Minimum delay Data Gathering in Radio Networks

Jean-Claude Bermond[1], Nicolas Nisse[1], Patricio Reyes[1⋆], and Hervé Rivano[1]

Projet Mascotte, INRIA–I3S(CNRS/UNSA), Sophia Antipolis, France. ⋆⋆

Abstract. The aim of this paper is to design efficient gathering algorithms (data collection) in a Base Station of a wireless multi hop grid network when interferences constraints are present. We suppose the time is slotted and that during one time slot (step) each node can transmit to one of its neighbors at most one data item. Each device is equipped with a half duplex interface; so a node cannot both receive and transmit simultaneously. During a step only non interfering transmissions can be done. In other words, the non interfering calls done during a step will form a matching. The aim is to minimize the number of steps needed to send all messages to the base station, a.k.a. *makespan* or *completion time*. The best known algorithm for grids was a multiplicative 1.5-approximation algorithm [Revah, Segal 08]. In such topologies, we give a very simple +2 approximation algorithm and then a more involved +1 approximation algorithm. Moreover, our algorithms work when no buffering is allowed in intermediary nodes, i.e., when a node receives a message at some step, it must transmit it during the next step.

Keywords: Sensor Networks, gathering, makespan, grid

1 Introduction

We address here the challenging problem of gathering information in a Base Station (denoted BS) of a wireless multi hop grid network when interferences constraints are present. This problem is also known as data collection and is particularly important in sensor networks, but also in access networks. The communication network is modeled by a graph. Here we consider grid topologies as they model well both access networks and also random networks (which approximatively behave like if the nodes were on a grid [1]). We suppose the time is slotted and that during one time slot, or *step*, each node can transmit to one of its neighbors at most one data item (referred in what follows as a message). Each vertex of the grid may have any number of messages to transmit : zero if it is not concerned (sleeping station or no sensor at this node or failed device) one or many. We also suppose that each device (sensor, station,. . .) is equipped with an half duplex interface; so a node cannot both receive and transmit during a step. In particular, this is the case in a mono-frequency smart antennas radio system: at any step, each device can configure its antenna array to shape a beam to reach any of its neighbours, but sending a message would prevent it from receiving because, among other causes, of near-far effects. So we refer to this model as the *smart-antennas model*. During any step only non interfering transmissions can be done, thus the non interfering calls done during a step will form a matching (set of independent edges). Our aim is to design algorithms to do a gathering under such hypotheses, which minimize the minimum number of steps needed to send all messages to BS, a.k.a. *makespan* or *completion time*.

1.1 Related Work

A lot of authors have studied the gathering problem under various assumptions (see the surveys [2] and [3]).

In [4], the smart antennas model is considered with the extra constraint that non buffering is allowed in intermediary nodes. That is, when a node receives a message at some step, it must transmit it during the next step. In this setting, optimal polynomial-time algorithms are presented for path and tree topologies [4, 5]. The work of [4] has been extended to general graphs in [6] and [7] but in the uniform case where each node has exactly one message to transmit. The case of grids is considered in [8] where a 1.5-approximation

⋆ Supported by CONICYT(Chile)/INRIA

⋆⋆ Supported by Project IST-FET IP AEOLUS

algorithm is presented. The gathering problem has also been studied when nodes can both emit and receive a message during the same step. When no buffering is allowed, this problem is known as the hot-potato routing problem and it is considered in [9, 10].

The case of omnidirectional antennas has been extensively studied. In this model, nodes can transmit at any of their neighbours at distance $d_T \geq 1$ but any emission creates some interferences. More precisely, when a node v transmits, any node at distance at most $d_I \geq 0$ of v cannot receive a message from another node than v during the same step. The following papers consider the case $d_I \geq d_T$. Moreover, any node has to transmit at least one message to BS and buffering is allowed. In this setting, computing the makespan is NP-hard [11]. A 4-approximation algorithm and lower bounds for general graphs are also provided in [11]. Moreover, a 4-approximation algorithm has been proposed to handle the online version [12]. In [13], the case of grids is considered when $d_T = 1$: an optimal polynomial-time algorithm is provided when BS stands at the center of the grid. Gathering in grids is also considered within a continuous model in [14].

1.2 Our results

We deal with the gathering problem in grids. We propose a very simple algorithm that achieves at most the makespan plus two, and a more involved +1 approximation algorithm. Our algorithms work as well when no buffering is allowed which considerably improves existing algorithms. Furthermore, following our algorithms a message arrives at most one step (or two steps) after what will happen if we have no interferences (provided that BS can receive only one message per step). So the average time is also very good. We present the results for the smart antennas model and when BS stands at some corner of the grid, but they can be easily extended to any binary distance-based interference model and to the case of any position of BS. Finally, we design a linear-time (in the number of vertices of the grid) distributed algorithm for the +2–approximation algorithm.

One helpful idea is to actually study the related one–to-many personalized broadcast problem in which the BS wants to communicate different data items to some other nodes in the network. Solving the above dissemination problem is equivalent to solve data gathering in sensor networks. Indeed, let T denote the makespan (delay), that is, the largest step used by a personalized broadcast algorithm; a gathering schedule with delay T consists in scheduling a transmission from node y to x during slot t iff the broadcasting algorithm schedules a transmission from node x to y during slot $T - t + 1$, for any t with $1 \leq t \leq T$.

2 Preliminaries

From now on, we consider the equivalent problem of personalized broadcasting where the Base Station BS has to transmit messages to some destination nodes in the grid.

2.1 Notations

In the following, we consider a $N \times N$ grid $G = (V, E)$ where vertices are given there natural coordinates. The base station BS, also called *the source*, has coordinates $(0, 0)$, and any vertex v has coordinates (x_v, y_v). A vertex v is *above* (resp., *below*) $w \in V$ if $y_v \geq y_w$ (resp., if $y_v \leq y_w$). Similarly, v is *to the right* (resp., *to the left*) of $w \in V$ if $x_v \geq x_w$ (resp., if $x_v \leq x_w$). Finally, a vertex v is *nearer to the source* than $w \in V$ is $d(v, BS) \leq d(w, BS)$, where $d(u, v)$ denotes the classical distance between nodes u and v.

We consider a set of $M \geq 0$ messages $\mathcal{M}$ that must be sent from the source BS to some destination nodes. Let $dest(m) \in V$ denote the destination of $m \in \mathcal{M}$. A message $m \in \mathcal{M}$ is *lower* (resp., *higher*) than $m' \in \mathcal{M}$ if $dest(m)$ is below (resp., above) $dest(m')$. A message m is *righter* (resp., *lefter*) than m', if $dest(m)$ is to the right (resp., to the left) of $dest(m')$. We use $d(m)$ to denote $d(dest(m), BS)$, and $m \preceq m'$ if $dest(m)$ is nearer to the source than $dest(m')$, that is, if $d(m) \leq d(m')$. We suppose in what follows that the messages are ordered by non increasing distance of their destination nodes, and we note $\mathcal{M} = \{m_1, \cdots, m_M\}$ where $m_i \succeq m_j$ for any $i \leq j \leq M$, so $d(m_1) \geq d(m_2) \geq \cdots \geq d(m_M)$.

