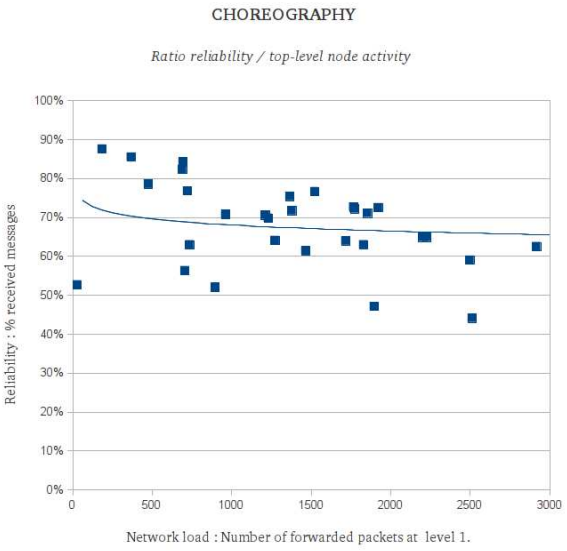

Figure 13. In a Choreography, there is no correlation between reliability and first level activity. The network works better, more depending on destination of the messages rather than their number.

(*Orchestration*) mainly affects top level nodes, and results in a higher energy consumption that eventually forces the choice of another path (by rebuilding the tree), quickly running all combinations out of energy, causing a total halt of the network. In contrast, obviously designing the same application as a Choreography spreads the network activity most randomly. Various nodes are involved, the network is more reliable and has longer lifetime.

VI. Conclusion

By integrating M2M paradigm, the characteristics of WSAN have changed: more different and various type of nodes, heterogeneous flows over the network, variable contents and sending frequencies. All these aspects lead to an organizational transformation from “many-to-one” to “one-to-one’, and finally different expectations from users. Exchanges in M2M are more diffuse. Network transmissions frequency is irregular and subject to bursts.

In this paper, we first use probabilistic models to show how the design of an application has an impact on the average length of paths. Our results show that the choreography (when it can be implemented) may propose paths of length up to 3 times shorter than an orchestrated design of the same application. Different distributions of the path length are presented. Then, the information collected during our testbed experiment shows measures of the real impact. When application’s design is orchestrated, it leads to an overload of top level nodes that is hardly managed by the network. By relieving them of a non negligible part of the traffic, choreography improves network quality and reduces power consumption at top level nodes, whose role in the overall accessibility is crucial. And the choreography never gets worse results than the orchestration in terms of path length. M2M applications over WSAN benefit from the use of choreography, as distributed software optimizes the longevity and reliability of the whole network.

In the future, we will continue to explore the pros and cons of such applications design on wireless network based on highly constrained devices with a perspective of M2M oriented approach.

Acknowledgment

The authors would like to thank Cyril Nicaud for his expertise on trees, distributions and for the mathematical formulation of the problem.

References

[1] Zigbee alliance. http://www.zigbee.org/.

[2] I. Akyildiz and I. Kasimoglu. Wireless sensor and actor networks: research challenges. *Ad hoc networks*, 2(4):351–367, 2004.

[3] B. "Alkazemi and E. Felemban. "towards a framework for engineering software development of sensor nodes in wireless sensor networks". In *"Proceedings of the 2010 ICSE Workshop on Software Engineering for Sensor Network Applications"*, pages 72–75. ACM, 2010.

[4] A. Barros, M. Dumas, and P. Oaks. Standards for web service choreography and orchestration: Status and perspectives. In *Business Process Management Workshops*, pages 61–74. Springer, 2006.

[5] A. Dunkels, B. Gronvall, and T. Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. local computer networks. In *Annual IEEE Conference on, 0*, pages 455–462, 2004.

[6] A. Dunkels, T. Voigt, and J. Alonso. Making TCP/IP viable for wireless sensor networks. In *Proceedings of the First European Workshop on Wireless Sensor Networks (EWSN 2004), work-in-progress session, Berlin, Germany*. Citeseer, 2004.

[7] S. Hadim and N. Mohamed. Middleware: Middleware challenges and approaches for wireless sensor networks. *IEEE Distributed Systems Online*, 7(3):1–23, 2006.

[8] M. Kuorilehto, M. Hännikäinen, and T. Hämäläinen. A survey of application distribution in wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2005(5):774–788, 2005.

[9] A. Nayak and I. Stojmenović. *"Wireless sensor and actuator networks: algorithms and protocols for scalable coordination and data communication"*. Wiley-Interscience, 2009.

[10] Z. Shelby. Embedded web services. *Wireless Communications, IEEE*, 17(6):52–57, 2010.

[11] T. "Watteyne, A. Molinaro, M. Richichi, and M. Dohler. "from manet to ietf roll standardization: A paradigm shift in wsn routing protocols". *"Communications Surveys & Tutorials, IEEE"*, (99):1–20, 2010.