Sensor Field: A computational model*

C. Àlvarez, A. Duch, J. Gabarro, and M. Serna

ALBCOM. LSI Dept. Universitat Politècnica de Catalunya, Barcelona.
{alvarez,duch,gabarro,mjserna}@lsi.upc.edu

Abstract. We introduce a formal model of computation for networks of tiny artifacts, the static synchronous sensor field model (SSSF) which considers that the devices communicate through a fixed communication graph and interact with the environment through input/output data streams. We analyze the performance of SSSFs solving two sensing problems the Average Monitoring and the Alerting problems. For constant memory SSSFs we show that the set of recognized languages is contained in DSPACE($n+m$) where n is the number of nodes of the communication graph and m its number of edges. Finally we explore the capabilities of SSSFs having sensing and additional non-sensing constant memory devices.

1 Introduction

The use of networks of tiny artifacts is becoming a key ingredient in the technological development of XXI century societies. An example of those networks are the networks with sensors, where some of the artifacts have the ability of sensing the environment and communicate among themselves. Naturally there is no easy way to design a universal sensor network that acts properly in all possible situations. However, it is important to understand the computational process and behavior of the different types of artifact's networks, which will help in taking the maximum profit of those networks. In the particular case of networks with sensors several proposals (taxonomies and surveys) that elucidate their distinguishing features and applications have been published ([1, 5, 12, 14]). These proposals state clearly the need of formal models that capture the clue characteristics of sensor networks.

The general sensing setting can be described by two elements: the observers or end users and the phenomenon, the entity of interest to the observers that is monitored and analyzed by a network with sensors. The corresponding information is discretized in two ways: first the environment is sampled on a discrete set of locations (sensor positions), and second the measures taken by the sensors are digitalized to the corresponding precision. To analyze the correctness and performance of the system we are faced with a double task; on one side there

* Partially supported by the ICT Program of the EU under contract number 215270 (FRONTS). The first, third, and fourth authors were supported by MEC TIN-2007-66523 (FORMALISM) and the second author by MEC TIN2006-11345 (ALINEX-2).

is a computational problem to be solved by a particular network; on the other hand, it is necessary to assess whether a computed solution is a valid observation of the phenomenon. Both tasks will require different analysis tools and we concentrate here on the first. The distinctive peculiarities of the computational system define new parameters to be evaluated in order to measure the performance of the system. Metrics are needed to allow us to estimate the suitability of an specific or generic network topology or the possibility of emergent behavior with pre-specified requirements.

The computational system can be modeled by combining the notion of graph automata [3] together with distributed data streams [6], a combination inspired in similar ideas developed in the context of concurrent programming [8]. Existing models coming from distributed systems [11], hybrid systems and ad-hoc networks [7, 10] capture some of such networks. Models coming from the area of population protocol models [2, 4] represent sensor networks, supposing that the corresponding sensing devices are extremely limited mobile agents (a finite state machine) that interact only in pairs by means of a communication graph.

We propose a general model capturing some characteristic features of sensor networks. A sensor field is composed by devices that can communicate one to the other and also to the environment. We concentrate our initial study in the case in which the devices and the communication links are static: do not appear and disappear during the computation. We also assume that those devices synchronize at barriers marking rounds, in a way similar to the BSP model [13]. During a computation round, a device access the received messages and data provided by the environment, performs some computation, and finally sends messages to its neighbors and to the environment. Those are the fundamental characteristics of the *Static and Synchronous Sensor Field* (SSSF) model. The model allows the definition of latency and complexity measures like round duration (called time in this work), message number or message length among others. In this setting we can formulate a general and natural definition of sensing problems by means of input/output data streams.

We introduce the Average Monitoring and Alerting problems and, supposing that the sensor field has as many devices as input streams, we analyze solutions for several topologies. We obtain upper and lower bounds on the solutions to the problem and for some concrete topologies we propose optimal algorithms.

Our proposed model can be seen as a non-uniform computational model in the sense that it is easy to introduce constraints to all or some of the devices of the sensor field and relate it to classic complexity classes. By restricting their memory capacity to be constant, we show that the decisional version of the functions computed by this restricted SSSF belong to the class DSPACE$(n+m)$ where n is the number of nodes of the communication graph and m its number of edges. Finally, by restricting the memory capacity to be constant and by allowing the inclusion in the network of non-sensing devices we show that there is a SSSF of polynomial size, time and latency, solving the monitoring problem for a property which is computable in polynomial time.

The paper is organized as follows. In Section 2 we introduce the SSSF model as well as the sensing problems and the performance measures. In Section 3 we study SSSFs solving the Average Monitoring problem. The SSSF with restricted memory capacity of the devices (specifically supposing that it is logarithmic or constant), is analyzed in Section 4. We then extend the studied networks to include non sensing devices in Section 5. Finally, in Section 6 we post some conclusions and possibilities of future work.

