A Generalized Distributed Consensus Algorithm for Monitoring and Decision Making in the IoT

Denis Carvin[1,2a], Philippe Owezarski[1,2], Pascal Berthou[1,2b]
[1]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[2]Univ de Toulouse, [a]INSA, [b]UPS, F-31400 Toulouse, France
Email: {carvin, owe, berthou}@laas.fr

***Abstract*—In this paper, we propose a method to distributively monitor a dynamic mobile network. For this purpose, we take advantage of the consensus theory to provide each node with a common view of the network. More specifically, we give a decentralized algorithm to estimate a time-varying distribution, where each node has a partial information on this distribution. Our algorithm allows a trade-off between the precision of its estimation and its bandwidth consumption. We validate our approach by simulation under NS3, considering the distribution of several network metrics. Simulation results demonstrate how our algorithm can give insights on the network behavior that could be exploitable in a decision making process.**

I. Introduction

The work presented in this paper considers the most general case of the IoT. All nodes are objects that can play three roles: they can produce data and send them to a particular receiver in the network, they can be the receivers of data produced by other objects in the network, and they can also serve as routers by being in charge of forwarding data packets with the required QoS. If these objects are mobile, with a mobility model that depends on their specific task, some of them could adapt their moves in favor of the network quality. Using such an approach, it could be possible for nodes to act for the benefit of the whole network in terms of QoS and resilience. To achieve such a behavior, objects must be able to monitor the local state of the network, and to infer the global status of the network by collaborating with their neighbors. For the seek of coordination, objects need to share the same view of the state of the network, in others words, they need to reach a *consensus* on this value. As it is often done in network monitoring, we derive the network's state from metrics that are measured at the node level. In particular, in this paper, we define the network's state as the distribution of metrics that have been periodically measured on each node. Our main contribution is to provide a decentralized algorithm to estimate this time-varying distribution when nodes are mobile.

The following section briefly reviews basic notions of descriptive statistic and introduces the related work. In Section III, we describe the consensus framework and present our decentralized density estimation method. We apply this method in the case of a wireless dynamic network of objects in Section IV, where we present our simulation results. Finally, in Section V, we discuss possible applications of inter-controlled mobility and give future directions.

II. Theoretical Background and Related Work

This section reviews basic notions of descriptive statistics as well as related work on the decentralized density estimation, which will be used throughout the paper.

A. Density Estimation

When measuring a metric in a network, knowing the average might be useful, but it is often said that the average value of a population is meaningless without its standard deviation. In fact, these parameters are only useful for someone who has prior knowledge on the underlying distribution. For instance, average and standard deviation fully characterize a normal distribution, but if the distribution is unknown and appears to be multi-modal, these parameters might be badly interpreted. Moreover, if one wants to deal with qualitative data, the average is not computable and other approaches must be used. Non-parametric estimation techniques can overcome this problem. We mostly associate them to bar-chart (qualitative data) or histogram (continuous data). Kernel Density Estimation (KDE) encompasses those methods which estimate a probability density function (*pdf*) without prior knowledge, in exchange for an additional computational cost.

In the case of continuous data, given a set $\mathcal{P} \subset \mathbb{R}^d$ of P d-dimensional points, drawn from an unknown *pdf*, f, the kernel density estimation $\hat{f}$ of f can be defined as

$$\hat{f}(\mathbf{p}) = \sum_{k=1}^{P} \alpha_k K'_k(\mathbf{p}), \tag{1}$$

where α_k and K'_k are the weight and the scaled kernel function associated to the sample (or location) $\mathbf{p}_k$, that is, $K'_k(\mathbf{p}) = |\mathbf{H}_k|^{\frac{-1}{2}} K_k(\mathbf{H}_k^{\frac{-1}{2}}(\mathbf{p} - \mathbf{p}_k))$. The scaled kernel function K'_k can be seen as the contribution of the sample $\mathbf{p}_k$ to the estimation of the density at the location $\mathbf{p}$. This contribution is defined by a bandwidth $\mathbf{H}_k$ which is a d-dimensional matrix and by a kernel function K_k which is symmetric and that integrates to one. In many cases, $\mathbf{H}_i$ and K_k are the same for all k. Besides, $\sum_{k=1}^{P} \alpha_k = 1$ and in general we have $\alpha_k = \frac{1}{P}$. It is important to note that choosing the bandwidth as well as the Kernel is crucial in the quality of the estimation, nevertheless, these aspects are beyond the scope of this paper. For the sake of simplicity, we consider the Gaussian kernel $G(\mathbf{p}) = (2\pi)^{-d/2} e^{\frac{-\mathbf{p}^T\mathbf{p}}{2}}$, and each node locally estimates the optimal bandwidth using the algorithm given by Scott in [1],

