Mutual Exclusion in Wireless Sensor and Actor Networks*

Ramanuja Vedantham, Zhenyun Zhuang and Raghupathy Sivakumar
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, USA
{ramv, zhenyun, siva}@ece.gatech.edu

***Abstract*— A typical Wireless Sensor Network (WSN) performs only one action: sensing the environment. The need for smart interaction with the environment has led to the emergence of Wireless Sensor and Actor Networks (WSANs). The evolution from WSNs, which can be thought of to perform only read operations, to WSANs, which can perform both read and write operations, introduces unique and new challenges that need to be addressed. In this context, we identify the problem of *mutual exclusion*, which is the requirement to act only to the desired level for any particular location and command. We define the different types of mutual exclusion and the associated challenges in the context of WSANs, and show the undesirable consequences of not providing mutual exclusion with example applications. To address this problem efficiently, we propose a greedy centralized approach, and a distributed and fully localized approach based on the centralized approach. Through simulations, we study the performance of the proposed solution with the centralized approach and a baseline strategy, and show that the proposed solution is efficient for a variety of network conditions.**

I. Introduction

Wireless Sensor Networks (WSNs) are widely used in a variety of applications in civilian, medical and military applications. However, the nodes in such a network are limited to one type of action: *sensing the environment*. The need for intelligent interaction with the environment has led to the emergence of a new class of networks capable of performing both sensing and acting on the environment, which we refer to as Wireless Sensor and Actor Networks (WSANs) [1].

The evolution from WSNs, which can be thought of to perform only *read* operations, to WSANs, which can perform both *read and write* operations, introduces unique and new challenges that need to be addressed. In this paper, we address one such challenge pertaining to utilization of actor resources. As an example, consider an automated sprinkler system, with humidity sensors and sprinklers as actors. The sprinklers are activated when the humidity sensor readings go below a certain threshold. Here, it is preferred that only a *minimum subset of sprinklers is activated to cover the entire region so that overall sprinkler resources (water), and energy is minimized*. Thus, depending on the nature of the application, the outcome of not acting to the appropriate level can result in (i) inefficient usage of actor resources, as in the example mentioned above, (ii) incorrect operation (e.g. heat sensors and alarm buzzers in an automated fire-alarm application, where there is *a unique signature related to the frequency and tone with which the alarm buzzes*. Here, when a fire is detected, only *a minimum subset of non-overlapping buzzers* should be activated, so that *the signature does not get scrambled and the fire is detected*), (iii) a catastrophic situation (e.g. *poison gas actors where one dose of the gas merely invalidates subject, but two doses can kill*). We refer to this problem of providing mutually exclusive acting regions to cover an event region as *mutual exclusion*, and identify the different types of mutual exclusion in Section II.

While developing solutions to provide mutual exclusion, the following challenges also need to be addressed: (i) How do we provide mutual exclusion, when there are events of varying intensities? (ii) Is the approach generic to address different types of events such as point/multi-point events as well as regional events? (iii) What happens when the event area decreases or increases? In this context, we propose a greedy centralized approach, and a localized and fully distributed approach to address the different types of mutual exclusion and the associated challenges. In addressing the problem and the associated challenges, (i) the communication overhead should be low without compromising on the delay bound specified by the application, and (ii) the solution should be able to cover the entire event region (correctness). Thus, we make the following contributions in this paper:

- *We identify the different types of mutual exclusion problem and the associated challenges in WSANs.*
- *We present a greedy centralized approach, and a distributed realization that addresses the different types of mutual exclusion and the associated challenges.*

The rest of the paper is organized as follows: Section II defines the context for this work, identifies the different types of mutual exclusion and the associated challenges. Section III presents the greedy centralized approach, while Section IV presents a distributed and fully localized approach that address the problem and the associated challenges. Section V evaluates the performance of the distributed approach with the centralized approach and a baseline strategy. Section VI discusses related work and Section VII concludes the paper.

*This work was funded in part by NSF grants CNS-0519733, CNS-0519841, ECS-0428329, CCR-0313005, and the Georgia Tech Broadband Institute.

