Adaptive Deployment for Pervasive Data Gathering in Connectivity-Challenged Environments

Tahiry Razafindralambo[◇,×], Nathalie Mitton[◇,×], Aline Carneiro Viana[◇],
Marcelo Dias de Amorim[⋆], and Katia Obraczka[†]

[◇] INRIA, [×] CNRS, Université Lille 1 [⋆] CNRS, UPMC Univ Paris 06 [†] UC Santa Cruz
{tahiry.razafindralambo,nathalie.mitton,aline.viana}@inria.fr, marcelo.amorim@lip6.fr, katia@cse.ucsc.edu

***Abstract*—Some current and future pervasive data driven applications must operate in "extreme" environments where end-to-end connectivity cannot be guaranteed at all times. In fact, it is likely that in these environments partitions are, rather than exceptions, part of the normal network operation. In this paper, we introduce Cover, a suite of adaptive strategies to control the trajectory of "infrastructure" nodes, which are deployed to bridge network partitions and thus play a critical role in data delivery. In particular, we focus on applications where end (or target) nodes are mobile and their mobility is unknown. Our goal is then to *deploy* and *manage* infrastructure nodes so that application-level requirements such as reliable data delivery and latency are met while still limiting deployment cost and balancing the load among infrastructure nodes. Cover achieves these goals using a localized and adaptive approach to infrastructure management based on the *observed* mobility of target nodes. To this end, Cover takes advantage of contact opportunities between infrastructure nodes to exchange information about their covered zones, and thus, help monitor targets in a more efficient fashion. Through extensive simulations, we show how Cover's adaptive features yield a fair distribution of targets per infrastructure node based only on limited network knowledge.**

I. Introduction

Context. Emerging pervasive communication systems will face a number of challenges including the need to operate in "extreme" environments and thus withstand frequent and arbitrarily long-lived connectivity disruptions. These disruptions in connectivity may be caused by a number of factors including node mobility, wireless channel impairments, participating nodes' energy and communication capability limitations, sparse deployments, etc. Examples of applications likely to operate in these extreme environments include emergency response and disaster recovery, environmental and habitat monitoring, vehicular networks, etc. In the literature, these networks have been referred to as intermittently-connected, highly-partitioned, or delay-tolerant networks (DTN) [1]. We focus specifically on scenarios where end nodes (or targets) such as vehicles, animals being monitored/tracked, humans, etc. are mobile and little a priori information about their mobility is known.

Motivation. It is then critical to design efficient protocols to support pervasive, "any time, any place" services in these networked environments prone to connectivity disruptions. Some of the main challenges include satisfying application-specific requirements under intermittent connectivity and without prior knowledge of network topology characteristics such as node location and mobility. This is a fairly complex zone coverage problem and constitutes the main focus of this paper.

Previous work on deployment management using mobile infrastructure nodes (e.g., robots) focused on static targets [2], [3], [4], [5], [6]. Here the goal is to "cover" multiple targets satisfying application-specific requirements (e.g., data freshness, delivery latency) without prior knowledge of the targets' location or mobility. Additionally, most efforts to-date use "flat" deployments, where mobile targets are also used for core network functions such as data routing and forwarding [7], [8], [9], [10], [11]. Instead, in this paper, we consider *two-tiered deployments composed not only by mobile targets but also by mobile infrastructure nodes.* Moreover, we also suppose that data-producing nodes (targets) do not have permanent network connectivity; instead, we assume that, through mobile infrastructure nodes, they are periodically connected to the network at most every t seconds. Such infrastructure nodes are specialized nodes whose trajectories are controlled and adapt over time to the targets' mobility. The main issue in such a context is then how to *deploy* and *manage* the infrastructure nodes in order to guarantee that all the target nodes are covered while respecting the application constraints and balancing load. In the sequel, a target node is said to be "covered" if it is connected to the network using an infrastructure node at least every t seconds.

Contributions. To improve the availability of the system with respect to the driving applications, this paper introduces Cover, an adaptive strategy for infrastructure node placement and (trajectory) control. Cover relies on localized mechanisms that combine information about characteristics of the nodes and application requirements. Such fundamental problem has received very little attention in the literature. In particular, we focus on the following network deployment problem: Given an area to monitor and a set of target nodes with unlimited mobility within this area:

- Cover is localized, i.e., every decision taken by in-

978-1-4244-5328-3/09/$25.00 ©2009 IEEE

frastructure nodes is based on local neighborhood information only. Cover takes advantage of contact opportunities between infrastructure nodes to exchange information about their covered zones, and thus, help monitor target displacement in a more efficient fashion. As a consequence, Cover scales better with regard to the network size and the mobility of the nodes.
- Cover ensures that every data-producing node is connected regularly to an infrastructure node.
- Cover balances the load between infrastructure nodes for an improved functioning of the system.

The remainder of this paper is structured as follows. In Section II we discuss the problem statement and present our system model and assumptions. In Section III, we describe the main components of Cover. Through extensive simulations, we evaluate Cover's performance under different node densities and mobility patterns in Section IV. We evaluate several performance metrics. The results show among other issues, the impact of the mobility pattern of targets on the shape of zones covered by infrastructure nodes. In addition, results confirm the good distribution of targets per node, which is reached independently of the mobility patterns of the targets. Finally, Section V presents the related work and Section VI concludes the paper.

II. Problem statement, model, and assumptions

In the system considered in this paper, target nodes (TNs) generate data that must be collected by infrastructure nodes (INs). Since the mobility of TNs is unpredictable and uncontrolled, the only solution is to control the trajectory and speed of INs to meet the required frequency of readings defined by the application. This means that every TN needs to encounter an IN regularly and these encounters have to occur at a minimum frequency (or within more some maximum delay). In other words, a TN is said to be covered if it is connected to the network using an IN at least every t seconds. Since the locations of TNs are constrained by the area to be monitored, covering each physical point of the area every t seconds ensures the coverage of each TN.Furthermore, for the sake of load balancing among the INs, we must balance the number of TNs to be monitored by INs.