2 Static Synchronous Sensor Field: the model

A *data stream* w is a sequence of data items $w = w^1 w^2 \dots w^i \dots$ that can be infinite. For any $i \geq 1$, $w[i]$ denotes the i-th element of w, i.e. $w[i] = w^i$. For any i, j, $1 \leq i \leq j$, $w[i, j]$ denotes the subsequence of w composed by all data items between the i-th and j-th positions, i.e. $w[i, j] = w^i \dots w^j$. For any $n \geq 1$, an n-data stream $\mathbf{w}$ is an n-tuple of data streams, $\mathbf{w} = (w_1, \dots, w_n)$. For any $i \geq 1$, $\mathbf{w}[i]$ denotes the n-tuple composed by all the i-th elements of each data stream, $\mathbf{w}[i] = (w_1[i], \dots, w_n[i])$. For any i, j such that $1 \leq i \leq j$, $\mathbf{w}[i, j]$ denotes the n-tuple composed by the subsequences between the i-th and j-th positions of each data stream, $\mathbf{w}[i, j] = (w_1[i, j], \dots w_n[i, j])$.

We use the standard graph notation. A *communication graph* is a directed graph $G = (N, E)$ where N is the set of nodes and E is the set of edges, $E \subseteq N \times N$. Unless explicitly stated we assume that N has n nodes that are enumerated from 1 to n and m edges. Each node k is associated to a device, let us say to device k, that has access to the k-th data stream. Each edge $(i, j) \in E$ specifies that device i can send messages to device j or what is the same, device j can receive messages from device i. Given a device k let us denote by $I(k) = \{i \mid (i, k) \in E\}$ the set of neighbors from which device k can receive data items and by $O(k) = \{j \mid (k, j) \in E\}$ the set of neighbors to which device k can send data. Let $in_k = |I(k)|$ and $out_k = |O(k)|$ be the in and out degrees of node k. Set $in_G = \max_{k \in N} in_k$ and $out_G = \max_{k \in N} out_k$. We use d_G to denote the diameter of the graph G.

A *Static Synchronous Sensor Field* consists of a *set of devices* and a *communication graph*. The communication graph specifies how the devices communicate one to the other. For the moment and without loose of generality, we assume that all devices are sensing devices that can receive information from the environment and send information to the environment. Since the model we consider is static we assume that the edges are the same during all the computation time. Moreover, each device executes its own process, communicates with their neighbors (devices associated to adjacent nodes) and also with the environment. All the devices work in a synchronous way, at the begining of each round they receive data from their neighbors and from the environment, then they apply their own transition function changing in this way their actual configuration and finish the round sending data to their neighbors and to the environment. Let us describe in detail the main components of the Static Synchronous Sensor Field.

Static Synchronous Sensor Field $\mathcal{F}$ (SSSF $\mathcal{F}$): Formally we define a Static Synchronous Sensor Field $\mathcal{F}$ by a tuple $\mathcal{F} = (N, E, U, V, X, (Q_k, \delta_k)_{k\in N})$ where

- $G_{\mathcal{F}} = (N, E)$ is the communication graph.
- U is the alphabet of data items used to represent the input data streams that can be received from the environment.
- V is the alphabet of items used to represent the output data streams that can be send to the environment.
- X is the alphabet of items used to communicate each device to the other devices. Each $m \in X^*$ is called message or packet. $U, V \subseteq X$. We denote by data items the elements of alphabets U and V and by communication items (or items) the elements of X.
- (Q_k, δ_k) defines for each device associated to a node $k \in N$ (device k) its set of local states and its transition function, respectively.

The *local computation of each device k in $\mathcal{F}$* is defined by (Q_k, δ_k) and depends on the communication with its neighbors and with the environment. Q_k is a (potentially infinite) set of local states and δ_k is a transition function. A state codifies the values of some local set of variables (ordinary program variables, message buffers, program counters ...) and all what is needed to describe completely the instantaneous configuration of the local computation. The transition function δ_k depends on its local state $q_k \in Q_k$ as well as on:

- the communication items received by device k from devices $i \in I(k)$,
- the data item that device k receives as input from the environment,
- the communication items sent by device k to devices $j \in O(k)$,
- and the data item that device k sends to the environment.