in practice this is an open choice. When data are qualitative or categorical, sample points $\mathbf{p}_i$ belong to a discrete and finite set of labels or categories. In that special case we can use the following kernel:

$$K'_k(\mathbf{p}) = K_k(\mathbf{p}) = \begin{cases} 1 & \text{if } \mathbf{p}_k = \mathbf{p}, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

B. Related work

There exist several lines of work that deal with the decentralized estimation of a distribution. We can classify them in two categories: parametric-estimation and non-parametric estimation. In the first case assumptions are made on the underlying distribution which can be determined by a set of parameters. The objective is to determine the parameters' value that minimize a given criteria, like the Maximum Likelihood. Specific Algorithms to distributively compute a Gaussian Mixture Model are given by Nowak in [2] and Jiang et al. in [3]. Other solutions proposed by Kowalczyk, Vlassis and Sfakianakis use a gossip protocol framework [4], [5]. Regarding non-parametric estimation, a distributed kernel estimation is given by Hu and al. in [6]. This last work also provides a compression method (with information loss) to consider nodes with limited resources. Even if these previous work have proven their efficiency, they are either very specific to the considered problem, not trivial to implement or make strong assumption on the network model that might not stick to the reality of wireless mobile networks. As far as the authors' knowledge, the problem of a time-varying distribution has not been treated yet and the adaptation of previous work to this new assumption could become fairly complex.

We argue that decoupling the diffusion of the information from the information itself facilitates both the understanding and the implementation. Consequently, dealing with the time-varying assumption will become straightforward.

In the next section, we explain how to take advantage of the average consensus to estimate a time-varying *pdf* in a decentralized way.

III. Decentralized Estimation of Time-Varying Distributions

We first introduce the average consensus framework and establish the notation, then we present our contribution.

A. The Average Consensus Framework

The term distributed consensus is often associated with the work of Fischer, Lynch and Paterson [7], but the consensus problem also has a strong background in the field of control theory [8]. In fact, a consensus usually refers to the reach of an agreement between several agents that share a protocol. In the case of the average consensus, the agreement to reach is the average of nodes' states. The theoretical framework to study the consensus problem in a dynamic system of networked agents has been introduced by Olfati-Saber and Murray in [9]. It is now applied in various domains such as synchronization of oscillators [10], load balancing [11], and sensor fusion [12]. In the scope of this paper, we consider network monitoring. The consensus framework can be used under different assumptions on the nature of the exchange between agents which can be either continuous or discrete. In our case, the exchange between agents is materialized by packets, which are discrete by nature.

Network Model: We place our work under a synchronous communication model in which a node iteratively communicates a consensus value at a discrete time index to its neighbors and updates this value by taking into account the values it has received. A network is modeled by a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ for a set $\mathcal{N} = [1, n]$ of n nodes (or agents) and a set $\mathcal{E} \in \mathcal{N}^2$ of edges. An edge $(i, j) \in \mathcal{E}$ materializes the possible reception by the node j of a monitoring packet from node i. The state of node i is $s_i \in S$. We denote by $\bar{s} := \frac{1}{n}\sum_{i=1}^{n} s_i$ the average value of the state over the network. In the case of dynamic states, the value of s_i can change over time, thus $s_i(t)$ stands for the sample of i at the discrete time index $t \in \mathbb{N}^0$. Similarly, we define $\bar{s}(t) := \frac{1}{n}\sum_{i=1}^{n} s_i(t)$. In the case of a dynamic topology, the communication graph evolves with agents' mobility. Agents that were able to communicate may not reach each other, whereas new communication links may appear. In that case, we consider that the set of edges can change over time whereas the set of nodes remains the same, thus, $\mathcal{G}(t) = (\mathcal{N}, \mathcal{E}(t))$ stands for the graph of the network at $t \in \{0, 1, \ldots\}$. Note that, even if $\mathcal{N}$ is constant, we can still model the arrival of a node by adding edges to an isolated node, and the departure of a node by removing its incident edges.