To meet the requirements listed above, Cover assigns geographical zones to INs. To this end, iINs constantly check the number of TNs they cover and adapt their trajectory when necessary. Below, we describe our scenarios and assumptions . Table I summarizes the notation used in this paper.

A. Target area and population

We assume that the target area to be monitored is known and is a square of size $L \times L$. We divide this area into C cells of size $l \times l$, where $l = L/\sqrt{C}$. In this area, we deploy N TNs and M INs. Note that the cell defines the minimum area

Table I
SUMMARY OF NOTATION.

Notation	Meaning
TN	Target node
IN	Infrastructure node
N	Number of TNs
M	Number of INs
C	Number of cells
m_i	Infrastructure node i
$f_{\min}$	Minimum reading frequency
$\Gamma(m_i)$	Set of m_i's neighbors
$T_{\max}$	Maximum number of TNs per cell
v_{IN}	Speed of INs
v_{TN}	Speed of TNs
$T(m_i)$	Number of TNs covered by m_i
T_{opt}	Optimal number of TNs per INs
$Z(m_i)$	Number of cells covered by m_i
$Z_{\max}$	Maximum number of cells an IN can cover

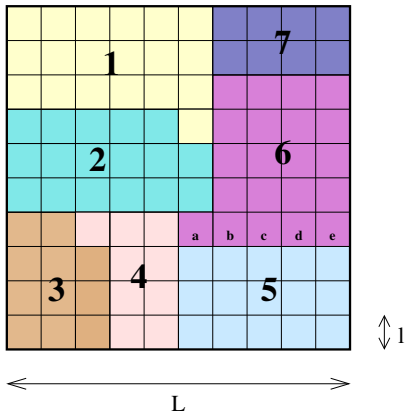


Figure 1. Illustration of an area of 10×10 cells covered by 7 INs. The area covered by each of them is denoted by the node's identifier.

that a stationary IN can continuously monitor. Thus, if the number of INs $M \geq C$, there is no need for infrastructure adaptation. In this paper, we consider the case where each IN has to monitor multiple cells (i.e., $M < C$). We denote the speed of INs and TNs as v_{IN} and v_{TN}), respectively.

Let $Z(m_i)$ be the number of cells in the zone covered by IN m_i and $T(m_i)$ be the number of TNs in these cells (i.e., the number of TNs monitored by m_i). We refer to these variables as, respectively, *zone number* and *cover number*. We also assume that there is a maximum number of TNs $T_{\max}$ that a single cell can accommodate, which limits the number of TNs covered by m_i to $T_{\max} \times Z(m_i)$.

We assume that TNs have the same computational–, memory–, energy–, and communication capabilities. On the other hand, we consider that INs have no energy constraints and longer communication ranges. Two INs are considered neighbors if the zones they cover share at least one border of size l. Fig. 1 illustrates one possible configuration for seven INs. In this example, m_2 has four neighbors (m_1, m_3, m_4 and m_6), while m_7 has only two neighbors (m_1 and m_6). We denote $\Gamma(m_i)$ the set of m_i's neighbors.

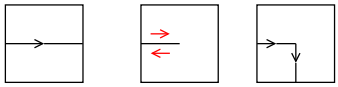


Figure 2. Three ways a cell can be traversed by INs.

B. Data reporting

By definition, TNs report to INs. An IN can retrieve data from a TN if and only if they occupy the same cell at the same time. An IN computes its trajectory as a function of the set of cells it has to monitor. INs have to regularly visit $Z(m_i)$ cells – the *coverage* of an IN is called a *cycle* which is accomplished by visiting all cells of a zone. The frequency at which a IN visits a cell is specified by $f_{\min}$. Finally, we assume that the time an IN spends in each cell is enough to retrieve data from at most a maximum number of TNs $T_{\max}$.

Cells can be traversed in three different ways as illustrated in Fig. 2. Note that traversing a cell always requires traveling a distance equal to l. We assume that INs have no prior knowledge on the mobility of the TNs.

III. Adaptable zone coverage

In this section, we describe how Cover guarantees coverage of the whole monitored area, while supporting the requirements of the application. Cover's algorithm is described in detail below and its pseudo-code presented by Algorithm 1.

A. Meeting constraints with preliminary settings

Frequency reading. l is chosen such that a TN (with a given v_{TN} speed) can cross at most one cell during a reading period. This implies that $v_{TN} \leq l \times f_{min}$. A cell has to be visited by an IN at least twice every $\frac{1}{f_{\min}}$ to ensure that every TN is covered at least once. Under these assumptions and for the reading frequency to be respected, the number of cells an IN can monitor (i.e., $Z(m_i) \leq Z_{\max}$) has to be bounded with regards to the IN speed and the required reading frequency $f_{\min}$, where $Z_{\max} = \frac{v_{IN}}{f_{\min}} \times \frac{1}{2l}$. Indeed, $\frac{v_{IN}}{2f_{\min}}$ represents the maximum number of cells that an IN can cover during a cycle of duration $\frac{1}{2f_{\min}}$.