$S \odot S'$ denotes the sequence obtained by concatenation of two sequences S and S'.

2.2 Lower bound

Consider a model whitout interferences, i.e., any node can receive and transmit simultaneously, but where the source can only send one message per step. Whatever be the broascasting scheme, a message m sent at step $t \geq 1$ will be received at step $t' \geq d(m)+t-1$. A broadcasting scheme is said *greedy* if, given an ordered sequence $\mathcal{S}$ of the messages, the source sends one message per step, in the ordering $\mathcal{S}$, and each message follows a shortest path toward its destination node. Note that, in the model without interferences, if the messages follow shortest paths, a vertex will never receive more than one message per step.

Definition 1. $LB = \max_{i \leq M} d(m_i) + i - 1$.

Lemma 1. *In the model without interferences, when the source emits at most one message per step, a greedy algorithm following the ordered sequence of messages $(m_1, m_2, \cdots, m_M)$ is optimal, with makespan LB.*

Proof. Clearly, sending the messages in the ordering of the sequence $(m_1, m_2, \cdots, m_M)$ along shortest paths will achieve such a makespan. Now, let us consider an optimal schedule of the messages $(s_1^*, \cdots, s_M^*)$ different from $(m_1, m_2, \cdots, m_M)$ and let $i \geq 1$ be the smallest integer such that $s_i^* \neq m_i = s_j^*$ $(j > i)$. We prove that sending the messages in the ordering of the sequence $(s_1^*, \cdots, s_{i-1}^*, s_j^*, s_{i+1}^*, \cdots, s_{j-1}^*, s_i^*, s_{j+1}^*, \cdots, s_m^*)$ does not increase the makespan. Indeed, only the i^{th} and j^{th} messages differ and $\max\{d(s_j^*)+i-1, d(s_i^*)+j-1)\} \leq d(s_j^*)+j-1)$ because $d(v_i, BS) = d(s_j^*) \geq d(s_i^*)$ and $j > i$. By iterating this process, we get that the ordering of the sequence $(m_1, m_2, \cdots, m_M)$ is also optimal. □

Corollary 1. *In the model with interferences, no algorithms can achieve a makespan less than LB.*

3 Personalized Broadcasting Algorithms

3.1 Horizontal-Vertical broadcasting

First, we present a very simple broadcasting scheme that we prove to be sufficient to obtain a good approximation of the optimal makespan.

Given a message whose destination node v has coordinates (x, y), the message is sent *horizontally* to v if it follows the shortest path from BS to v passing through $(x, 0)$. The message is sent *vertically* if it follows the shortest path from BS to v passing through $(0, y)$.

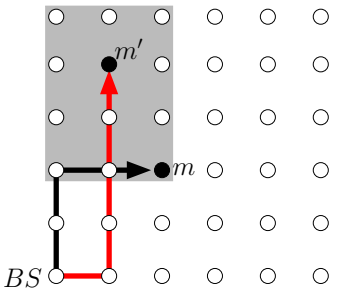
Definition 2. *A Horizontal-Vertical broadcasting scheme, or HV-scheme, takes an ordering $\mathcal{S}$ of $\mathcal{M}$ as an input and proceeds as follows. A direction, horizontal or vertical, is chosen for the first message. Then, the source sends one message every step in the ordering $\mathcal{S}$ and alternating horizontal and vertical messages.*

Let us do some easy remarks about any HV-scheme. Consider two distinct messages sent by the source x time-slots apart. Since these messages follow shortest paths, while the first message has not reached its destination, both messages are separated by a distance at least x. Hence,

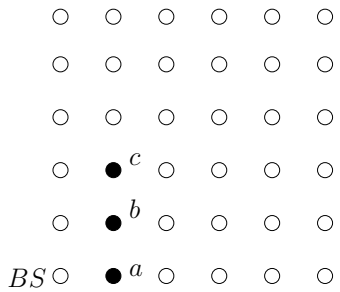
Claim 1 *In a HV-scheme, only consecutive messages may interfer.*

Let us characterize forbidden and acceptable configurations in HV-scheme. Assume that two messages are sent consecutively. It is possible to guess the respective positions of their destination nodes by knowing whether both messages interfer or not. In Figure 1(a), nodes in the grey part are the nodes that are higher and lefter than y. Figure 1(a) illustrates the following Claim.

Claim 2 *Let m, m' be 2 messages sent consecutively by a HV-scheme, with m sent vertically and m' sent horizontally. Messages m and m' interfer if and only if their destinations are distinct and m' is higher and lefter than m.*



(a) Configuration when consecutive messages interfer.



(b) Configuration when the trivial lower bound cannot be achieved.

Fig. 1. Two particular configurations

Input: $\mathcal{M} = \{m_1, \cdots, m_M\}$, the set of messages ordered in non increasing distance order
Output: $(s_1, \cdots, s_M)$ an ordered sequence of $\mathcal{M}$ satisfying (i) and (ii)
begin
  Case $M = 0$ **return** $\emptyset$
  Case $M = 1$ **return** (m_1)
  Case $M \geq 2$
    Let $\mathcal{O} \odot p = TwoApprox(\{m_1, \cdots, m_{M-2}\})$
    Let q be the lowest message in $\{m_{M-1}, m_M\}$ and let r be the other one
    if p is higher than q **return** $\mathcal{O} \odot (p, q, r)$
    else return $\mathcal{O} \odot (m_{M-1}, p, m_M)$
end.

Fig. 2. Algorithm $TwoApprox$

Before continuing, let us remark that there exist configurations for which no gathering protocol can achieve better makespan than $LB + 1$. Figure 1(b) represents such a configuration. Indeed, in Figure 1(b), the three destinations a, b and c have coordinates $(1, 0), (1, 1)$ and $(1, 2)$, and $LB = 3$. However, to achive such a makespan, the first message must be sent to c (because c is at distance 3 from g) and the second message must be sent to b (because the message start after the first step and must go at distance 2). To avoid collision, the only possibility is to send the first message vertically, and the second one horizontally. But then, the last message cannot reach a before step 4.