Linear Averaging Iterations: We now review the known results concerning linear iteration for the discrete-time average consensus problem. Existing results differ mainly according to whether the topology and the states of nodes are static (constant) or dynamic (time-varying). In the case of a static topology with static states the authors of [13], [14] rewrite the distributed linear averaging consensus as a product of matrices. At each step, nodes iterate their values using the scheme $x_i(t+1) = \sum_{j=1}^{n} w_{ij} x_j(t)$ where w_{ij} is the weight given by the node i to the node j. In the matrix notation the equation becomes $\mathbf{x}(t+1) = \mathbf{W}\mathbf{x}(t)$, with $\mathbf{x}(0) = \mathbf{s}$. In order to respect the network topology, the values of the weights are constrained by $\mathcal{G}$ and $\forall (i, j) \notin \mathcal{E}$ we have $w_{ij} = 0$. When the topology evolves over time, the iteration becomes $\mathbf{x}(t+1) = \mathbf{W}(t)\mathbf{x}(t)$, where $\mathbf{W}(t)$ is constrained by $\mathcal{E}(t)$. It is said that the consensus converges if $\lim_{t\to\infty} \mathbf{x}(t) = \bar{s}$ which is equivalent to the matrix equation $\lim_{t\to\infty} \prod_{k=0}^{t} \mathbf{W}(k) = \frac{\mathbf{1}\mathbf{1}^T}{n}$.

In our context, states and topology are both dynamic. This case is treated in [15]. The authors propose a n-order linear algorithm but we only consider the First Order Dynamic Average Consensus (FODAC) algorithm. The FODAC iteration can be re-written in the following way,

$$x_i(t+1) = \delta s_i(t) + \sum_{j=1}^{n} w_{ij} x_j(t), \quad (3)$$

where $\delta s_i(t) = s_i(t) - s_i(t-1)$. The authors in [15] derive

an upper bound on the steady state error which is valid under the following assumptions:

(A.0) the weighting matrix $\mathbf{W}(t)$ is doubly stochastic,
(A.1) the first order dynamic of states is relatively bounded,
(A.2) the network has a periodical strong connectivity.

The assumption (A.1) formally means that

$$\exists B \in S \quad | \quad \forall t \quad \max_i(\delta s_i(t)) - \min_i(\delta s_i(t)) < B.$$

Regarding the assumption (A.2), a succession of graphs $\mathcal{G}_1$, $\mathcal{G}_2$, ..., $\mathcal{G}_n$ with the same set of nodes N, has a periodical strong connectivity if $\exists P \in \mathbb{N}^* \quad | \quad \forall t \quad \mathcal{G}_u = (N, \cup_{i=t}^{t+P} \mathcal{E}_i)$ is strongly connected.

In the remainder of the paper, we use the consistent metropolis weighting policy from [14], which is an heuristic that shows better convergence than its concurrents and verifies assumption (A.0). This policy defines the weights that nodes give to their neighbors (and consequently the weighting matrix $\mathbf{W}(t)$) as follows:

$$w_{ij}(t) = \begin{cases} 1/(1+\max(d_i(t), d_j(t))) & \text{if } i \in V_j(t), \\ 1 - \sum_{j \neq i} w_{ij}(t) & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where $V_i(t)$ is the neighborhood of node i at step t and $d_i(t)$ its associated cardinal (degree of node i). Note that this policy reasonably assumes that the graph is undirected, which means that, node i considers the value of node j if and only if it is reciprocal. The assumption (A.1) is respected since the states we consider have bounded norms. Finally, the verification of assumption (A.2) relies on the network under study and will determine the convergence towards the average of states.

B. Taking advantage of the Consensus Framework

As we just explained, the dynamic average consensus framework allows nodes to track the average of their states while the communication graph can evolve over time. Note that Equation (3) only requires states to be part of a vector space. Our approach is to rewrite the decentralized density estimation problem in an average consensus problem. To do so, we define the set of states in such a way that their average is the estimation of the distribution we want to track. It means that we need to provide the state space with an internal addition and a scalar multiplication that respect the axioms of a vector space. The main advantages of this approach are the immediacy of convergence proofs and, regarding practical aspects, the triviality of an oriented object implementation. The following subsection explains the transformation of a density estimation problem into an average consensus problem.