Coverage: At bootstrap, we assume that INs are uniformly distributed (cf. [12]) so that the area to be covered, which is composed of C cells, is equally partitioned among INs. Since an IN can cover at most $Z_{\max}$ cells, the system needs a minimum number of INs, $M_{\min}$, such that $M_{\min} = \left\lceil \frac{C}{Z_{\max}} \right\rceil$. Thus, INs are assigned zones such that:

1) $\left\lceil \frac{C}{M} \right\rceil$ cells are assigned to $(C \bmod M)$ INs.
2) $\left\lfloor \frac{C}{M} \right\rfloor$ cells are assigned to $(M - (C \bmod M))$ INs.

Indeed, at bootstrap, every cell is covered by exactly one IN. As we will see later, since at each step of the algorithm an IN delegates a cell if and only if another IN accepts to monitor it, complete coverage is always guaranteed.

Algorithm 1 Cover- Run periodically at every m_i.

```
1:  LEAVE← 0,
2:  if T(m_i) > T_opt + θ then
3:     {IN m_i covers too many TNs, it has to share.}
4:     LEAVE← 1
5:  end if
6:  if (LEAVE=0) then
7:     if Check_Neigh(m_i, α) = TRUE then
8:        LEAVE← 1
9:     else
10:       flag(m_i)← 0
11:       LEAVE← 0
12:    end if
13: end if
14: if LEAVE= 1 then
15:    u ←SELECT_IN(m_i)
16:    if u ≠NULL then
17:       LEAVE_CELL(u)
18:    else
19:       flag(m_i)← 1
20:    end if
21: end if
```

Load balancing among INs: Cover tries to balance the load among INs by trying to assign an equal number of TNs per IN. Thus, the number of TNs covered by m_i has to tend to an optimal value $T_{\text{opt}} = N/M$ in order to balance the load among INs. Indeed, if every IN covers T_{opt} distinct TNs, by definition, the whole set of TNs is covered and coverage sets are non-overlapping.

B. Periodic learning

The proposed algorithm is run in a distributed way at each IN m_i. Every m_i is aware of its cover number $T(m_i)$ and zone number $Z(m_i)$ at all times, which it regularly broadcasts to its 1-hop neighbors. If correctly performed by the INs, the algorithm balances the number of TNs covered by each IN while ensuring the reading frequency and the coverage of the whole area. The algorithm works as follows. IN m_i regularly compares its cover number to $T_{\text{opt}} + \theta$, where θ is a constant such that $2\theta > T_{\max}$. If $T(m_i) > T_{\text{opt}} + \theta$, this means that m_i monitors too many TNs (cf. lines $2-5$ in Algorithm 1). For a given m_i, if $T(m_i) \geq T_{opt} + \theta$, a variable LEAVE is set to 1. In this case, m_i checks whether it can delegate a cell to one of its neighbors u (cf. line 15 in Algorithm 1; the procedure SELECT_IN(m_i) is detailed in Algorithm 2). Cells are delegated one at a time (cf. line 17 in Algorithm 1). So, m_i delegates cells till $T(m_i) < T_{\text{opt}} + \theta$ or when no neighbor m_j is able to receive an extra cell. However, as we will see later, this latter case is quite unfrequent and is considered in lines $6-13, 19$ in Algorithm 1.

Algorithm 2 `Select_IN`

1: $V \leftarrow m_j \in \Gamma(m_i)$ such as $Z(m_j) < Z_{\max}$
2: {V is the set of neighbors of IN m_i which zone number is smaller than the maximum allowed.}
3: $V' \leftarrow m_j \in V$ such as $T(m_j) < T_{opt} - \theta$
4: {V' is the set of neighbors of IN m_i which cover number is smaller than the maximum allowed.}
5: **if** $V' \neq \emptyset$ **then**
6: $u \leftarrow m_w \in V'$ which minimizes $T(m_w) \times Z(m_w)$
7: Return u
8: **else**
9: Return NULL
10: **end if**

C. Neighbor selection SELECT_IN()

Neighborhood discovery is performed by exchanging hello messages. IN m_i observes the cover and zone numbers of each of its neighbors to decide to which one it should delegate a cell. It first selects the neighbor m_j such that: (i) $Z(m_j) < Z_{\max}$, $\forall m_j \in \Gamma(m_i)$, in order to ensure the reading frequency is met (cf. line 1 in Algorithm 2), and (ii) $T(m_j) < T_{\text{opt}} - \theta$, $\forall m_j \in \Gamma(m_i)$, in order to balance the number of TNs per IN (cf. line 3 in Algorithm 2). As θ is such that $2\theta > T_{\max}$, delegating a cell to m_j cannot increase m_j's cover number above $T_{\text{opt}} + \theta$, which avoids a flip-flop phenomenon.

If the subset of neighbors satisfying the requirements listed above is non empty, m_i computes, for each of them, the product of their cover by their zone number, $Z(m_j) \times T(m_j)$, and selects the one with the smallest product (cf. line 6 in Algorithm 2). In case of ties, m_i chooses at random between candidates. The product allows considering both features at the same time, without privileging one over the other.

D. Cell delegation LEAVE_CELL()

When assigning a cell to a neighbor (line 17 in Algorithm 1), m_i gives preference to cells that allow keeping its zone connected and as "compact" as possible. By compact, we mean that m_i tries to minimize the size of its border. For instance, in Fig. 1, if m_6 decides to delegate a cell to m_5, it has the choice between cells a, b, c, d and e. If m_6 decides to delegate cell b, its zone would be disconnected since cell a will be no longer adjacent to another cell of m_6. If m_6 chooses cell c or d, its remaining zone will be less compact (its border is enlarged by $2l$. Finally, m_6 has choice between cells a and e. In order to optimize its trajectory, m_6 selects cell a (its border is then reduced by $2l$). In this way, its zone remains connected and compact. Note that in some cases there might be only one candidate cell to be transferred.