The transition function is defined as $\delta_k : Q_k \times (X^*)^{in_k} \times U \longrightarrow Q_k \times (X^*)^{out_k} \times V$. The meaning of $\delta_k(q_k, (x_{ik})_{i\in I(k)}, u_k) = (q'_k, (y_{kj})_{j\in O(k)}, v_k)$ is that if device k of $\mathcal{F}$ is in its local state $q_k \in Q_k$, receives $x_{ik} \in X^*$ from each of its neighbors $i \in I(k)$, and receives the input data item $u_k \in U$ from the environment, then in one computation step device k changes its local state to $q'_k \in Q_k$, sends $y_{kj} \in X^*$ to each of its neighbors $j \in O(k)$ and outputs $v_k \in V$ to the environment. In the case that device k does not send any value, we denote this 'no value' or 'does not care' by the special symbol $\perp$. For any device k, let q_k^0 be the initial local state. For any $t \geq 1$, *the t-th computation round of device k* is described as follows: If the local state of device k is q_k^{t-1}, and it receives $(x_{ik}^t)_{i\in I(k)}$ from its input neighbors, u_k^t from the environment and $\delta_k(q_k^{t-1}, (x_{ik}^t)_{i\in I(k)}, u_k^t) = (q_k^t, (y_{kj}^t)_{j\in O(k)}, v_k^t)$ then device k changes its local state from q_k^{t-1} to q_k^t, sends $(y_{kj}^t)_{j\in O(k)}$ to its outgoing neighbors and v_k^t to the environment.

A *computation* of $\mathcal{F}$ is a sequence $\mathbf{c}^0, \mathbf{d}^1, \mathbf{c}^1, \mathbf{d}^2, \ldots, \mathbf{c}^{t-1}, \mathbf{d}^t, \mathbf{c}^t, \ldots$, eventually infinite, where $\mathbf{c}^0 = (q_k^0)_{k\in N}$ is the n-tuple of the initial local states of the n devices, and for each $t \geq 1$, $\mathbf{c}^t = (q_k^t)_{k\in N}$ is the n-tuple of the local states after t computation rounds. The tuple $\mathbf{d}^t = (d_k^t)_{k\in N}$ represents the input/output data of the t-th computation round (i.e. the transition from round $t-1$ to round t). In particular, for device k the input/output data of the t-th round is represented by

$d_k^t = ((x_{ik}^t)_{i\in I(k)}, u_k^t, (y_{kj}^t)_{j\in O(k)}, v_k^t)$. Note that in one round device k receives $(x_{ik}^t)_{i\in I(k)}$ from its neighbors ($x_{ik}^t = y_{ki}^{t-1}$), receives u_k^t from the environment, changes its state from q_k^{t-1} to q_k^t, sends $(y_{kj}^t)_{j\in O(k)}$ to its neighbors and sends v_k^t to the environment.

The *stream behavior of a computation* $\mathbf{c}^0, \mathbf{d}^1, \mathbf{c}^1, \mathbf{d}^2, \ldots, \mathbf{c}^{t-1}, \mathbf{d}^t, \mathbf{c}^t, \ldots$ *of* $\mathcal{F}$ is defined as $(\mathbf{u}, \mathbf{v})$ where $\mathbf{u} = (u_k)_{k\in N}$ is the tuple composed by the input data streams of each device k, $u_k = u_k^1 u_k^2 \ldots u_k^t \ldots$ and $\mathbf{v} = (v_k)_{k\in N}$ is the tuple composed by the output data stream of each device $v_k = v_k^1 v_k^2 \ldots v_k^t \ldots$ Notice that this information can be extracted from the computation $\mathbf{c}^0, \mathbf{d}^1, \ldots, \mathbf{c}^{t-1}, \mathbf{d}^t, \mathbf{c}^t, \ldots$. Thus the sensor field $\mathcal{F}$ outputs the tuple of output data streams $\mathbf{v} = (v_k)_{k\in N}$ given the tuple of input data streams $\mathbf{u} = (u_k)_{k\in N}$ or what is the same, $\mathbf{v}[1,t]$ given $\mathbf{u}[1,t]$ for each $t \geq 1$.

We define the function $f_{\mathcal{F}}$ associated to the stream behavior of $\mathcal{F}$ as follows: Given any pair of tuples of data streams $\mathbf{u}$ and $\mathbf{v}$ and any $t \geq 1$, $f_{\mathcal{F}}(\mathbf{u}[1,t]) = \mathbf{v}[1,t]$ if and only if the sensor field $\mathcal{F}$ computes $\mathbf{v}[1,t]$ given $\mathbf{u}[1,t]$.

Function computed by $\mathcal{F}$: A function f (defined on data streams) is *computed by a sensor field* $\mathcal{F}$ *with latency* d if for all (appropriate) tuple of data streams $\mathbf{u}$, and for all $t \geq 1$, $f_{\mathcal{F}}(\mathbf{u}[1,t+d])[t+d] = f(\mathbf{u}[1,t])[t]$. That is the SSSF outputs at time $t+d$ the t-th element of f. We say that f is *computed by a sensor field* $\mathcal{F}$ if there exists d for which f is computed by $\mathcal{F}$ with latency at most d.

Note that $\mathbf{u}$ and $\mathbf{v}$ have in general infinite length. In order to express formally the behavior of a SSSF we consider all the finite prefixes of the input stream ($\mathbf{u}[1,t]$) and those of the output stream ($\mathbf{v}[1,t]$). However, take into account that each sensor will output only one data item ($\mathbf{v}[t]$) per round.