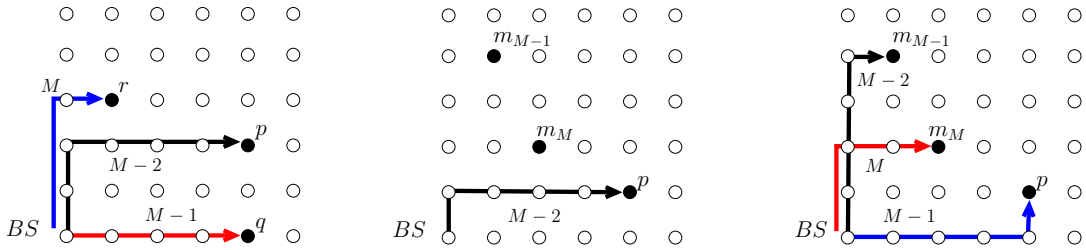
3.2 +2 approximation

Recall that $(m_1, \cdots, m_M)$ denotes the ordered sequence of the messages in the non increasing ordering of the distance to their destinations. In this section, we give the Algorithm $TwoApprox$, depicted in Figure 2, that computes an ordered sequence $\mathcal{S} = (s_1, \cdots, s_m)$ of the messages satisfying the two following properties:

(i) HV-scheme($\mathcal{S}$) broadcasts the messages without collisions, sending the last message vertically, and
(ii) $s_i \in \{m_{i-2}, m_{i-1}, m_i, m_{i+1}, m_{i+2}\}$ for any $i \leq M$, and $s_M \in \{m_{M-1}, m_M\}$

Theorem 1. *Algorithm TwoApprox computes an ordering $\mathcal{S}$ of the messages satisfying properties (i) and (ii) and so HV-scheme($\mathcal{S}$) achieves makespan at most $LB + 2$.*

Proof. To prove the correctness of Algorithm $TwoApprox$, we proceed by induction on M. If $M \leq 2$, the result holds obviously. Let us assume that the ordering of the sequence computed by $TwoApprox(\{m_1, \cdots, m_{M-2}\})$



(a) $\{q, r\} = \{m_{M-1}, m_M\}$ and p strictly higher than q.

(b) Both m_{M-1} and m_M are higher than p.

(c) New scheduling in Case of Figure 3(b)

Fig. 3. $M-2$ messages have been scheduled, finishing with the one to $p \in \{m_{M-2}, m_{M-3}\}$. When the next two messages must be scheduled, two cases occur according to the position of m_{M-1} and m_M relatively to p. In the figures, an arrow with label i represents the route of the i^{th} message.

satisfies properties (i) and (ii). Let p be the last message of this sequence. By the induction hypothesis, $p \in \{m_{M-3}, m_{M-2}\}$ is sent vertically. Let t be the message before p in this sequence. By Claim 2, p must be higher or lefter than t. The sequence is denoted by $\mathcal{O} \odot p = \mathcal{O}' \odot (t, p)$.

Let q be the lowest message in $\{m_{M-1}, m_M\}$ and let r be the other one. We consider two cases depending on the positions of p, q and r.

a) **Case p is higher than q.** It is sufficient to send q horizontally at step $M-1$, and r vertically at step M. This case is depicted in Figure 3(a). Indeed, by Claim 1 only p and q, or q and r may interfer. By Claim 2, there are no interferences. It is easy to check that $\mathcal{O} \odot (p, q, r)$ satisfies (i) and (ii).

b) **Case q and r are higher than p.** Since $q, r \preceq p$ (i.e. q, r closer to BS than p.), they are higher and lefter than p. This case is depicted in Figure 3(b). In this case, instead of sending p at step $M-2$, the source sends m_{M-1} vertically at step $M-2$, then p horizontally at step $M-1$, and then m_M vertically at step M. The transformation is depicted in Figure 3(c). Clearly, $\mathcal{O} \odot (m_{M-1}, p, m_M)$ satisfies (i) and (ii). By Claim 1 only t and m_{M-1}, or m_{M-1} and p, or p and m_M may interfer. Since m_{M-1} is higher and lefter than p that is higher or lefter than t, by Claim 2, m_{M-1} interfers neither with t nor with p. Similarly, m_M is higher and lefter than p and these messages do not interfer.

□

3.3 +1 approximation

In this section, we give the Algorithm *OneApprox*, depicted in Figure 3.3, that computes an ordered sequence $\mathcal{S} = (s_1, \cdots, s_m)$ of the messages satisfying the following properties:

(i) HV-scheme($\mathcal{S}$) broadcasts the messages without collisions, sending the last message vertically, and
(iii) $s_i \in \{m_{i-1}, m_i, m_{i+1}\}$ for any $i \leq M$ (in particular, either $s_M = m_M$, or $s_M = m_{M-1}$ and $s_{M-1} = m_M$).

We need some definitions. An ordered sequence $\mathcal{S} = (s_1, \cdots, s_M)$ of $\mathcal{M}$ satisfying (i) and (iii) is said *valid*. Sequence $\mathcal{S}$ is said *i-almost valid* if it satisfies all the desired properties, but s_i may interfer with s_{i+1}, for a unique $i \leq M$. Clearly, for any valid sequence $\mathcal{S}$, HV-scheme($\mathcal{S}$) achieves makespan at most $LB + 1$.

In the following, we prove that Algorithm *OneApprox* computes a valid ordered sequence of $\mathcal{M}$. The proof is by induction on M. Roughly, starting from a valid ordered sequence of $\{m_1, \cdots, m_{M-2}\}$, the algorithm includes m_{M-1} and m_M in this ordered sequence. Then, either the obtained sequence $\mathcal{S}$ is valid, or it is $(M-4)$-almost valid and satisfies some other properties (see def. 3). In the latter case, Subprocedure $MakeValid$ is recursively applied to $\mathcal{S}$ which we prove to result in a valid sequence.

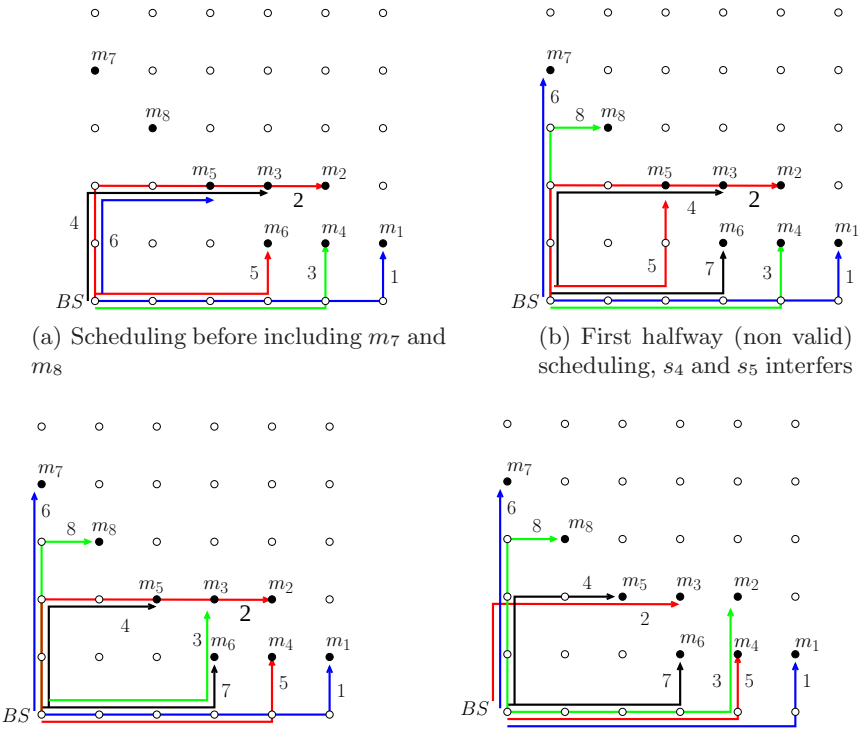
Before proving Theorem 2, we detail the execution of Algorithm *OneApprox* on the example depicted in Figure 6. In this example, BS must send 8 messages $\{m_1, \cdots, m_8\}$ to distinct vertices in a 6×6-grid.