After selecting the target neighbor m_j, m_i initiates a negotiation phase with m_j. m_j may refuse m_i's request if it is already in a negotiation phase with another IN. If so, m_i has to select another neighbor following the same rules as previously described. This condition guarantees that any IN receives at most one cell per step. This also prevents it from violating its cover number and having to delegate a cell.

E. Dealing with exceptions Check_Neigh()

It may happen that an IN has too many TNs to monitor and cannot delegate any of its cells to a neighbor. This may occur for two reasons: (i) neighbors have reached the proper cover number, (ii) or they have reached the maximal allowed number of cells to monitor. To better understand this case, let us consider again the example of Fig. 1. Assume that m_5 needs to give a cell to one of its neighbors, i.e., to m_4 or m_6. Also assume that m_4 and m_6 cannot receive any cell due to their current cover number being between $T_{opt} - \theta$ and $T_{opt} + \theta$. m_5 is thus "deadlocked". During a deadlock situation, m_5 sets a flag to 1 in the beacon it regularly sends (cf. line 19 in Algorithm 1). Since $LEAVE = 0$ for m_4 and m_6, they meet condition of line 6 in Algorithm 1. Check_Neigh is called by m_4 and m_6 (line 7 in Algorithm 1 and described in Algorithm 3). This function is called with two parameters m_i (let us say m_4), the id of the IN and α. If $\alpha = 0$ when Check_Neigh is called, a new random positive integer is used (cf. line 3 in Algorithm 3). This function returns a boolean called $force\_cell\_leaving$ which is set to FALSE at the beginning of the function. α is decreased by 1. If the value of α is 0, and one or more of m_4's neighbors has a flag set to 1, $force\_cell\_leaving$ is set to TRUE (cf. lines $6 - 12$ in Algorithm 3). This is the case in our example since m_5 is in deadlock. On the other hand, if $\alpha > 0$ or no neighbor has a flag set to 1, $force\_cell\_leaving$ is left to FALSE.

The return value of Check_Neigh (TRUE or FALSE) is used in line 7 in Algorithm 1. If Check_Neigh returns TRUE, a cell has to be delegated, and LEAVE is set to 1 (cf. line 8 in Algorithm 1). On the other hand, if Check_Neigh returns FALSE, LEAVE is left to 0 (cf. line 11 in Algorithm 1). In this case, the IN will re-enter Algorithm 1 with the previous value of α. Yet, in Fig. 1, if m_4 triggers a smaller value of α than m_6, it can delegate a cell to m_3 and then receive a cell from m_5 which will recover from the deadlock condition. It is worth noting that using the remaining value of α increases fairness. Indeed, if m_5 has more cells to delegate, m_6 will probably have a smaller (remaining) value of α than m_4.

IV. Performance Analysis

We evaluate the performance of Cover using the WSNet simulator [13]. Since by construction, coverage and reading frequency requirements are always satisfied, we mainly focus on IN load balancing. Due to the fact that TNs may be mobile, the number of TNs in a cell can not be bounded. Therefore, the parameter θ is set to 1 since the average number of TNs in a cell is $N/C < 1$ (cf. Table II). Although

Algorithm 3 `Check_Neigh`(m_i, α)

force_cell_leaving$\leftarrow FALSE$
if $\alpha = 0$ **then**
  $\alpha \leftarrow rand()$
end if
$\alpha \leftarrow \alpha - 1$
if $\alpha = 0$ **then**
  for all $m_j \in \Gamma(m_i)$ **do**
    if flag(m_j)=1 **then**
      force_cell_leaving$\leftarrow TRUE$
    end if
  end for
end if
Return force_cell_leaving

Table II
SIMULATION SETTINGS.

Parameter	Value
network area L	$1,008\text{m} \times 1,008\text{m}$
size of cells l	28 m $\times$ 28m
Number of cells = C	$1,296$
NUmber of TNs = N	250
mobility models	STA, BRO, RWP, REB
simulation duration	500s

such a value for θ slows down the system and constraints cell exchanges, it allows us to evaluate Cover's robustness.

We consider a square field of $1008\text{m}\times1008\text{m}$ divided into 36×36 cells of edge $l = 28\text{m}$. We evaluate the evolution of the network under the following mobility models for the TNs:

- **STA:** (for stationary) TNs are stationary and uniformly distributed in the area. This model is only used as a performance baseline. It also helps highlighting the algorithm's performances under stable conditions.
- **BRO**: (for random walk mobility) TNs choose a direction between north, south, east, and west and move toward it for 1s at the speed of 1m/s. Targets stop then for 2s and repeat the process. This model may illustrate movement of cars in a city like New York City.
- **RWP:** (for random way point) TNs travel from a starting point to a randomly chosen destination at a randomly chosen speed between $[0.1; 5]$m/s. When it reaches its destination, it pauses for 2s before randomly choosing a new destination. This model may illustrate worst movement case of animals in large space.
- **REB**: (for rebound) TNs draw a random angle between $[0, 2\pi]$ and a speed between $[0.1; 5]$m/s. When they reach a border, they bounce with the same angle. Unlike in RWP, here, TNs are more likely to be spread out in the field.

These mobility models were chosen due their unique features [14], such as:

- Memory-less, to avoid INs from learning the mobility pattern and thus to simulate worst case. Note that if INs can estimate the mobility pattern or if the TNs follow a group mobility models, the performance of the algorithm can be increased. This investigation is left to future works.
- Different node distribution. This characteristic helps us to evaluate the performance of the algorithm when TNs are not uniformly distributed.

Infrastructure nodes run Cover every t seconds, where t is randomly and uniformly chosen at each step in the interval $[0.9; 1.1]$ in order to break synchronization and to have an up-to-date coverage. Simulation parameters are summarized in Table II.