The computational resources used by a sensor to compute a function of this kind are the following. For each device and computation round we can measure

- *Time*. The number of operations performed in the given round of the device. This is a rough estimation of the "physical time" needed to input data, receive information from other sensor, compute, send information and output data.
- *Space*. The space used by the device in such computation round.
- *Message Length*. The maximum number of data items of a message sent by the device in such computation round.
- *Number of messages*. The maximum number of messages sent by the device in such round.

We consider the following worst case complexity measures taken over any device and computation round of a sensor field $\mathcal{F}$:

– *Size*: The number of nodes or devices of the communication graph G.
– *Time* ($\mathcal{T}$): The maximum time used by any device in any of its rounds.
– *Space* ($\mathcal{S}$): The maximum space used by any device of in any of its rounds.
– *MessageLength* ($\mathcal{L}$): The maximum message length of any device of in any of its rounds.
– *MessageNumber* ($\mathcal{M}$): The maximum number of messages sent by any device in any of its rounds.

In general we analyze these complexity measures with respect to the *Size* of the communication graph which usually will coincide with the number n of data streams, we denote by $\mathcal{T}(n)$ the *Time*, by $\mathcal{S}(n)$ the *Space*, by $\mathcal{L}(n)$ the *MessageLength* and by $\mathcal{M}(n)$ the *MessageNumber*.

Computational problems that are susceptible of being solved by sensor fields can be stated in the following way:

Sensing Problem Π: Given an n-tuple of data streams $\mathbf{u} = (u_k)_{1\leq k\leq n}$ for some $n \geq 1$, compute an m-tuple of data streams $\mathbf{v} = (v_k)_{1\leq k\leq m}$ for some $m \leq n$ such that $R_\Pi(\mathbf{u}[1,t], \mathbf{v}[1,t])$ is satisfied for every $t \geq 1$. R_Π is the relation that output data streams have to satisfy given the input data streams, i.e. the property that defines the problem.

Problem Solved by $\mathcal{F}$: A sensor field $\mathcal{F}$ *solves problem* Π *with latency* d if for every pair of data streams $\mathbf{u}$ and $\mathbf{v}$, and every $t \geq 1$, if $f_\mathcal{F}(\mathbf{u}[1,t]) = \mathbf{v}[1,t]$ then $R_\Pi(\mathbf{u}[1,t], \mathbf{v}[1+d, t+d])$. A sensor field $\mathcal{F}$ *solves the problem* Π if there is a d such that $\mathcal{F}$ solves problem Π with latency d.

Let us post two examples of sensing problems. First we consider a problem in which it is needed to monitor continuously a wide area. This implies "sensing locally" and "informing locally" about a global environmental phenomena.

Average Monitoring: Given n data streams $(u_k)_{1\leq k\leq n}$ for some $n \geq 1$, compute n data streams $(v_k)_{1\leq k\leq n}$ such that $v_k[t] = (u_1[t] + \cdots + u_k[t])/n$.

The second example we consider is related to "fire detection alarm". In this case it is desired to detect the situation in which there is a high risk of fire. One element to be measured is the level of smoke in the air of such area and if this level is higher than a certain value then the alert has to be activated. A specific device (number 1 for instance) acts as a master and outputs the result.

Alerting: Given n data streams $(u_k)_{1\leq k\leq n}$ for some $n \geq 1$, and threshold value A, device 1 has to compute a data stream v_1 such that

$$v_1[t] = \begin{cases} 1 & \text{if } \exists k : 1 \leq k \leq n : u_k[t] \geq A \\ \perp & \text{otherwise} \end{cases}$$

3 SSSFs Solving the Average Monitoring Problem

We study the requirements of a SSSF for solving the Average Monitoring problem. It is divided into two parts. We start by giving some lower bounds on the latency, the *MessageNumber*, and the *MessageLength*, required by some types of SSSF for solving the Average Monitoring. Later we provide optimal algorithms for the problem in particular topologies.

Lower Bounds. In order to be able to state lower bounds, we make an additional assumption: all the sent messages are formed only by tuples of data items (without compression).

Lemma 1. *A SSSF $\mathcal{F}$ with communication graph G solving the Average Monitoring problem requires at least latency d_G.*

Proof. By definition there are at least two nodes (or devices) $i, j \in N$ such that the minimum distance between i and j is d_G, therefore, if at round t node i takes from the environment the data item $u_i[t]$ then, the node j can't receive it in round $t' \leq t + d_G$. □

For the following results we assume, in addition, that along the whole computation the flow of packets from node i to node j, for any $i, j \in N$, follows a fixed path $p_{i,j}$. Thus the algorithm uses a fixed *communication pattern* $P = (p_{i,j})_{i,j \in N}$. We say that the fixed path $p_{i,j}$ is a *diametral path* in P if device j is at distance d_G from device i.