Input: $\mathcal{M} = \{m_1, \cdots, m_M\}$, the set of messages ordered in non increasing distance order
Output: $(s_1, \cdots, s_M)$ an ordered sequence of $\mathcal{M}$ such that $m_i \in \{s_{i-1}, s_i, s_{i+1}\}$ for any $i \leq M$
begin
Case $M = 0$ **return** $\emptyset$
Case $M = 1$ **return** (m_1)
Case $M \geq 2$
Let $\mathcal{O} \odot p = OneApprox(\{m_1, \cdots, m_{M-2}\})$
Let q be the lowest message in $\{m_{M-1}, m_M\}$ and let r be the other one
if p is higher than q **return** $\mathcal{O} \odot (p, q, r)$
else if $p = m_{M-2}$ **return** $\mathcal{O} \odot (m_{M-1}, p, m_M)$
else
/* This last case may occur only if $M > 3$ */
Let $(s_1, \cdots, s_{M-4}) \odot (m_{M-2}, m_{M-3}) = OneApprox(\{m_1, \cdots, m_{M-2}\})$
return $MakeValid((s_1, \cdots, s_{M-4}) \odot (m_{M-3}, m_{M-1}, m_{M-2}, m_M), 2)$
end.

Fig. 4. Algorithm *OneApprox*

Input: A $(j-1)$-good (see def. 3) sequence $\mathcal{O} = (s_1, \cdots, s_M)$ of a set of messages $\{m_1, \cdots, m_M\}$, and an integer j, $1 < j \leq \lfloor M/2 \rfloor$.
Output: A valid sequence of $\mathcal{M}$
if s_{M-2j} and s_{M-2j+1} do not interfer
/* In particular, this case occurs if $M - 2j = 0$ */
return $\mathcal{O}$
else if $s_{M-2j} = m_{M-2j}$
/* In particular, this case occurs if $M - 2j = 1$ */
return $(s_1, \cdots, s_{M-2j-2}) \odot (s_{M-2j-1}, s_{M-2j+1}, s_{M-2j}, s_{M-2j+2}) \odot (s_{M-2j+3}, \cdots, s_M)$
else return
/* This last case may occur only if $M - 2j \geq 2$ */
/* Note that, in this case, $s_{M-2j} = m_{M-2j-1}$ and $s_{M-2j-1} = m_{M-2j}$ */
$MakeValid((s_1, \cdots, s_{M-2j-2}) \odot (s_{M-2j}, s_{M-2j+1}, s_{M-2j-1}, s_{M-2j+2}) \odot (s_{M-2j+3}, \cdots, s_M), j+1)$
end.

Fig. 5. MakeValid



(a) Scheduling before including m_7 and m_8

(b) First halfway (non valid) scheduling, s_4 and s_5 interfers

(c) Second halfway (non valid) scheduling, s_2 and s_3 interfers

(d) Final valid scheduling

Fig. 6. Recursive modifications of the scheduling

Algorithm *OneApprox* first computes a valid ordered sequence $(s_1, \cdots, s_6) = (m_1, m_2, m_4, m_3, m_6, m_5)$ of first 6 messages. This scheduling is depicted in Figure 6(a). Then, the positions of m_7, m_8 and $s_6 = m_5$ are compared. In the example, m_8 is the lowest message among m_7 and m_8, and it is higher than s_6. Moreover, $s_6 \neq m_6$. Hence, Algorithm *OneApprox* applies Subprocedure $MakeValid$ to integer $j = 2$ together with the ordered sequence $(s_1, \cdots, s_4) \odot (m_5, m_7, m_6, m_8) = (m_1, m_2, m_4, m_3, m_5, m_7, m_6, m_8)$. The scheduling corresponding to this sequence is depicted in Figure 6(b). It is easy to check that this sequence is 4-almost valid, i.e., it is valid except for the interference between m_3 and m_5. Note that the integer variable j in the input of Subprocedure $MakeValid$ simply indicates that the interference may appear between the $M - 2j^{th}$ and the $M - 2j + 1^{th}$ messages of the given sequence. The goal of Subprocedure $MakeValid$ is to locally modify the sequence in order to remove interference between the $M - 2j^{th}$ and the $M - 2j + 1^{th}$ messages. However, a new interference may appear between the $M - 2(j + 1)^{th}$ and the $M - 2(j + 1) + 1^{th}$ messages of the obtained sequence, in which case Subprocedure $MakeValid$ is recall recursively. Such a situation occurs in the example. Indeed, in the sequence $(m_1, m_2, m_4, m_3, m_5, m_7, m_6, m_8)$, m_3 and m_5 interfers and the fourth message of this sequence is not m_4. Then, Subprocedure $MakeValid$ is applied to the sequence $(m_1, m_2) \odot (m_3, m_5, m_4, m_7) \odot (m_6, m_8)$ with $j = 3$. This sequence is depicted in Figure 6(c) and is $M - 2j$-almost valid since m_2 and m_3 interfer. Note that the second message of this sequence interfers and that this message is actually m_2. Therefore, the next call to Subprocedure $MakeValid$ only exchanges m_2 and m_3 (Case 2 of the subprocedure) and returns the ordered sequence $(m_1, m_3, m_2, m_5, m_4, m_7, m_6, m_8)$. The scheduling corresponding to this sequence is depicted in Figure 6(d) and it is easy to check that it is valid.

We now prove the correctness of Algorithm *OneApprox* and Subprocedure $MakeValid$.

Theorem 2. *Algorithm OneApprox computes an ordering $\mathcal{S}$ of the messages satisfying properties (i) and (iii) and so HV-scheme($\mathcal{S}$) achieves makespan at most $LB + 1$.*

Proof. We prove that Algorithm *OneApprox* computes an ordered sequence of messages satisfying the properties (i) and (iii). We proceed by induction on M. If $M \leq 2$, the result holds obviously. Let us assume that the sequence $OneApprox(\{m_1, \cdots, m_{M-2}\})$ satisfies (i) and (iii). Let p be the last message of this sequence. By the induction hypothesis, $p \in \{m_{M-3}, m_{M-2}\}$ is sent vertically. The sequence is denoted by $(s_1, \cdots, s_{M-2})$.