Simulation results are divided into three parts. Section IV-A is dedicated to the evaluation of the zone coverage along time for different mobility models. Results show that the shape of the zones strongly depends on the mobility pattern of the TNs. Section IV-B focuses on the quality of distributions of the cover and zone numbers. Results reveal that Cover exhibits good performances as the distribution of TNs per IN follows a normal distribution around the average. They also show that more than 50% of the INs cover the optimum number of TNs T_{opt}. This reveals the good cell distribution per IN. Finally, Section IV-C shows the evolution of the zone and cover numbers per IN along time to measure the quality of the load balancing. Simulations show that our algorithm evenly distributes the number of TNs per IN. Furthermore, it is shown up that Cover balances the number of TNs per IN independently from the mobility of the TNs. Moreover, Cover is highlighted to adapt very quickly to changes due to the mobility of TNs (in less than 10 rounds). We also show that there is a tradeoff between the zone number and the number of TNs per IN. All the gotten results are discussed in detail in the following.

A. Zone coverage evaluation

We first evaluate the zone coverage evolution along time. At initialization, all INs have the same zone number (or almost the same for uneven topologies). Targets are randomly and uniformly distributed over the field. Fig. 3 shows the evolution of zone shapes for different mobility models at different simulation times. Different grayscales denote different coverage zones (one per IN, 9 INs).

When TNs are static and uniformly distributed, Fig. 3 shows that the zone distribution is close to the initial one. We can also see that after 160s, the zone distribution reaches its final shape. In the RWP model, the IN in the middle of the field has a small number of cells. This can be explained as follows. In the RWP model, TNs are concentrated in the middle of the field [15], [16]. Therefore, in order to balance its cover number, the corresponding IN gives a large part of its cells to other nodes. On the other hand, zone distributions

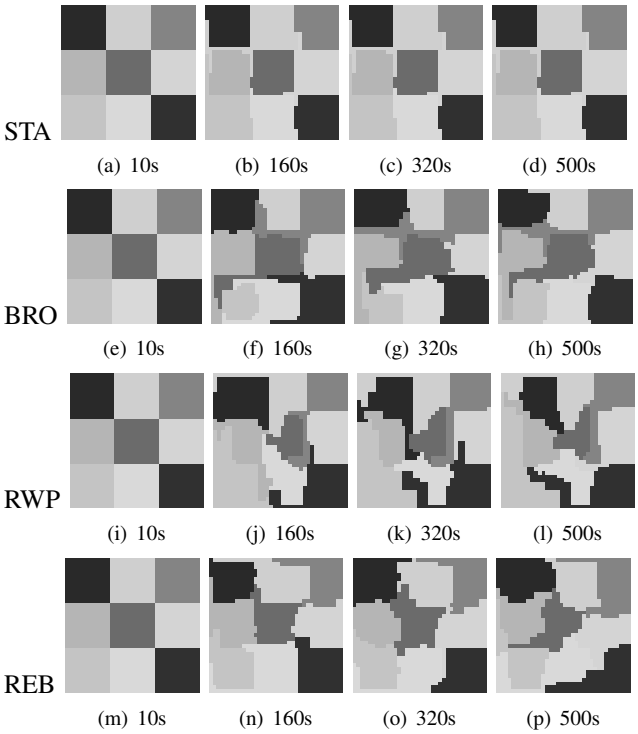


Figure 3. Example of zone evolution with 9 INs depending on simulation time. The zone shape evolves with time and is related to the mobility pattern.

for the BRO and the REB mobility models are more likely to be spread out over the field.

B. TN distribution

In this section, we plot the distribution of TNs for two different numbers of INs (30 and 70) and for the four different mobility models. The distribution is computed after 500s of simulation and we run 30 simulations for each graph. The simulation setup gives the following indications: the average number of TNs per IN is $\sim$8.33 for 30 INs and $\sim$3.5 for 70 INs, while the average numbers of cells per IN are, respectively, $\sim$43.2 and $\sim$18.5.

Distribution of the cover number: Fig. 4 shows the distribution of TNs per IN. In each case, the distribution is close to a normal distribution. Note that more than 50% of the INs cover the average number of TNs. Thus, Cover performs well independently from the mobility of TNs.

We also conducted a number of simulations to check the dependency of the number of INs. The results are shown in Fig. 5(a). As we can see, Cover fairly balances the number of TNs per IN, whatever the number of INs and the mobility pattern followed by the TNs.

Distribution of the zone number: Fig. 6 plots the distribution of cells per IN. The results show that, as expected, the performance of our algorithm is increased when the TNs are static. Fig. 6 shows that the zone number distribution