C. Average Consensus over Kernel Functions

When looking at Equation (1), it is clear that the kernel density estimation of a *pdf* is a weighted average of functions. Furthermore, if we set $\alpha_i = \frac{1}{P}$, it becomes a simple average. Thus, if we define the state of an agent with the correct function, it results that the FODAC algorithm will track the value $\hat{f}$. Hence, it follows that we need to model a function space, its addition and scalar multiplication.

Dictionary Representation: In our case functions are linear combinations of scaled kernels fully characterized by a tuple $(K_k, \mathbf{H}_k, \mathbf{p}_k)$. As a result, we represent a function f with a dictionary (or associative array), $D_f : \mathcal{K}_f \to \mathcal{V}_f$ where keys $c \in \mathcal{K}_f$ are tuples characterizing scaled kernels and values $v \in \mathcal{V}_f \subset \mathbb{R}^*$ are the corresponding nonzero real coefficients. In the remainder of the section we consider D_f,D_g, D_h as the representation of three functions f, g, h. $D_\emptyset$ stands for the empty dictionary.

Addition: We define the addition between dictionaries as follows.

$$\begin{gathered} h = f + g \\ \iff \\ \mathcal{K}_h = \mathcal{K}_f \cup \mathcal{K}_g - \{c \in \mathcal{K}_f \cap \mathcal{K}_g \,|\, D_f(c) + D_g(c) = 0\}, \\ \text{and } \forall c \in \mathcal{K}_h \text{ we have} \\ D_h(c) = \begin{cases} D_f(c) & \text{if } c \in \mathcal{K}_f \cap \overline{\mathcal{K}}_g \\ D_g(c) & \text{if } c \in \overline{\mathcal{K}}_f \cap \mathcal{K}_g \\ D_f(c) + D_g(c) & \text{else.} \end{cases} \end{gathered} \tag{5}$$

Dictionaries are merged, and for each common key, values are summed, if the sum is null, the key is removed.

Scalar Multiplication: We define the scalar multiplication as follows for $\gamma \in \mathbb{R}^*$:

$$\begin{aligned} &h = \gamma f \iff \mathcal{K}_h = \mathcal{K}_f, \text{ and } \forall c \in \mathcal{K}_h, \quad D_h(c) = \gamma D_f(c), \\ &h = 0f \iff \mathcal{K}_h = \emptyset, \end{aligned} \tag{6}$$

which means that for each key, the value is multiplied by the scalar, if this scalar is zero, then the dictionary is cleared.

Satisfy Axioms: The way we define addition and scalar multiplications allows to satisfy axioms of vector spaces. We will not detail their proof which are trivial.

Initial Value and Convergence: The state of agents are functions that can vary over time. This variation can be due, for instance, to the evolution of the observed set of sample points, their associated kernel or their associated bandwidth. However, for the seek of clarity, we simplify the notation by using s_i, x_i, $\hat{f}_i$ and $\hat{f}$ instead of $(s_i(t))(\mathbf{p})$, $(x_i(t))(\mathbf{p})$, $(\hat{f}_i(t))(\mathbf{p})$ and $(\hat{f}(t))(\mathbf{p})$, respectively. Let $\mathcal{P}$ be the set of P sample points partitioned between agents where each of them has a subset $\mathcal{P}_i \subset \mathcal{P}$ of P_i samples. We denote $K'_{i,r}$ the scaled kernel that agent i associates to its r^{th} sample point and we organize the scaled kernels in a multiset $(\mathcal{K}', m)$ where $\mathcal{K}' = \bigcup_{i,r}\{K'_{i,r}\}$ and m is the multiplicity function. Thus, the centralized KDE of the distribution given the scaled kernel of all the agents with equal weight is

$$\begin{gathered} \hat{f} = \frac{1}{P}\sum_{i=1}^{N}\sum_{r=1}^{P_i} K'_{i,r} = \frac{1}{P}\sum_{K' \in \mathcal{K}'} m(K')K', \\ \text{where } P = \sum_{i=1}^{N} P_i = \sum_{K' \in \mathcal{K}'} m(K'). \end{gathered} \tag{7}$$

We now define the state of agent i by $s_i = \sum_{k=1}^{P_i} K'_{i,k}$. Since the consensus value x_i of node i tracks the value $\bar{s}$, we have

$$x_i \approx \bar{s} = \frac{1}{N}\sum_{i=1}^{N} s_i = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{P_i} K'_{i,k}, \quad \forall i. \tag{8}$$