For any $i < M-2$, $\mathcal{O}_i$ denotes $(s_1, \cdots, s_{M-2-i})$, i.e., $(s_1, \cdots, s_{M-2}) = \mathcal{O}_i \odot (s_{i+1}, \cdots, s_{M-2})$. Recall that by induction hypothesis, $p = s_{M-2} \in \{m_{M-3}, m_{M-2}\}$ and has been sent vertically. Let q be the lowest message in $\{m_{M-1}, m_M\}$ and let r be the other one. We consider the 3 cases of Algorithm *OneApprox* (when $M \geq 2$).

a) **Case p is higher than q.** We proceed like in case (a) of the proof of Theorem 1. In this way it is easy to check that $\mathcal{O}_1 \odot (p, q, r)$ is valid.

b) **Case q and r are higher than p, and $p = m_{M-2}$.** We proceed like in case (b) of the proof of Theorem 1. The transformation is depicted in Figure 3(c). The sequence $\mathcal{O}_1 \odot (m_{M-1}, p, m_M)$ is valid.

c) **Case q and r are higher than p, and $p \neq m_{M-2}$** (i.e., $s_{M-3} = m_{M-2}$ and $p = s_{M-2} = m_{M-3}$). Moreover, p is sent vertically. This case is depicted in Figure 3.3 (with $M = 8$).
In this case, Algorithm *OneApprox* returns the result of $MakeValid(\mathcal{O}_{M-4} \odot (m_{M-3}, m_{M-1}, m_{M-2}, m_M), 2)$.
We now prove that the computed sequence is valid.
By Claim 2 and because $s_{M-3} \preceq s_{M-2} = p$, s_{M-3} is lower than $s_{M-2} = p$. Indeed, these messages do not interfer in the ordered sequence $(s_1, \cdots, s_{M-2})$ computed by Algorithm *OneApprox*. Then, it is possible to send messages $m_{M-3}, m_{M-1}, m_{M-2}$ and m_M, alternatively horizontal and vertical (starting horizontally) without any interference between these four messages. Therefore, the scheduling $\mathcal{O}_{M-4} \odot (m_{M-3}, m_{M-1}, m_{M-2}, m_M)$ is either valid or $(M-4)$-almost valid.
Actually, the scheduling $\mathcal{O}_{M-4} \odot (m_{M-3}, m_{M-1}, m_{M-2}, m_M)$ has some more useful properties. More precisely, it is 1-good, where a i-good sequence ($i \geq 2$) is defined as follows:

Definition 3. *Let $i \in \{2, \cdots, \lfloor M/2 \rfloor\}$. An ordered sequence $\mathcal{S} = (s_1, \cdots, s_M)$ is $(i-1)$-good if it is $(M-2i)$-almost valid, and $s_{M-2i+1} = m_{M-2i+1}$, $s_{M-2i+2} = m_{M-2i+3}$ and $s_M = m_M$.*

Let us do the following easy remarks.
1. In a $(i-1)$-good sequence $\mathcal{S}$, if s_{M-2i} does not interfer with s_{M-2i+1} then $\mathcal{S}$ is valid.
2. In particular, if $2i = M$, a $(i-1)$-good sequence is valid.

Let i, $1 < i \leq \lfloor M/2 \rfloor$ and let $\mathcal{S}$ be a $(i-1)$-good sequence. By reverse induction on i, we prove that $MakeValid(\mathcal{S}, i)$ eventually computes a valid sequence. More precisely, we prove that $MakeValid(\mathcal{S}, i)$ directly returns a valid sequence (first two cases of Subprocedure $MakeValid$), or it returns the result of $MakeValid(\mathcal{S}', i+1)$ where $\mathcal{S}'$ is an i-good sequence, and so the result holds by induction.
By definition of an $(i-1)$-good sequence,
$\mathcal{S} = (s_1, \cdots, s_{M-2i-2}) \odot (s_{M-2i-1}, s_{M-2i}, s_{M-2i+1}, s_{M-2i+2}) \odot (s_{M-2i+3}, \cdots, s_M)$
$\quad = (s_1, \cdots, s_{M-2i-2}) \odot (s_{M-2i-1}, s_{M-2i}, m_{M-2i+1}, m_{M-2i+3}) \odot (s_{M-2i+3}, \cdots, s_M)$

First we prove the intialisation of the induction, i.e., we consider the case when $2i \in \{M-1, M\}$.
If $2i = M$, then s_{M-2i} does not interfer with s_{M-2i+1} (because s_{M-2i} is not defined). $MakeValid(\mathcal{S}, i)$ returns $\mathcal{S}$ which is valid by Remark 2.
If $2i = M-1$ and s_1 and s_2 do not interfer, $MakeValid(\mathcal{S}, i)$ returns $\mathcal{S}$ that is valid by Remark 1. Otherwise, if $2i = M-1$ and s_1 and s_2 interfer, then $\mathcal{S} = (s_1, m_2, m_4) \odot (s_4, \cdots, s_{2i+1})$. This implies that $s_1 = m_1$. Moreover, by parity and because s_{2i+1} is sent vertically, m_2 is sent horizontally. By Claim 2 and because $m_1 \preceq m_2 \preceq m_4$, m_1 must be lower than m_2 that is lower than m_4. In this case, $MakeValid(\mathcal{S}, i)$ returns $(m_2, m_1, m_4) \odot (s_4, \cdots, s_{2i+1})$. By Claim 1, the only possible interferences in the resulting scheduling may occur between m_1 and m_2, or m_2 and m_4. The respective positions of the destination nodes of m_1, m_2, m_4 imply that this sequence is valid.

Now, let us consider the case $2i < M-1$. For purpose of induction, let us assume that, for any $j > i$, if $\mathcal{S}$ is $j-1$-good, $MakeValid(\mathcal{S}, j)$ returns a valid sequence. We prove that, for any $(i-1)$-good sequence $\mathcal{S}$, $MakeValid(\mathcal{S}, i)$ returns a valid sequence.

c.1 s_{M-2i} and s_{M-2i+1} do not interfer.
In this case, $MakeValid(\mathcal{S}, i)$ returns $\mathcal{S}$ that is valid by Remark 1.