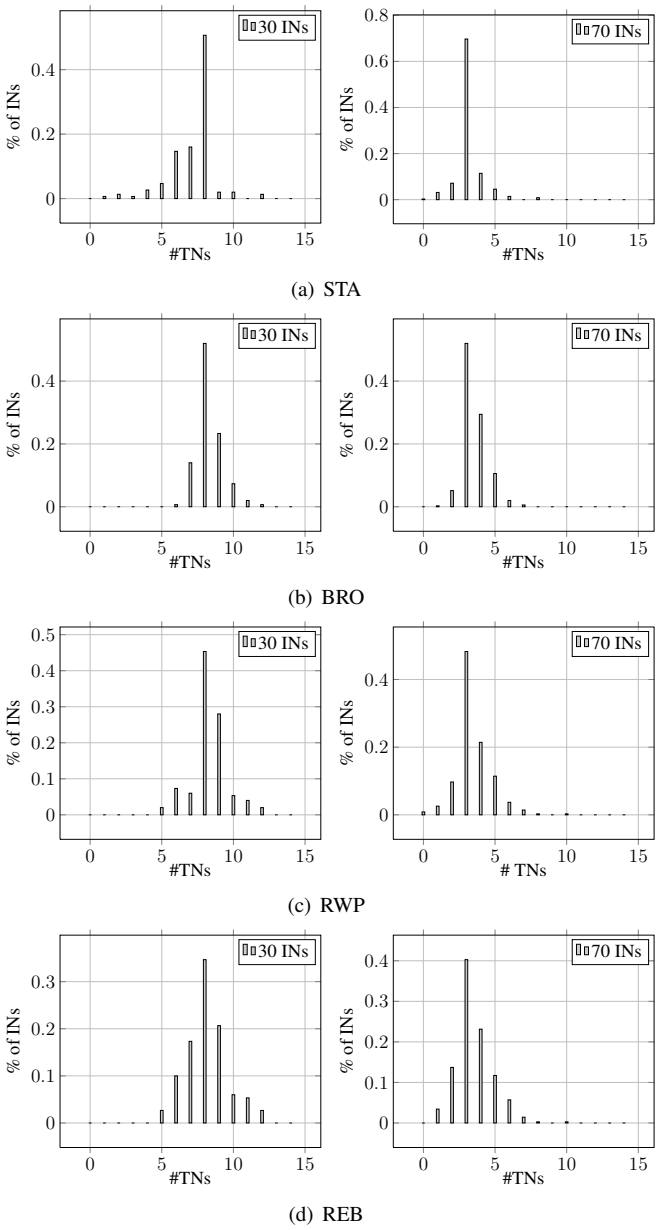
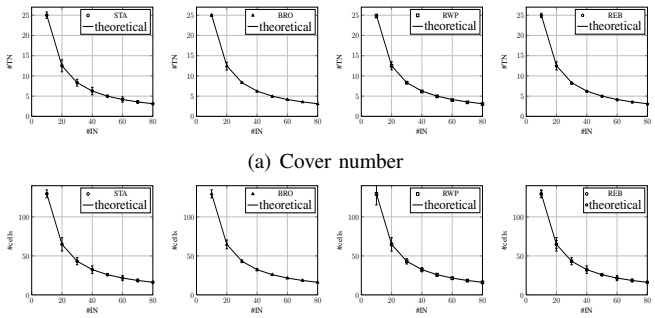


Figure 4. Distribution of the cover number for different number of INs (30 and 70) for different mobility model. The number of TNs per INs is very close to a normal distribution which shows the efficiency of the algorithm for load-balancing.

approaches a normal distribution around the average number, except for RWP. For this latter case, the result is related to the one provided in Section IV-A, where the IN in the middle of the field has a smaller number of cells, and that INs in the border of the field have a larger zone number. When the number of INs increases, this behavior is alleviated because the number of INs in the middle of the field increases, which can reduce the cover number of the INs. However, this behavior is expected in our algorithm since Cover does not try to balance the cover zone in priority. It first adapts to TNs



(a) Cover number

(b) Zone number

Figure 5. Cover number and Zone number w.r.t. the number of INs. (95% confidence interval). We also plot the theoretical value. The number of TNs per INs is close to the optimal theoretical value for all the mobility pattern.

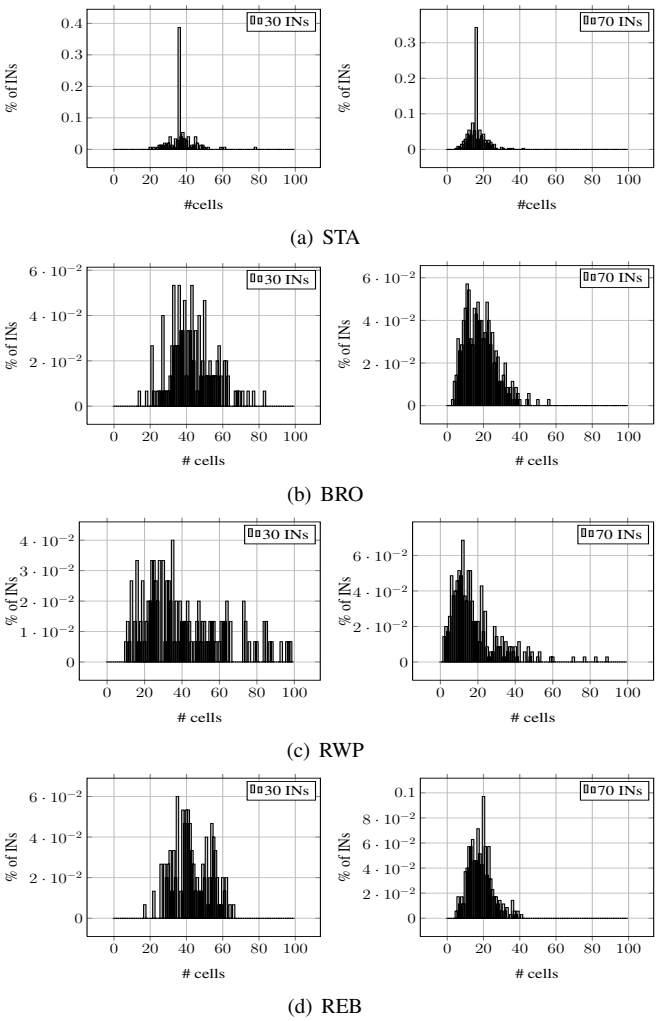
mobility. Since TNs do not end up in a uniform distribution, the same applies to the zone numbers. However, Fig. 5(b) shows that the average number of cells per IN corresponds to the theoretical average. Further investigations are needed to provide a better trade-off between the two metrics. Finding this tradeoff is left to future works.