By identifying $\hat{f}$ in the right hand side of (8) we obtain

$$x_i \approx \bar{s} = \frac{P}{N}\hat{f} = \sum_{K' \in \mathcal{K}'} \frac{m(K')}{N} K'. \tag{9}$$

This last equation states that each node participating to the FODAC algorithm asymptotically estimates the value of $\hat{f}$ with a scaling factor $\frac{P}{N}$, which is the average number of sample points per agent. Each $(K', \frac{m(K')}{N})$ is identifiable to a *(key,value)* couple in the dictionary representation of $\hat{f}$. As a result, the sum over the value of this dictionary is $\sum_{K' \in \mathcal{K}'} \frac{m(K')}{N} = \frac{P}{N}$. Hence, each node can compute its own scaling factor by summing all the values of its dictionary.

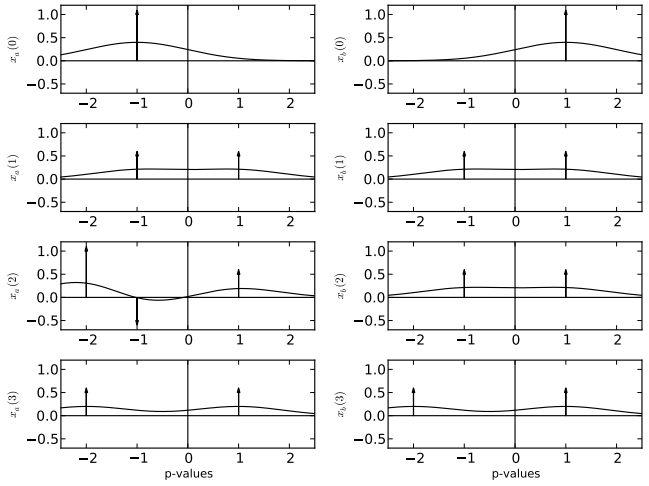

Fig. 1. Four First Estimation Step of a 2-Nodes Network

A Trivial Example: To illustrate this framework, we consider the trivial scenario where two agents a and b are always connected. In this configuration the weighting matrix is constant and $\mathbf{W}(t) = \frac{\mathbf{1}\mathbf{1}^T}{2}$, agents exchange their value and compute the average. At a given time $t \in \{1, 2, \ldots\}$, a and b observe their own unique sample $p_a(t)$ and $p_b(t)$ in $\mathbb{R}$. We choose the following values $p_a(t) = -1, -2, -2, -2, ...$ and $p_b(t) = 1, 1, 1, 1, ...$ Figure 1 shows the Gaussian estimation x_a (left) and x_b (right) of agents for the first four steps and the corresponding dictionaries. In the dictionary representation, a key is illustrated by the position of a row, whereas its value is represented by its length and direction.

Remarks: One can notice that the consensus algorithm given by Equation (3) introduces a difference between states (equivalent to the introduction of negative kernels), δs_i. It results from this difference that x_i is not necessarily positive and consequently is not a *pdf* (as usually defined). However, this is not a concern since x_i is an estimation of $\hat{f}$. This effect is mainly due to transient nodes' states, it vanishes with the stabilization of the set of observations, as illustrated for $x_a(t)$, for $t > 1$ in Figure 1.

D. The Precision-Resources Trade-off

Besides computational resources, a consensus algorithm requires memory, and communication bandwidth. The required memory is directly linked to the size of the representation of the consensus value. The required communication bandwidth is evaluated by multiplying this size with the rate at which messages are exchanged. We suggest to adjust these parameters, that, also affect the precision of the estimation. For instance, the reduction of the exchanges' rate, save some communication bandwidth. However, if this rate is too low, the FODAC algorithm faces an undersampling situation and the steady state error evaluated using the upper bound in [15] will be important. Reducing the size of the consensus representation is another solution to save both communication and memory resources, this reduction implies information loss and also degrade the estimation.