Let $\beta_P(k)$ be the out-degree of node k in the subgraph G' of G formed by all the diametral paths in P starting at k. Set $\beta(G, P) = \max_{k \in N} \beta_P(k)$. Observe that those subgraphs are critical in terms of the delivery of packets in d_G rounds. It is easy to show the following lower bound on *MessageNumber*.

Lemma 2. *Let $\mathcal{F}$ be a SSSF, with communication graph G and communication pattern P, solving the Average Monitoring problem with latency d_G. It holds that for any round $t > d_G$, there is a device sending at least $\beta(G, P)$ packets simultaneously.*

Proof. Any device k has to send simultaneously a packet through at least each of its $\beta_P(k)$ out-going edges in subgraph G'. Otherwise, by Lemma 1, it is not possible for k to send less than $\beta_P(k)$ messages and reach all these devices with latency d_G. Taking the maximum among all the nodes in G we get the bound $\beta(G, P)$. □

Taking into account that the different communication flows must be pipelined along paths with critical length we can prove the following lower bound on *MessageLength*.

Lemma 3. *Let $\mathcal{F}$ be a SSSF, with communication graph G and communication pattern P, solving the Average Monitoring problem with latency d_G. Then, if $p = k_1, \ldots, k_{d+1}$ is the fixed communication path used by devices $k_1, \ldots, k_d$ to send their data to device k_{d+1} in P and p is included in a diametral path in P, then there is a round $t_0 > d$ such that for any round $t > t_0$, there is a device receiving a message composed of at least d data items.*

Proof. Let t be any round such that $t > d$ and $p = k_1, \ldots, k_{d+1}$ the fixed path used by devices $k_1, \ldots, k_d$ to flood their data to device k_{d+1}. Let us suppose that device k_d sends always less than d data items. Then, the number E_d of data items that arrive to k_{d+1} from round 1 to round t is such that $E_d \leq (t-d)(d-1)+1+2+\ldots+d-1+d-1 < (t-d)d+1+2+\ldots+d-1+2d-t$.

The number R_{d+1} of data items that should have been arrived to device k_{d+1} from round $d+1$ to round t corresponding to devices $k_1, \ldots, k_{d+1}$ in order to output the correct average with latency d is $R_{d+1} = d(t-d)$. Taking $t_0 =$

$1+2+\ldots+d-1+2d$ it is easy to see that $E_d < R_{d+1}$, implying that device k_{d+1} has not receive enough data items to output the average, therefore contradicting the hypothesis that device k_d can send less than d data items by round. □

Algorithms. We first propose a generic SSSF with optimal latency provided that the communication graph is strongly connected. In general, this algorithm is not optimal in *Space*, *MessageLength* and *MessageNumber*, we show that when the topology of the communication graph is known in advance, it is possible to obtain SSSFs with specific topologies that optimize such parameters. In what follows, we assume that every device in the SSSF is aware of the total number of devices n and the diameter d_G of the communication graph (if this is not the case, both values can be calculated incrementing the latency in d_G).

Lemma 4. *Let G be a strongly connected communication graph with n nodes. There is a SSSF $\mathcal{F}$ with communication graph G solving the Average Monitoring problem with latency d_G, $\mathcal{T}(n) = O(n\ d_G(in_G + out_G))$, $\mathcal{L}(n) = O(n\ d_G + \log n)$, $\mathcal{S}(n) = O(n\ d_G + \log n)$ and $\mathcal{M}(n) = out_G$.*

Proof. We consider the following flooding algorithm in which each sensor keeps a table M of size $d_G \times n$ of data items. The computation at each sensor is the folowing:

algorithm Generic Average Monitoring
▷ Initially
$M[1\ldots d] \times [1\ldots n] = (\perp)_{d\times n}$
$id =$ identifier of the node
▷ round
// receive
for $i \in I(k)$ {receive M_i from incoming neighbors; $M =$ update (M, M_i)}
// compute
$v = (M[d][1] + \ldots + M[d][n])/n$
for $p = d, \ldots, 2$ $\{M[p] = M[p-1]\}$
$M[1] = (\perp)_n$
$M[1, id] = u$
// send
for $j \in O(k)$ flood M, k
output v
end algorithm

In the generic algorithm, for any device k, table entry $M[\tau, i]$ at round t contains the value $u_i[t-\tau]$, provided that this data has arrived to the node k. The update of table M incorporates to M all the new values received from device i. Observe that, as $d = d_G$, the flooding guarantees that, when the data reaches the last row of M, all the data readings at time $t-d$ are present in the table. □

Algorithms with optimal latency. When the topology of the communication graph is known it is possible to improve the generic algorithm to obtain optimal

algorithms provided that latency is kept at its minimum. The lower bounds follow from Lemmas 1, 2 and 3 taking into account the considered topologies.

Theorem 1. *The Average Monitoring problem can be solved with latency d_G and optimal MessageNumber and MessageLength by SSSFs whose communication graph are bidirectional cliques or oriented rings, respectively.*

Proof. The two algorithms that follows are adaptations of the generic average monitoring algorithm and have the restriction that all the sent messages are formed only by tuples of data items (without compression).