C. Evolution of zone and cover numbers

In this section, we plot the evolution of the cover number and zone number of INs over time. Results provided in Fig. 7 show that our algorithm always tries to maintain the number of TNs close to the average and also show that under stable condition, convergence is fast. In these simulations, we consider 80 INs, which leads to 3 TNs per IN and 16.2 cells per IN in average. It is important to notice here that Fig. 7 shows the evolution of the cover number and zone number for a specific INs (randomly chosen).

At the beginning of the simulation, the number of cells for a specific IN m_i is equal to the average value (due to simulation setup) but $T(m_i)$ can be greater or lower than T_{opt}. In Figure 7(a) when TNs are static and $T(m_i) < T_{\text{opt}}$, we can see that $T(m_i)$ increases until it reaches the average value in less than 10 seconds (10 rounds). At the same time, the number of cells also increases. This shows that our algorithm tries to evenly distribute the number of TNs per IN instead of maintaining the number of zone per IN. This behavior explains the results provided in Section IV-B, which shows the distribution of zone per IN. When TNs are mobile, $T(m_i)$ can vary; therefore, cells are exchanged between INs and their neighbors. Figures 7(b), 7(c) and 7(d) show that the maximum number of TNs per INs is 6 and the minimum is 0 (for the specific observed IN) and that the algorithm tries to evenly distribute the number of TNs per INs.

Fig. 7 also shows that our algorithm presents fast reaction when the number of TNs vary, although exchanged cells does not necessarily contain TNs.



(a) STA

(b) BRO

(c) RWP

(d) REB

Figure 6. Distribution of the zone number per INs for different mobility models.

V. Related work

Most existing solutions to the area coverage problem have been designed to cover static points/targets and they can be divided in three main categories: 1) random deployment of sensors, which consists of having a in a large number of sensors randomly deployed with activity scheduling or power control techniques being used to reduce the network density [17], [18]; 2) off-line computation of sensor placements, which is based on network performance, connectivity and area coverage [19], [20], [21]; and 3) sensor repositioning schemes, which mainly focuses on the sensor (re)positioning or online placement [22], [23]. Some similar approaches to Cover also take advantage of node mobility to enhance network connectivity [24] but in these approaches, the mobility of target nodes is known and controlled. For the interested reader, a good survey of current approaches is presented by Younis and Akkaya [25].

Here, we only focus on coverage involving mobile devices

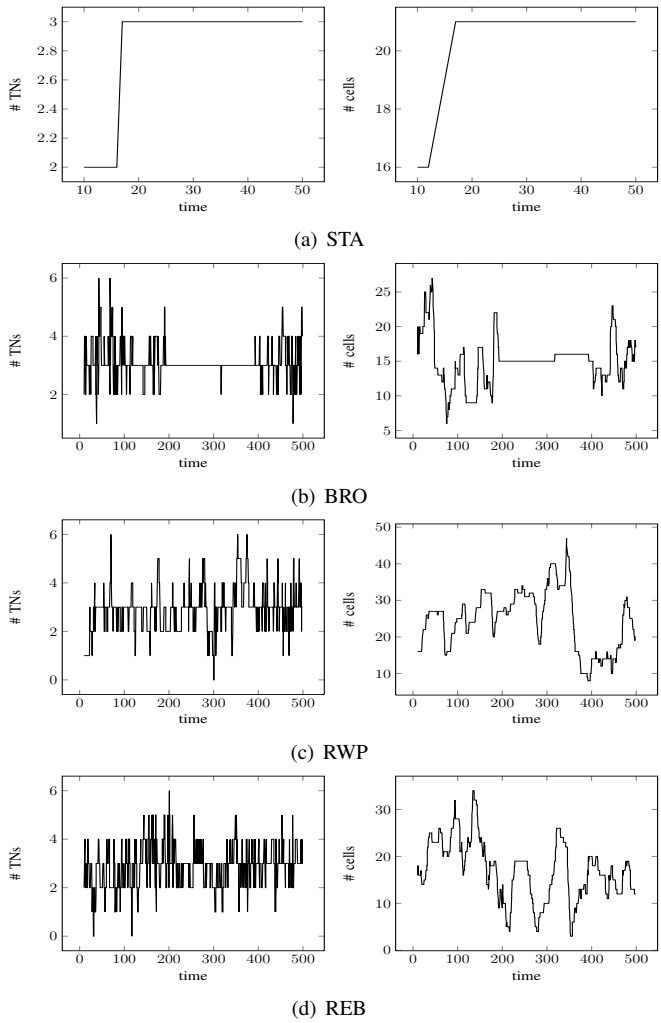


Figure 7. Evolution of the cover and zone numbers over time for a specific IN. In this simulation, we consider 80 INs and a simulation time of $500s$. Optimal value of TNs per IN is ~ 3 and average value of Cells per IN is ~ 16.

(sensor repositioning scheme) since in our algorithm INs are mobile and are used for coverage purpose. Coverage requirement provided in the literature can be divided into three main categories:

- In the full coverage problem, sensors have to maximized the covered area. The work proposed in [22] and [23] uses virtual force based movement to increase the covered area. By using a combination of mutually opposing forces, each mobile node maximizes its coverage.
- In barrier coverage problem [26], sensors have to form a barrier that detects any events crossing the barrier. A barrier is defined as a segment between two points of the sensor field between which the sensors have to be evenly distributed. The work proposed in [27] uses virtual forces to relocate sensors.
- In the points of interests coverage, only some specific points of the sensor field need monitoring. [28] consider points coverage. In these papers, the authors propose an algorithm to periodically monitor some specific points.