Variable	Description
δ	Delay bound between event detection and action initiation
R	Event region
$a_1 \ldots a_k$	Set of actors in R
M	Minimum set of actors to cover R (Actor Set Cover)
R_m	Region covered by m actors
R_i	Acting range of actor, a_i
R_j	Acting range of actor, a_j
r_i	New area covered by any actor a_i
o_i	Overlap between a_i and already existing overlapping regions in R_m
n_i	New overlapping region between R_m and a_i
VM_i	Benefit function of actor, a_i

Fig. 1. Notations to Define Types of Mutual Exclusion

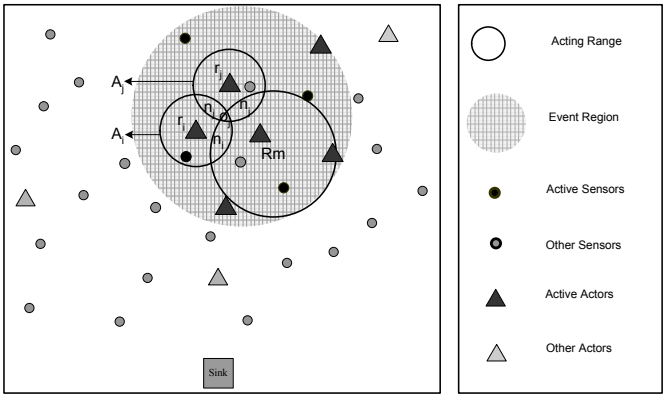


Fig. 2. Different Regions based on the Notation

II. Problem Definition

A. Context

In this paper, we consider an architectural model, where there is a sink to help in the coordination of sensors and actors. This is an extension of the existing architecture for wireless sensor networks, where the sink serves as the coordination entity and issues directives to both sensors and actors. We assume that once the event has been detected, the actors should act on the environment within a delay bound, Δ. This limit is application specific and is representative of the maximum tolerable delay in addressing an event. For simplicity, we consider only one type of action that takes a fixed amount of time, given by the event processing time, T_{EP}. Thus, the maximum delay bound between the detection of an event and the initiation of action is given by $\Delta - T_{EP}$. This delay bound has to be adhered by any solution that addresses mutual exclusion, and is denoted by δ.

For the above model, we focus on a generic class of applications, where there are *regional events that require one round of operation per directive*. The solution to address other classes of applications such as applications where the events (i) are point-events, (ii) require multiple rounds of operation, and (iii) are of variable intensity, can be easily adapted as we explain later in Section IV.

Given the environment, the goal is to identify and address the different types of mutual exclusion in WSANs. The conventional distributed mutual exclusion provides atomic access to a shared critical resource among a group of processes [2]. However, the problem of mutual exclusion is *unique* in the context of WSANs, and is defined as follows: *Given a set of actors in an event region, what is the minimum subset of actors that covers the entire event region such that there is minimal overlap in the acting regions?*

B. Types of Mutual Exclusion

In the above definition of mutual exclusion, the definition of minimal overlap in acting regions can have different connotations subject to application requirements. Figure 1 describes a consistent set of notations used in the rest of the paper, while Figure 2 illustrates the different regions identified in the notation pictorially. Given these definitions, we identify the different types of mutual exclusion in WSANs based on *the strictness of mutual exclusion semantics* and *the intensity of action desired*.

1) Resource Critical Mutual Exclusion: In this definition, the goal is to maximize the non-overlapped acting regions of each actor within the event region in order to utilize the actor resources to the least extent. Thus, in this definition, maximizing the non-overlapped region tacitly defines the minimal overlap in acting regions.

As an example, consider a fire extinguisher application, where there are heat sensors to detect the presence of a fire and sprinklers serve as actors to quench the fire. Consider the case where there is limited amount of water available in the sprinklers. For this application, it is necessary to ensure that there is minimal wastage of water while dousing a fire. It is desirable that each actor selected in the event region has non-overlapping acting regions so that the number of actors selected to cover the event region is minimum. In other words, the new area covered by any actor, a_i, that is added to the already existing cover set should maximize the non-overlapped region, r_i, of that actor (refer Figure 2).

Definition 1: The resource critical mutual exclusion problem is to determine the minimum set of actors, M, that *maximizes* the overall benefit function defined by the sum of individual benefit functions. Here, the individual benefit function, VM_i, of an actor, a_i, is given by:

$$VM_i = r_i \tag{1}$$

2) Overlap-Type Critical Mutual Exclusion: Here, actors are chosen to maximize the non-overlap area in the acting regions while also reducing the amount of new overlap in the acting regions with already existing cover set. This definition is applicable when there is a threshold for the desired level of action and any amount of action beyond this threshold is perceived as undesirable.

Consider an intruder-detection and automated-tranquilizer application, where the sensors are image sensors that detect the presence of an intruder, and the actors are poisonous tranquilizer guns where one dose merely invalidates subject, but two doses can kill. In such a case, when the amount of poison is greater than twice the normal dosage, it results in a fatality irrespective of the exact dosage. For this application, it is imperative to minimize any new overlap as it will result in twice the normal dosage, which will be lethal. However, once an overlap has occurred, it does not matter if there is any additional overlap in the same (overlapped) region by another actor, as long as it minimizes the occurrence of a new overlap region. In other words, it is desirable to choose a new actor that overlaps with previously overlapped regions, as long as it minimizes the new overlap region.