In the case of bidirectional cliques the problem can be solved just in one round so the latency is 1 and the complexity measures are $\mathcal{T}(n) = \Theta(n)$, $\mathcal{S}(n) = \Theta(n)$, $\mathcal{L}(n) = \Theta(1)$ and $\mathcal{M}(n) = n - 1$.

In the case of oriented rings we now consider a network with n sensing devices connected into an oriented ring, where sensor k is connected to sensor $(k + 1)$ mod n. In this case the problem can be solved with an optimal latency of $n - 1$ rounds by means of an algorithm that works as follows. At each round, device k take a measure, receives $n - 1$ data items from its predecessor corresponding to measures taken by device at distance i (for $i \in \{1, \ldots, n - 1\}$) at the i-th previous round, computes and outputs the average of the $(n - 1)$th round and sends to its successor the $n - 1$ corresponding measures (the $n - 2$ received from its predecessor plus its new taken one). Hence, the complexity measures are as follows, $\mathcal{T}(n) = \Theta(n)$, $\mathcal{S}(n) = \Theta(n \log n)$, $\mathcal{L}(n) = n - 1$ and $\mathcal{M}(n) = 1$. □

Improving the message length. By data aggregation or allowing a larger latency, it is possible to improve the *MessageLength*. In this case, messages are no longer tuples of data items but sums of data items. The synchronization needed to compute the right sums can force an increment on the latency.

Theorem 2. *The Average Monitoring problem can be solved with latency $2n-1$, $\mathcal{T}(n) = O(n)$, $\mathcal{S}(n) = O(n \log n)$, $\mathcal{L}(n) = O(\log n)$ and $\mathcal{M}(n) = O(1)$ by a SSSF in which the communication network is an oriented ring.*

Proof. Informally the algorithm works as follows, one device acts as a leader (say device 1) and another device (say device n) computes, collects and distributes the averages. It is assumed that each sensor knows the number of sensors n and its position inside the ring. As before, all the nodes start reading from the environment at the same time.

Device 1 takes and floods its first taken measure to device 2 at round 1. At round 2, device 2 receives the first taken measure of device 1, adds it to its own taken measure at round 1 and forwards the sum to device 3. Eventually, sensor n receives the sum of measures taken by devices $1, 2, \ldots, n - 1$ at round 1, adds it to its own taken measure at round 1 and computes the first meaningful average and forwards it to other devices. □

Theorem 3. *The Average Monitoring problem can be solved with latency d_G, $\mathcal{T}(n) = O(\log n)$, $\mathcal{S}(n) = O((\log n)^2)$, $\mathcal{L}(n) = O(\log n)$ and $\mathcal{M}(n) = \Theta(1)$ by a SSSF in which the communication network is a complete binary tree.*

Proof. In this case the generic algorithm is modified by changing the table of values of each node by two vectors each of size h, where h is the height of the node in the tree (we consider that the leaves have height 1 and therefore the root has height $O(\log n)$). The first vector is used to delay input data items h rounds in order to synchronize with communication items (partial sums) coming from its subtrees and send the partial sum to the father. After some rounds (the height of the tree) the root computes the average corresponding to round t and send it back to its two sons. Children receive this value, store it in the second vector and send it to their children respectively. Averages in the second table have to be delayed in order to fully synchronize the output. □

4 SSSFs of Devices with Constant Memory Capacity

Up to now we have not considered the possible memory restrictions of the *tiny* devices involved in a SSSF, but in applications, devices can have limited memory. The SSSF model can be adapted to take into account this fact, therefore we can consider devices with constant or bounded memory capacity. To this end, we also assume that each device has a buffer of limited size to store the data received from its neighbors. We assume that the communication graph might have any degree, but that a device cannot receive more packets in one round that those that can fit in the buffer. In the case that there are more incoming packets an arbitrary subset of them, filling the buffer, will be retrieved. In the opposite direction we assume that sending data to all the outgoing neighbors can be performed in constant time and space.

The Alerting problem can be solved in constant memory SSSFs by the following algorithm. Initially all the nodes are in a non-alert state. At any round, if an unalerted device receives an alert message or reads a data that provokes an alert changes it state to alert and sends an alert message. An alerted device, different from device 1 does nothing. Device one upon achieving the alert state outputs 1 at each succesive round. Thus we have the following.

Lemma 5. *Let G be a communication graph in which there is a path from any node to node 1. There is a SSSF $\mathcal{F}$ with communication graph G solving the Alerting problem with latency bounded by d_G, $\mathcal{T}(n) = \mathcal{S}(n) = \mathcal{L}(n) = \mathcal{M}(n) = \Theta(1)$.*

In general, we can say that by restricting the memory capacity of each device to be a constant w.r.t. the total number of devices then the kind of problems solved by these SSSFs are not more difficult than the ones in DSPACE($O(n+m)$) In order to prove it formally let us define the decisional version of $f_{\mathcal{F}}$.