In this paper, we focused on a different coverage problem which targets coverage of mobile targets by mobile infrastruture nodes. As in the case of point of interest coverage, mobile targets have to be covered by the mobile infrastructure nodes. Moreover, since targets are moving, a full coverage is needed to cover all possible location. Since our infrastructure nodes are mobile, we also consider online deployment issues. These requirements and assumptions lead to a complex coverage problem that we think is part of a new category of coverage strategies.

VI. Summary and Outlook

In this paper, we presented Cover, a distributed algorithm for adaptive infrastructure deployment in two-tiered intermittently connected networks. Infrastructure nodes track mobile target nodes while respecting the reading frequency requirement and keeping a (close to) optimal ratio of the number of targets per infrastructure node. We showed through theoretical analysis and extensive simulations that our algorithm converges when the target nodes are fixed and that the number of target nodes per infrastructure node is close to the optimum at any time independently of the mobility pattern of the target nodes.

Future work includes considering different assumptions regarding the communication stack used in each node, investigating ways to reduce the number of infrastructure nodes, and proposing loose algorithms that allow some parts of the area to be temporarily uncovered. This will require combining different coverage techniques such as barrier coverage and sweep coverage.

References

[1] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proc. of ACM SIGCOMM*, 2003.

[2] M. A. Batalin and G. S. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," *Telecommunication Systems Journal, Special Issue on Wireless Sensor Networks*, vol. 26, pp. 2–4, 2004.

[3] I. Chatzigiannakis, A. Kinalis, and S. Nikoletseas, "Sink mobility protocols for data collection in wireless sensor networks," in *Proc. of ACM MobiWac*, Malaga, Spain, 2006.

[4] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley, "Mobility improves coverage of sensor networks," in *Proc. of ACM MobiHoc*, Urbana-Champaign, IL, USA, 2005.

[5] G. Wang, G. Cao, and T. L. Porta, "Movement-assisted sensor deployment," in *Proc. of IEEE Infocom*, San Francisco, USA, 2003.

[6] W. Wang, V. Srinivasan, and K. C. Chua, "Trade-offs between mobility and density for coverage in wireless sensor networks," in *Proc. of ACM Mobicom*, Germany, 2007.

[7] E. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *Proc. of ACM MobiHoc*, 2007.

[8] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot, "Delegation forwarding," in *Proc. of ACM MobiHoc*, 2008.

[9] P. Hui, J. Crowcroft, and E. Yoneki, "BUBBLE Rap: Social-based forwarding in delay tolerant networks," in *Proc. of ACM MobiHoc*, 2008.

[10] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Efficient routing in intermittently connected mobile networks: The single-copy case," *IEEE JNET*, vol. 16, no. 1, pp. 63–76, 2008.

[11] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," Duke University, Tech. Rep., 2000.

[12] K. Huguenin, A.-M. Kermarrec, and E. Fleury, "Route in mobile wsn and get self-deployment for free," in *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Marina del Rey CA, USA, 2009.

[13] "Wsnet simulator," wsnet.gforge.inria.fr.

[14] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, pp. 483–502, 2002.

[15] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," in *Proc. of IEEE Infocom*, 2003.

[16] J.-Y. L. Boudec and M. Vojnovic, "Perfect simulation and stationarity of a class of mobility models," in *Proc. of IEEE Infocom*, 2005, pp. 2743–2754.

[17] J. Carle and D. Simplot-Ryl, "Energy-efficient area monitoring for sensor networks," *Computer*, vol. 37, no. 2, pp. 40–46, 2004.

[18] D. Simplot-Ryl, I. Stojmenovic, and J. Wu, "Energy-efficient backbone construction, broadcasting, and area coverage in sensor networks," *Handbook of Sensor Networks*, pp. 43–380 343–380, 2005.

[19] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, "Near-optimal sensor placements: maximizing information while minimizing communication cost," *Information Processing in Sensor Networks, 2006. IPSN 2006. The Fifth International Conference on*, pp. 2–10, 0-0 2006.

[20] D. Pompili, T. Melodia, and I. F. Akyildiz, "Deployment analysis in underwater acoustic wireless sensor networks," in *WUWNet '06: Proceedings of the 1st ACM international workshop on Underwater networks*. New York, NY, USA: ACM, 2006, pp. 48–55.

[21] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T. H. Lai, "Deploying wireless sensors to achieve both coverage and connectivity," in *MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2006, pp. 131–142.

[22] G. Tan, S. A. Jarvis, and A.-M. Kermarrec, "Connectivity-guaranteed and obstacle-adaptive deployment schemes for mobile sensor networks," in *ICDCS '08: Proceedings of the 2008 The 28th International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 429–437.

[23] S. Poduri, S. Pattem, B. Krishnamachari, and G. S. Sukhatme, "Using local geometry for tunable topology control in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 2, pp. 218–230, 2009.

[24] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *Proc. of ACM MobiHoc*, 2004.

[25] M. Younis and K. Akkaya, "Strategies and techniques for node placement in wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 6, no. 4, pp. 621 – 655, 2008.

[26] S. Kumar, T. H. Lai, and A. Arora, "Barrier coverage with wireless sensors," in *Proc. of the 11th annual international conf. on Mobile computing and networking (MobiCom)*. New York, NY, USA: ACM, 2005, pp. 284–298.

[27] C. Shen, W. Cheng, X. Liao, and S. Peng, "Barrier coverage with mobile sensors," in *ISPAN '08: Proceedings of the The International Symposium on Parallel Architectures, Algorithms, and Networks*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 99–104.

[28] W. Cheng, M. Li, K. Liu, Y. Liu, X. Li, and X. Liao, "Sweep coverage with mobile sensors," in *IPDPS*, 2008, pp. 1–9.