The size of the representation of the *pdf* follows an $\mathcal{O}(|\mathcal{K}'|)$ progression. Intuitively, we can say that the kernel set of a constant distribution is collected in a number of step equal to the network's diameter, increasing quickly the size of the exchanged messages. This is specifically true in the case of a time-varying distribution where new kernels are regularly introduced, while older might never disappear. To tackle this side-effect, Hu and al suggested in [6] the use of compression algorithm between exchanges. While these methods are applicable for constant distribution, they should be carefully applied in our case. Using a compression algorithm is equivalent to changing the vector space addition, and in the meantime losing its associativity. Consequently, the properties of the FODAC algorithm are not guaranteed anymore, possibly letting nodes progressively derive from the average. Actually, an other way to proceed that conserve the addition's associativity is to limit the size of the kernel set. Indeed, the representation of the consensus only grows with the introduction of a new kernel, thus, using a finite set of kernels will limit this growth. As a result, our approach is to bound and discretize the set of kernel, or equivalently, to use categorical kernels.

IV. Monitoring of Wireless Mobile Networks

Our method has been designed to monitor wireless network, thus, we evaluate it through NS-3 simulations. This section describes the considered scenario. Then, we present our estimation results.

A. Network Topology and Traffic

We simulate a wireless mobile ad-hoc network during 100 seconds. For this purpose, we created two primary geographical zone and placed 60 nodes in each of them. The size of the zones are set such as nodes from the same zone are (most of the time) in the same graph component. Indeed, in our case, the communication graph can be modeled by a

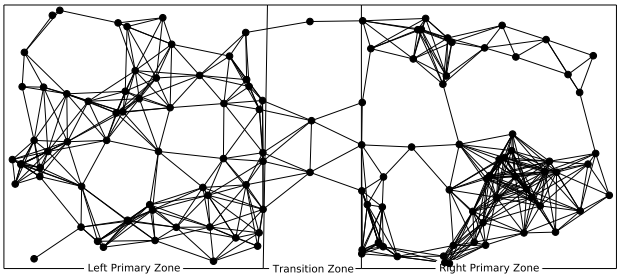

Fig. 2. Considered Network Topology With 3 Transition Nodes

random geometric graph, thus its probability to have a unique component depends on the communication range, the size of a zone and the number of nodes. The two primary zones are separated by a transit zone, as illustrated in Figure 2. Nodes in different primary zones can not reach each other without the help of a node in the transit zone. All Nodes follow a random-way point mobility model. During the simulation they move in their zone between 1 and 3 times at the average speed of 4.0m/s. Nodes have a unique wireless interface, their Medium Access Control and Physical layers are simulated using the NS-3 YansWifi model where controllers are set in an ad-hoc mode and use the adaptive auto rate fallback algorithm without any quality of service. Routes are discovered through the use of AODV. In terms of traffic, each node chooses 3 destinations to which it sends an *ON/OFF* Constant Bit Rate (CBR) UDP traffic. During the *ON* period, all sources use the same data rate (10KB/s) and packet size (1400 bytes).

B. Network Metrics and Consensus

We assumed that each second, a node knows its own position and collects metrics that are derived from its local traffic (and in practice, could be extracted from their Management Information Base). Every consensus period τ, node communicate the information required to update their consensus value (degree and dictionary). The metrics related to the traffic are the number of byte processed by the IP layer from the local UDP sources ($sent$), the number of bytes processed by the IP layer from other sources (fwd) and the number of bytes dropped by the IP layer ($dropped$). To describe the local level of satisfaction, we derived from these three metrics a trivial ratio $sat = \frac{sent+fwd}{sent+fwd+dropped}$, which decreases when a node is not able to process a packet (route problem, overload...). We conducted three dynamic average consensus, the first one tracks the average value of the sat metric, the second tracks its distribution in the whole network. In a last consensus nodes estimate the geographical (multivariate) density of their population.

C. Simulation Results

In this subsection, we successively explain the three consensus and illustrate each of them by a figure.

1) Estimation of the Average Satisfaction: A simple solution for a node to estimate the average value of a metric is to use its local value. In that case the error is given by the mean absolute deviation. In this first consensus each node estimates the average value of their satisfaction. Figure 3 shows how the mean absolute error is reduced by the FODAC algorithm and illustrates the trade-off that exists between its precision and its bandwidth consumption. On the top left sub-figure we plot the satisfaction $sat_i(t)$ of each node i every second as well as their average. The average satisfaction decreases progressively due to nodes' dynamic and to the start of UDP sources. The top right and the bottom left sub-figures represent the FODAC estimation $x_i(t)$, of this average for the consensus period $\tau = 0.33s$ and $\tau = 0.1s$, respectively. We can notice a reduction of the average error with the consensus period. In the last 20s, one node estimated the average satisfaction to zero, it was a disconnected node in the transit zone. On the last sub-figure we detail the mean absolute error $||\mathbf{x}(t+1) - \mathbf{1}.\bar{x}(t)||_1/n$ for several magnitudes of period consensus. From this last sub-figure, we can intuitively say, that at some point, in terms of information, the reduction of the uncertainty does not worth the price of the additional required bandwidth. We let this optimal Shanon trade-off as an open question.