Language associated to $\mathcal{F}$: Let $\mathcal{F}$ be a SSSF and let $f_{\mathcal{F}}$ be the function associated to the behavior of $\mathcal{F}$. We define the language associated to the behavior of $\mathcal{F}$, denoted by $L(\mathcal{F})$ as follows:

$$L(\mathcal{F}) = \{\langle \mathbf{u}[1], \mathbf{v}[1], \ldots, \mathbf{u}[t], \mathbf{v}[t]\rangle \mid t \geq 1 \text{ and } f_{\mathcal{F}}(\mathbf{u}[1,t]) = \mathbf{v}[1,t]\}.$$

Theorem 4. *Let $\mathcal{F}$ be a constant space SSSF. Then, the language $L(\mathcal{F}) \in DSPACE(O(n+m))$.*

Proof. We are going to present a deterministic Turing machine M that decides the language $L(\mathcal{F})$ in space $O(n+m)$. Since each device k has constant memory capacity, then the size of Q_k is also bounded by a constant as well as it is the number of items composing each sent or received packet. Recall that the behavior of $\mathcal{F}$ is described by a sequence $\mathbf{c}^0, \mathbf{d}^1, \mathbf{c}^1, \mathbf{d}^2, \ldots, \mathbf{c}^{i-1}, \mathbf{d}^i, \ldots$, eventually infinite, where $\mathbf{c}^0 = (q_k^0)_{k \in N}$ is the n-tuple of the initial local states of the n devices, and for each $i \geq 1$, $\mathbf{c}^i = (q_k^i)_{k \in N}$ is the n-tuple of the local states after i computation rounds. $\mathbf{d}^i = (d_k^i)_{k \in N}$ represents the input/output data and the sent/received messages of each device k in the transition from round $i-1$ to round i. The Turing machine M on any input $\langle \mathbf{u}[1], \mathbf{v}[1], \ldots \mathbf{u}[t], \mathbf{v}[t] \rangle$ will compute such a sequence in the following way:

1. Initially M computes the initial configuration $\mathbf{c}^0$ and suppose that devices have neither received a message nor an input data item.
2. Simulates the i-th computation round computing $(\mathbf{c}^i, \mathbf{d}^{i+1})$ from $(\mathbf{c}^{i-1}, \mathbf{d}^i)$. In order to do this, for each device k, M applies δ_k considering that the input data item is given by $u_k[i]$ and verifies that the output data item is $v_k[i]$. If it is the case then M considers the next computation round $i+1$; otherwise, M rejects.
3. Once M has consumed all its input word, it accepts.

M needs space $O(n+m)$ to decide $L(\mathcal{F})$. Note that in part 2 of the simulation M only needs to store the messages send/received, one for each edge of the communication graph, and it also needs the current state for each one of the nodes of the graph. □

In [4] it is shown that all predicates stably computed in the model of Mediated Population Protocols are in the class of NSPACE($O(m)$). In this case the nondeterminism is required to verify that there exists a stable configuration reachable from the initial configuration.

5 Trading space/time for size

In this section we analyze SSSFs with an additional amount of nodes in the communication graph in which the attached devices participate in the computation but do not play any active role in sensing. In such a network we have a communication graph with S nodes and we want to solve a problem that involves only $n < S$ input data streams.

Constant time*:* In a balanced communication tree we suppose that there are n sensing devices placed on the leaves of a balanced binary tree, edges to leaves are replaced by paths in such a way that all the leaves are at the same distance to the root. Thus, the tree has depth $O(\log n)$ and n leaves. In such a network

we can consider an algorithm with two flows. In the bottom-up flow each node receives from its children the average of the data at the subtree leaves, together with the number of leaves, and computes its corresponding values to be sent to its parent. The top-down computation is initiated by the root that computes the average value which flows to the leaves. The analysis is summarized as follows.

Theorem 5. *Let G be a balanced communication tree whose n leaves are sensing devices with constant space and whose internal nodes are non-sensing devices with $O(\log n)$ space. There is a SSSF $\mathcal{F}$ with communication graph G solving the Average Monitoring problem with latency d_G, $\mathcal{S}(n) = \mathcal{L}(n) = O(\log n)$ and $\mathcal{T}(n) = \mathcal{M}(n) = O(1)$.*

In the algorithm described above the nodes in the communication tree require different levels of internal memory, ranging from constant at the leaves to $\log n$ in the upper levels. The following result shows that by increasing the number of auxiliary nodes we can solve sensing problems with constant memory components in an adequate topology within constant time.

Constant space devices*:* Let $\mathcal{P}$ be a property defined on U^n. We consider the following sensing problem:

Monitoring Problem for property $\mathcal{P}$: Given an n-tuple of data streams $u = (u_k)_{1 \le k \le n}$ for some $n \ge 1$, compute an n-tuple of data streams $v = (v_k)_{1 \le k \le m}$ such that $\mathbf{v}[t] = \mathcal{P}(\mathbf{u}[t])$ for every $t \ge 1$.