Let us then consider the case when s_{M-2i} and s_{M-2i+1} interfer.
We first do general remarks on the relative positions of some destination's nodes. By parity, in $\mathcal{S}$, s_{M-2i} is sent vertically and s_{M-2i+1} horizontally. By Claim 2, s_{M-2i+1} is higher and lefter than s_{M-2i}. Moreover, since s_{M-2i+1} and s_{M-2i+2} do not interfer, then s_{M-2i+2} must be higher or lefter than s_{M-2i+1}. Similarly, s_{M-2i-1} must be either lower or righter than s_{M-2i}.
There are two cases to be considered according to the value of s_{M-2i}. Recall that, because $s_{M-2i+1} = m_{M-2i+1}$, $s_{M-2i} \in \{m_{M-2i}, m_{M-2i-1}\}$.

c.2 s_{M-2i} and s_{M-2i+1} do interfer and $s_{M-2i} = m_{M-2i}$. In this case, $MakeValid(\mathcal{S}, i)$ returns the sequence $(s_1, \cdots, s_{M-2i-2}) \odot (s_{M-2i-1}, s_{M-2i+1}, s_{M-2i}, s_{M-2i+2}) \odot (s_{M-2i+3}, \cdots, s_M)$.
Because of their respective positions, no interferences are created between messages s_{M-2i-1}, s_{M-2i+1}, s_{M-2i} and s_{M-2i+2}, when sending them alternatively horizontal and vertical (starting horizontally). Moreover, only $s_{M-2i+1} = m_{M-2i+1}$ and $s_{M-2i} = m_{M-2i}$ have been switched. Therefore, it is easy to check that it is valid.

c.3 s_{M-2i} and s_{M-2i+1} do interfer and $s_{M-2i} = m_{M-2i-1}$. Note that this implies $s_{M-2i-1} = m_{M-2i}$.
In this case, $MakeValid(\mathcal{S}, i)$ returns $MakeValid(\mathcal{S}', i+1)$ where
$\mathcal{S}' = (s_1, \cdots, s_{M-2i-2}) \odot (s_{M-2i}, s_{M-2i+1}, s_{M-2i-1}, s_{M-2i+2}) \odot (s_{M-2i+3}, \cdots, s_M)$
$= (s_1, \cdots, s_{M-2i-2}) \odot (m_{M-2i-1}, m_{M-2i+1}, m_{M-2i}, s_{M-2i+2}) \odot (s_{M-2i+3}, \cdots, s_M)$.
Because of their respective positions, no interferences are created between messages s_{M-2i}, s_{M-2i+1}, s_{M-2i-1} and s_{M-2i+2}, when sending them alternatively horizontal and vertical (starting horizontally). Hence, the only possible interference in the sequence $\mathcal{S}'$ is between s_{M-2i-2} and to s_{M-2i}: It is easy to check that $\mathcal{S}'$ is i-good. Thus, the result is valid by the induction assumption.

□

We prove that Algorithm *OneApprox* performs in linear time, with respect to the number of messages.

Theorem 3. *The time complexity of Algorithm OneApprox is $O(M)$.*

Proof. We note $\chi(i)$ the time-complexity of $OneApprox(\{m_1, \cdots, m_i\})$. We prove by induction on i that $\chi(i) \leq 2i \cdot O(1)$. Let $S = (s_1, \cdots, s_M)$ be the ordered sequence computed by $OneApprox(\mathcal{M})$. The pivot s_{M-2P} of this sequence is the message such that $s_{M-2P} = m_{M-2P}$ and minimizing P. More precisely, we prove that $\chi(M) \leq O(1) \cdot (2(M-2P)+P)$. If $M \leq 1$, the result is trivial. Let us assume $M \geq 2$. By induction, the computation of $OneApprox(\{m_1, \cdots, m_{M-2}\})$ takes time at most $O(1) \cdot (2(M-2-2P')+P')$ where P' is the pivot of the obtained sequence. There are three possible cases corresponding to the three cases the algorithm (when $M \geq 2$).

a. Clearly, in this case, $\chi(M) = \chi(M-2) + O(1)$. Moreover, either $P = P' + 1$ (if $r = m_{M-1}$) or $P = 0$ (if $r = m_M$). In both cases, we get $\chi(M) \leq O(1) \cdot (2(M-2P)+P)$.
b. $\chi(M) = \chi(M-2) + O(1)$ and $P = P' + 1$, thus $\chi(M) \leq O(1) \cdot (2(M-2P)+P)$.
c. Let $\chi'(M)$ be the complexity of $Makevalid((s_1, \cdots, s_M), 2)$. In this case, $\chi(M) = \chi(M-2) + \chi'(M)$ and $P = 0$ (because in the computed sequence, $s_M = m_M$). Finally, when executed $Makevalid((s_1, \cdots, s_M), 2)$, the same subprocedure $MakeValid$ is recursivelly executed until s_{M-2j} and s_{M-2j+1} do not interfer, or $s_{M-2j} = m_{M-2j}$. Therefore, it is executed at most P' times and each execution takes $O(1)$. Hence, $\chi(M) \leq O(1) \cdot 2(M-2P') + O(1) \cdot P' \leq 2M$.

□

4 Distributed Algorithm

We present a distributed algorithm for the gathering in a grid. This algorithm is based on the Algorithm *TwoApprox*, for personalized broadcasting presented in section 3.2.

4.1 Distributed Model

The network is assumed to be synchronous. Each node has only a local view of the network. However, it has access to the following global information: its position (x, y) in the grid, the position of BS (for sake of simplicity, we assume that BS has coordinates $(0, 0)$), and the size $N \times N$ of the grid (an upper bound on N is sufficient). Finally, any node v has $m(v) \geq 0$ messages that it must send to BS. At every step, a node can send or (exclusive) receive a control message, or *signaling*, of size $O(\log N)$ to (from) one of its neighbours. In the following, for any $i \leq 2N$, $Diag(i)$ denotes the set of vertices at distance i from BS. We refer to $Diag(i)$ as the *diagonal* i. The *central node* $c(2a)$ (resp., $c(2a+1)$) of $Diag(2a)$ (resp., $Diag(2a+1)$) is the node with coordinates (a, a) (resp., $(a+1, a)$). Finally, let $AntiDiag$ be the set that consists of the vertices $c(i)$ for all $i \leq 2N$. The algorithm consists of four phases that we describe now.

4.2 Basic Description of Distributed Algorithm

Our algorithm aims at giving to any message m its position in the ordering $\mathcal{S}$ computed by Algorithm $TwoApprox$ (in terms of personalized broadcasting) and the makespan. This is performed in $Y = O(N^2)$ steps (Y will be specified below) using $O(N^2)$ signalings. Then, with this information, any message can compute its starting time, given that the first message will be sent at step $Y + 1$.