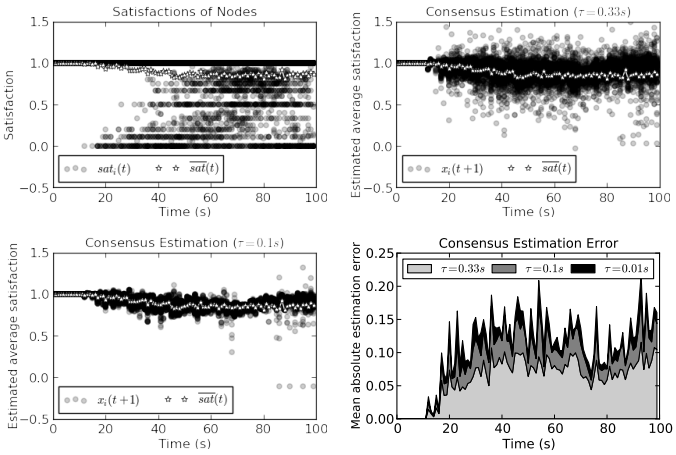

Fig. 3. Estimation of The Average Satisfaction

2) Estimation of Satisfaction Distribution: Our main contribution is to estimate a time varying distribution. We illustrate this estimation in Figure 5 with a consensus period $\tau = 0.1s$. The top sub-figure is the estimation of the satisfaction distribution across time (which is similar to the centralized estimation) for a node located in the left zone. The bottom sub-figure is the estimation of the same distribution for the node located in the transit zone that was not well connected with its neighbors during the last 20 seconds. At the beginning of the simulation nodes estimate that all the network is fully satisfied. This distribution evolves over time and both nodes are aware of this phenomena and agree on a new estimation. As expected their estimation differs during the last 20 seconds because the node in the transit zone is disconnected. However, since assumption (A.2) is respected, this node recovers the estimation when it joins the network.

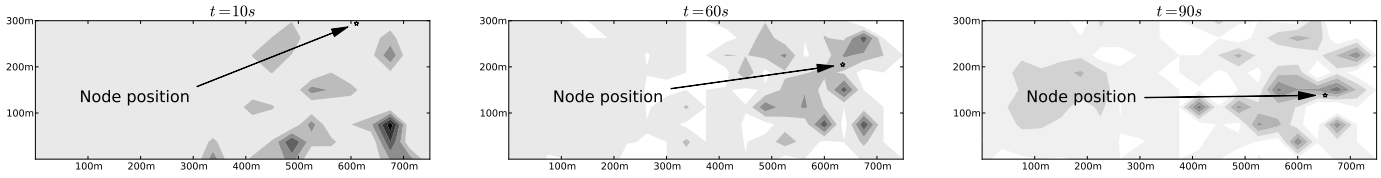

Fig. 4. Geographical Density Estimation of a Node Moving in the Right Zone at $t = 10s$, $t = 60s$ and $t = 90s$ (From Left to Right) with 216 kernels