Any polynomially computable property can be decide by a uniform family of circuits with polynomial size. Furthermore those circuits can be assumed to be layered and to have bounded fan in and fan out by adding propagator gates. The communication network is formed by the circuit with sensors attached to the corresponding inputs together with a communication tree that flows the result to the inputs. As the circuit is layered we can guarantee the pipelined flow of partial computations with constant time and memory within latency equal to the circuit's depth plus the tree depth. Thus, we have polynomial in n.

Theorem 6. *Let $\mathcal{P}$ be a property defined on U^n computable in polynomial time. There is a constant space SSSF that solves the associated sensing problem in polynomial size and latency (with respect to n) with $\mathcal{S}(n) = \mathcal{T}(n) = \mathcal{L}(n) = \mathcal{M}(n) = O(1)$.*

6 Conclusions and Future Work

We have proposed a model for networks that abstracts some of the main characteristics of the problems that are expected to be solved on a network with sensors. In parallel we have introduced a prototypical family of sensing problems. The model has allowed us to analyze the complexity of some problems providing optimal SSSFs with respect to some performance measures. The analysis shows, as expected, that different sensing problems will require different sensor capabilities for storing data and message size. Our algorithms for the Average Monitoring

problem can be easily adapted to solve the monitoring problem associated to other aggregation functions like: maximum, minimum, addition, median, etc, the complexity of the above algorithms differ depending on the structural properties of the aggregation function (see [9] for a classification) and the size of the aggregated data.

There is a clear trade-off between the internal memory allowed to each device and the number of additional computing units in the network as shown in Lemma 6. It will be of interest to characterize those sensing problems that can be solved with logarithmic or constant space with no (or a small number of) additional nodes.

Although in the present paper the communication network has been assumed to be fixed, the model is flexible enough to allow the incorporation of a dynamic communication graph. Complexity measures and problem solving on such models will require additional effort.

On the other hand the hypothesis of fixed communication graphs models the idea of maintaining a virtual fixed topology, this topology will be maintained until the network task changes. In this situation the communication graph will be perceived as the same graph, although the devices taking care of one node might change over time. Our complexity analysis on fixed topologies should be combined with a study of the conditions that guarantee the existence, creation an maintenance of the virtual topology.

All through the paper we have not considered the energy consumption as a performance measure. For making an energy analysis we should have to incorporate a particular energy model to the sensor field. The performance measures taken in this paper proportionate the basic ingredients for analyzing energy consumption where sending/receiving a message has the same cost for all the nodes, like for example the unit disk graphs. It is of interest (and topic of future research) to consider energy models in which each link in the communication graph has different weights (or set of weights) representing the constants in the function that determines the cost of sending a message along the link.

Acknowledgments We want to thank the anonymous referees for their careful reading and helpful comments.

References

1. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A Survey on Sensor Networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.
2. D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed computing*, 18(4):235–253, 2006.
3. J. Berstel. Quelques applications des reseaux d'automates. Thèse de 3ème Cycle, Faculté des Sciences de Paris, 1967.
4. I. Chatzigiannakis, O. Michail, and P.G. Spirakis. Mediated Population Protocols. In *ICALP 09*, LNCS to appear, 2009.

5. D. Estrin, D. Culler, K. Pister, and G. Sukatme. Connecting the Physical World with Pervasive Networks. *Pervasive Computing*, 6(2):59–69, 2002.
6. P.B. Gibbons and S. Tirthapura. Estimating simple functions on the union of data streams. In *SPAA 01*, 281–291, 2001
7. T.A. Henzinger. The Theory of Hybrid Automata. In *LICS 96*, 278–292, 1996.
8. C.A. R. Hoare. A calculus of total correctness for communicating processes. *Sci. Comput. Program.*, 1(1-2):49–72, 1981.
9. H. Karl and A. Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons Ltd, 2005.
10. N. Lynch, R. Segala, and F. Vaandrager. Hybrid I/O Automata Revisited. *Hybrid Systems: Computation and Control (HSCC'01)*, LNCS 2034:403–417, 2001.
11. D. Peleg. *Distributed Computing. A Locality-Sensitive Approach*, chapter 2. SIAM Monographs on Discrete Mathematics and Applications, 2003.
12. S. Tilak, N.B. Abu-Ghazaleh, and W. Heinzelman. A Taxonomy of Wireless Micro-Sensor Network Models. *Mobile Computing and Communications Review*, 6(2):28–36, 2003.
13. L.G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.
14. M. Vinyals, J.A. Rodriguez-Aguilar, and J. Cerquides. A Survey on Sensor Networks from a Multi-Agent Perspective. In *AAMAS 2008*, 1071–1078, 2008.