Let us give a rough description of the four phases of the distributed algorithm. First two phases consist in giving to any message m its position in the non increasing order of their distance to BS such that nodes in the same diagonal are ordered up to down (the ordering of messages hosted at a same node is arbitrary). Moreover, each message m_{2a+1} with $a \geq 0$, resp., m_{2a+2}, (actually, the node hosting this message) will learn the position(s) of messages $m_{2a+2}, m_{2a+3}, m_{2a+4}$, resp., $m_{2a+1}, m_{2a+3}, m_{2a+4}$. Then the third phase starts. With the information previously learnt, according to Algorithm $TwoApprox$, message m_1 can decide the ordering in $\mathcal{S}$ of the first three messages: s_1, s_2, s_3. Two of these three positions are occupied by m_1 and m_2. The remaining place is occupied by m_3 or m_4 (This comes from the definition of the $TwoApprox$ algorithm). Then, at some step, the message s_{2a+3} is fixed. With this information, we prove that message m_{2a+3} can extend the ordering to s_{2a+4} and s_{2a+5} using the $TwoAlgo$ algorithm. At the end of this phase, any node knows its position in $\mathcal{S}$ and BS knows the makespan. During the last phase, BS broadcasts the makespan to any node. With this information, each node can compute its starting time for the gathering process.

4.3 Formal Description of Distributed Algorithm

Phase 1. The first phase is divided into two processes that are executed "almost" simultaneously.

- The first one is executed in parallel by all diagonals. For any $i \leq 2N$, it aims at collecting some information in $c(i)$, the central node of $Diag(i)$. When this process ends up at step $i + 5$, $c(i)$ has learnt
  - the number of messages l_i standing in $Diag(i)$ in nodes with greater ordinate than $c(i)$,
  - the number of messages r_i standing in $Diag(i)$ in nodes with smaller ordinate than $c(i)$,
  - the position(s) of the three messages with greatest ordinate in $Diag(i)$.

  Moreover, at the end of the phase, any node v with coordinates (x, y) in $Diag(i)$ has learnt the position of the (at most 3) node(s) of $Diag(i)$ hosting the closest 3 messages that are higher (if $y \geq x$) or lower (if $y \leq x$) than v.
  To do so, two signalings $D1_i$ and $D1'_i$, initiated by nodes $(i, 0)$ and $(0, i)$ respectively, are propagated toward $c(i)$. From $(i, 0)$ (resp., from $(0, i)$), $D1_i$ (resp., $D1'_i$) is transmitted to node $(i, 1)$ and then to $(i-1, 1)$ (resp., to $(1, i)$ and then to $(1, i-1)$), and so on until reaching $c(i)$. To avoid interferences, $D1_i$ and $D1'_i$ are initiated at step 1 by $(i, 0)$ and $(0, i)$ if i is odd. If i is even, $D1_i$ is initiated at step 5 by $(i, 0)$, and $D1'_i$ is initiated at step 6 by $(0, i)$. It is easy to see how information can be aggregated as $D1_i$ and $D1'_i$ go along, in order to obtain the desired information. Moreover, signalings $D1_i$ (resp. $D1'_i$) have size $O(\log N)$ since they contain: the number of messages they met, the position(s) of the first three messages they met, and the position(s) of the last three messages they met.

– At step 7, a signaling $A1$ is initiated in BS and is propagated along $AntiDiag$ towards $(N-1, N-1)$. When $c(i)$ receives $A1$ at step $i+6$, it learns the total number of messages hosting by nodes in $\bigcup_{j<i} Diag(j)$ and the position(s) of the three messages in $\bigcup_{j<i} Diag(j)$ that are further to BS and with greatest ordinate. Then, using the information propagated by messages $D1_i$ and $D1'_i$, $c(i)$ updates message $A1$ and sends it to $c(i+1)$ during the next step.
The signaling $A1$ arrives to $(N-1, N-1)$ at step $2N+5$ which concludes this phase.

Phase 2. The second phase is divided into three successive processes.

– At step $2N+6$, a signaling $A2$ is initiated in $(N-1, N-1)$ and is propagated along $AntiDiag$ towards $(0,0)$. When $c(i)$ receives $A2$ at step $4N+6-i$, it learns the total number of messages M and the position(s) of the three messages in $\bigcup_{j>i} Diag(j)$ that are closest to BS and with smallest ordinate.
Note that after step $4N+6-i$, $c(i)$ knows the interval of the positions occupied by messages in $Diag(i)$, i.e., from $M - \bigcup_{j\leq i} Diag(j) + 1 = \bigcup_{j>i} Diag(j) + 1$ to $\bigcup_{j>i} Diag(j) + l_i + r_i + m(c(i))$.

The goal of next processes is that: any message m knows its position in the non increasing order of their distance to BS, i.e., its position in the ordered sequence $\mathcal{M}$, and any message m_{2a} (resp., m_{2a+1}) knows the position(s) of messages $m_{2a+1}, m_{2a+2}, m_{2a+3}$ (resp., $m_{2a}, m_{2a+2}, m_{2a+3}$).

– At step $4N+6-i+3$ if i is odd and $4N+6-i+5$ if i is even, a signaling $D2_i$ is initiated in $c(i)$ and is propagated toward $(i,0)$. $D2_i$ transmits: the next position (in $\mathcal{M}$) to be attribuated to the messages in $Diag(i)$ with smaller ordinates than $c(i)$, i.e., from $\bigcup_{j>i} Diag(j) + l_i + m(c(i))$ to $\bigcup_{j>i} Diag(j) + l_i + r_i + m(c(i))$ (in such a way that any message lower than $c(i)$ in $Diag(i)$ leanrs its number in the ordering when it meets the signaling $D2_i$), the position(s) of the last three messages met by this signaling, and the position(s) of the three messages in $\bigcup_{j<i} Diag(j)$ furthest to BS and with greatest ordinates.
– At step $4N+6-i+2$ if i is odd and $4N+6-i+6$ if i is even, a signaling $D2'_i$ is initiated in $c(i)$ and is propagated toward $(0,i)$. $D2'_i$ transmits: the next position (in the ordering) to be attribuated to the messages in $Diag(i)$ with greater ordinates than $c(i)$, i.e., from $\bigcup_{j>i} Diag(j)$ to $\bigcup_{j>i} Diag(j) + l_i$ (in such a way that any message higher than $c(i)$ in $Diag(i)$ leanrs its number in the ordering when it meets the signaling $D2'_i$), the position(s) of the last three messages met by this signaling, and the position(s) of the three messages in $\bigcup_{j<i} Diag(j)$ furthest to BS and with greatest ordinates.
This phase ends at slot $4N+12$.

Phase 3. During this phase, any message learns its position in the final ordering $\mathcal{S}$.

We define the start of this phase at slot $4N+13$ after finishing phase 2.

At the beginning of this phase, message m_{2a+1} ($a \geq 0$) knows its position in the ordered sequence $\mathcal{M}$ and the position(s) of m_{2a+2}, m_{2a+3}, and m_{2a+4}.