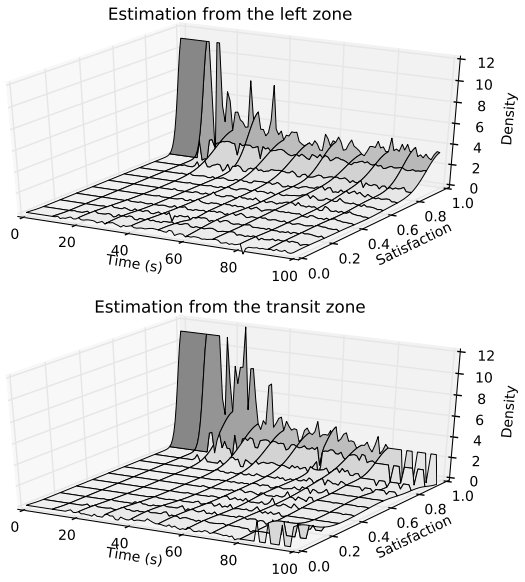

Fig. 5. Estimation of Satisfaction Distribution

3) Estimation of The Geographical Density: Finally, we consider the estimation of a time varying multivariate density. For this last case, we consider the geographical repartition of nodes over the map. As we noticed in section III-D the size of messages grows with the cardinal of the kernel set. Since most of compression method could not be used without loosing the addition's associativity, our technique is to limit the kernel set. In this case, all nodes consider a set of 216 kernels whose locations are uniformly distributed on the map ($\approx 9 \times 10^{-4}$ kernel/m^2). Instead of communicating the kernel associated to their own position, node use the nearest kernel in the kernel set. Figure 4 is the geographical density estimation of a node located in the right zone at different instants. During the simulation, transit nodes relay the information between zones, and the node improve its view of the entire map over time, being able to distinguish the different zones. An interesting point is that it has a precise view of its neighborhood while approximations only concerns distant zones.

V. Conclusion

Motivated by the distributed monitoring of networks with dynamic topology, we developed in this paper a method to estimate a time varying multivariate distribution in a decentralized way. Based on their local measurement and that of their neighbors, nodes iteratively exchange messages to track a distribution in the network. Our work relies on the $f-$consensus theory from which we can derive convergence conditions and error bounds. The resulting method allows a trade-off between precision, and resource consumption. Simulations showed that using several consensus, nodes could behave for the benefit of the whole network. In the present paper, by coupling the satisfaction distribution with the geographical density, some nodes could decide to move towards the transit zone and increase the global network quality. Finally, we note that the convergence speed is limited by the fact that all nodes are highly mobile and participate to the consensus. However, if we relax those two assumptions, we could build overlay networks and drastically improve the estimation giving the possibility to efficiently monitor large fixed networks.

References

[1] D. W. Scott, *Multivariate density estimation : theory, practice, and visualization*, ser. Wiley series in probability and mathematical statistics. Wiley, 1992.

[2] R. Nowak, "Distributed em algorithms for density estimation and clustering in sensor networks," *Signal Processing, IEEE Transactions on*, vol. 51, no. 8, pp. 2245–2253, Aug 2003.

[3] H. Jiang and S. Jin, "Scalable and robust aggregation techniques for extracting statistical information in sensor networks," 2006.

[4] W. Kowalczyk and N. Vlassis, "Newscast em," in *In NIPS 17*. MIT Press, 2005, pp. 713–720.

[5] N. Vlassis, Y. Sfakianakis, and W. Kowalczyk, "Gossip-based greedy gaussian mixture learning," in *10th Panhellenic Conf. on Informatics*, 2005.

[6] Y. Hu, J.-G. Lou, H. Chen, and J. Li, "Distributed density estimation using non-parametric statistics," in *Distributed Computing Systems, 2007. ICDCS '07. 27th International Conference on*, June 2007, pp. 28–28.

[7] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, no. 2, pp. 374–382, Apr. 1985.

[8] V. Borkar and P. Varaiya, "Asymptotic agreement in distributed estimation," *Automatic Control, IEEE Transactions on*, vol. 27, no. 3, pp. 650–655, Jun 1982.

[9] R. Saber and R. Murray, "Consensus protocols for networks of dynamic agents," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 2, June 2003, pp. 951–956.

[10] V. Preciado and G. C. Verghese, "Synchronization in generalized erdsrnyi networks of nonlinear oscillators," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, Dec 2005, pp. 4628–4633.

[11] C. Xu and F. C. Lau, *Load Balancing in Parallel Computers: Theory and Practice*. Norwell, MA, USA: Kluwer Academic Publishers, 1997.

[12] S. Martinez, "Distributed representation of spatial fields through an adaptive interpolation scheme," in *American Control Conference, 2007. ACC '07*, July 2007, pp. 2750–2755.

[13] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, 2003.

[14] L. Xiao, S. Boyd, and S. Lall, "Distributed average consensus with time-varying metropolis weights," 2006.

[15] M. Zhu and S. Martnez, "Discrete-time dynamic average consensus," *Automatica*, vol. 46, no. 2, pp. 322 – 329, 2010.