The procedure starts as follows. Node m_1 knows m_2, m_3, m_4. Using $TwoApprox$ algorithm with input (m_1, m_2, m_3, m_4), it computes the ordering of the first three positions of $\mathcal{S}$. According to the algorithm the possible configurations for the first three messages in $\mathcal{S}$ are (m_1, m_2, m_3), (m_1, m_3, m_2), (m_1, m_2, m_4), (m_2, m_1, m_3), (m_2, m_3, m_1), (m_2, m_1, m_4). Note that, although the algorithm returns also a message for the fourth position, it is not definitive because it could be modified when the next pair of messages (m_5, m_6) is included. The first message s_1 is decided arbitrarily to be vertical.

Then, m_1 computes the current makespan, i.e., $\max_{j\in\{1,2,3\}} d(BS, s_j) + m_j - 1$ and propagates the information to m_2 and m_3. That is, m_1 sends them the ordering of the first three messages (s_1, s_2, s_3) of $\mathcal{S}$ (again, $\{s_1, s_2, s_3\} \subset \{m_1, \cdots, m_4\}$) and the current makespan. The corresponding signaling is sent at step $4N+13$ to m_3 and at step $4N+15$ to m_2. The signaling reachs m_3 at step $4N+12+t$ where t is the distance between m_1 and m_3.

The process continues iteratively until m_{2a+3} receives a signaling from m_{2a+1} at step $4N+12+t$, for $t = \sum_{0\leq k\leq p} dist(m_{2k+1}, m_{2k+3})$. This signaling contains the positions of messages $s_{2a+1}, s_{2a+2}, s_{2a+3}$, and the current makespan, i.e., the makespan restricted to messages s_1 to s_{2a+3}. At this step, m_{2a+3} must decide which messages will occupy positions s_{2a+4} and s_{2a+5} in $\mathcal{S}$. This decision is taken according to Algorithm

$TwoApprox$. Note that, Algorithm $TwoApprox$ requires as input the next pair of messages m_{2a+5}, m_{2a+6} and the message $m^* \in \{m_1, \cdots, m_{2a+4}\}$ whose position in $\mathcal{S}$ has not been decided yet. By property of Algorithm $TwoApprox$, $m^* \in \{m_{2a+3}, m_{2a+4}\}$

Thus, m_{2a+3} is able to decide which messages will occupy positions s_{2a+4} and s_{2a+5} in $\mathcal{S}$, and then it can update the current makespan. Finally, at step $4N + 12 + t + 1$ (resp., at step $4N + 12 + t + 3$), message m_{2a+3} sends a signaling to m_{2a+5} (resp., to m_{2a+4}). This signaling contains the current makespan, s_{2a+3}, s_{2a+4} and s_{2a+5}. The signaling is received by m_{2a+5} at step $4N + 12 + t + t'$ where t' is the distance between m_{2a+3} and m_{2a+5}. The end of this phase is upper bounded by step $4N + 12 + 2N^2$.

Phase 4. At the end of previous phase, BS learns the makespan of a HV-scheme realizing the computed ordering and starts broadcasting it to any node at step $4N + 13 + 2N^2$.

This is done thanks to a signaling through $AntiDiag$, and signalings from $c(i)$ to $(i, 0)$ and $(0, i)$ $(i \leq 2N)$ in a similar way as Phase 2. This process ends at step $6N + 19 + 2N^2$.

Defining $Y = 6N + 19 + 2N^2$, each node knows the step when it has to send the message given that the starting step is $Y + 1$. Moreover, the message s_j is sent horinzontally or vertically according to the parity of j.

5 Conclusion and Further Works

In this paper, we have presented algorithms for the minimum makespan personalized broadcasting in grid networks. In these settings, the problem is strictly equivalent to the data gathering problem. One can note that our network model assumes that an optimal MAC layer is available. It would be interesting to investigate on the behavior of the problems under weaker assumptions. Another direction to investigate is the online version of the problems. It is worth pointing out that, in this case, personalized broadcasting and gathering are no longer equivalent.

References

1. Klasing, R., Lotker, Z., Navarra, A., Pérennes, S.: From balls and bins to points and vertices. In: Proceedings of the 16th Annual International Symposium on Algorithms and Computation (ISAAC 2005). Volume 3827 of Lecture Notes in Computer Science., Springer Verlag (December 2005) 757–766
2. Bonifaci, V., Klasing, R., Korteweg, P., Stougie, L., Marchetti-Spaccamela, A.: Data Gathering in Wireless Networks. In: Graphs and Algorithms in Communication Networks. Springer Monograph Springer-Verlag (2009)
3. Gargano, L.: Time optimal gathering in sensor networks. In Prencipe, G., Zaks, S., eds.: Structural Information and Communication Complexity, 14th International Colloquium, SIROCCO 2007, Castiglioncello, Italy, June 5-8, 2007, Proceedings. Volume 4474 of Lecture Notes in Computer Science., Springer (2007) 7–10
4. Florens, C., Franceschetti, M., McEliece, R.: Lower bounds on data collection time in sensory networks. Selected Areas in Communications, IEEE Journal on **22**(6) (2004) 1110–1120
5. Revah, Y., Segal, M.: Improved bounds for data-gathering time in sensor networks. Computer Communications **31**(17) (November 2008) 4026–4034
6. Gargano, L., Rescigno, A.A.: Optimally fast data gathering in sensor networks. In Kralovic, R., Urzyczyn, P., eds.: MFCS. Volume 4162 of Lecture Notes in Computer Science., Springer (2006) 399–411
7. Gargano, L., Rescigno, A.A.: Collision-free path coloring with application to minimum-delay gathering in sensor networks. Discrete Applied Mathematics **In Press**
8. Revah, Y., Segal, M.: Improved algorithms for data-gathering time in sensor networks ii: Ring, tree and grid topologies. In: Networking and Services, 2007. ICNS. Third International Conference on. (2007) 46
9. Busch, C., Herlihy, M., Wattenhofer, R.: Hard-potato routing. In: Proceedings of the thirty-second annual ACM symposium on Theory of computing, Portland, Oregon, United States, ACM (2000) 278–285
10. Mansour, Y., Patt-Shamir, B.: Many-to-one packet routing on grids. In: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing, Las Vegas, Nevada, United States, ACM (1995) 258–267
11. Bermond, J.C., Galtier, J., Klasing, R., Morales, N., Pérennes, S.: Hardness and approximation of gathering in static radio networks. Parallel Processing Letters **16**(2) (2006) 165–183
12. Bonifaci, V., Korteweg, P., Marchetti-Spaccamela, A., Stougie, L.: An approximation algorithm for the wireless gathering problem. Operations Research Letters **36**(5) (2008) 605 – 608

13. Bermond, J.C., Peters, J.: Efficient gathering in radio grids with interference. In: Septièmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel'05), Presqu'île de Giens (May 2005) 103–106
14. Gomes, C., Pérennes, S., Reyes, P., Rivano, H.: Bandwidth allocation in radio grid networks. In: 10èmes Rencontres Francophones sur les Aspects Algorithmiques de Télécommunications (AlgoTel'08). (May 2008)