Definition 2: The overlap-type critical mutual exclusion problem is to find the minimum set of actors, M, such that each actor that is added to the cover *maximizes* the non-overlapping region of that actor and *minimizes* any new overlap with the already existing actor cover. The goal is to maximize the overall benefit function defined by the sum of individual benefit functions, where the individual benefit function of an actor is given by:

$$VM_i = r_i - \alpha \times n_i \tag{2}$$

In this equation, α is a constant that represents the cost incurred in having new overlaps in the event region.

3) Overlap-Area Critical Mutual Exclusion: In this definition, it is not only necessary to maximize the amount of non-overlapped region covered by each actor but also equally important to minimize the amount of overlapping regions (both old and new). This corresponds to the scenario where any action that occurs beyond the desired level is unacceptable and the net overlap should be minimized irrespective of whether it occurred in the same region or elsewhere.

Consider the fire extinguisher example mentioned above with heat sensors for sensing and sprinklers as actors. Let the appropriate level of action be described by one actor (sprinkler) acting on a fire event in any particular region. Consider the case, when the acting ranges of sprinklers overlap. In such a case, regions where overlaps occur will result in flooding. In this scenario, apart from maximizing the non-overlapped region, it is required that the sum of all overlapped region is minimized irrespective of whether they are localized or otherwise.

Definition 3: The overlap-area mutual exclusion problem is to determine the minimum set of actors, M, that *maximizes* the non-overlapping and *minimizes* the total overlapping regions of the actor cover, where the individual benefit function of each actor is given by:

$$VM_i = r_i - \beta \times (n_i + o_i) \tag{3}$$

Here, β is a constant that represents the cost incurred in having any kind of overlap in the event region.

4) Overlap-Intensity Critical Mutual Exclusion: This is the most generic case of the mutual exclusion problem, where the actors are chosen not only based on the new and old overlaps but also the intensity of the overlaps. In other words, every overlap beyond a threshold is deemed as undesirable, and the weight of the function depends on the number of times the overlap occurs for a particular region (intensity of overlap).

Consider the same fire extinguisher example mentioned above with heat sensors for sensing and sprinklers as actors. Let the appropriate level of action be described by one actor (sprinkler) acting on a fire event in a region. Consider the case, when the acting ranges of sprinklers overlap. In such a case, the regions where the overlap occur will result in flood. If the overlap occurs multiple times, the flooding will be severe in those regions depending on the intensity of overlap for those regions. Here, it is likely that the weights to the benefit function are different for varying intensity of overlap, and they are defined based on the intensity of overlap. In this scenario, apart from maximizing the non-overlapped region, it is necessary that the overall benefit function is optimized.

Definition 4: The intensity-based mutual exclusion problem is to determine the minimum set of actors, M, that *maximizes* the non-overlapping and *minimizes* the total overlapping regions based on the intensity, where the individual benefit function of each actor is given by:

$$VM_i = r_i - \sum_{I_i} (\beta_{I_i} \times I_i \times (n_i, o_i)) \tag{4}$$

Here, β_{I_i} is the weighting factor that represents the cost incurred in having an overlap with intensity I_i in the event region.

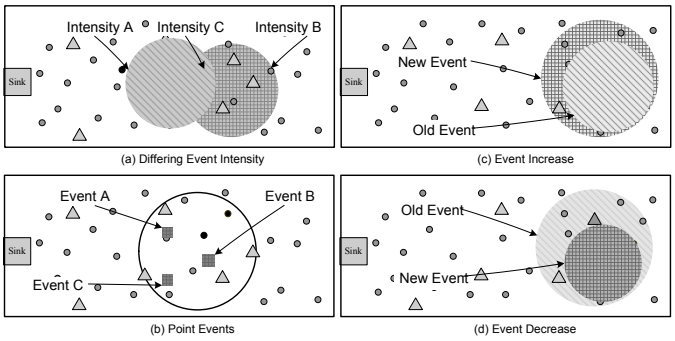


Fig. 3. Challenges Illustration

C. Challenges

So far, we have discussed the mutual exclusion in the context of regional events requiring only one round of execution with no event dynamics. However, for other types of applications, we need to address the following challenges while addressing the problem:

- *Differing Event Intensity*: In some applications, event intensity may not be the same across the entire event region (see Figure 3 (a)). If the event intensity is different

in two non-overlapping regions, then the actions should be performed to reflect the desired event intensity. In such a case, it may require multiple rounds of action in certain regions depending on the event intensity. Note that this is *orthogonal* to the overlap-intensity critical mutual exclusion, where the intensity of action required is the same but the penalty incurred with overlaps is different. Consider the fire extinguisher application as described before, where the fire intensity is different in different regions. Here, it is necessary that the actions performed match the event intensity in that region.

- *Point/Multi-point Events*: Thus far, we have only described the mutual exclusion problem for a regional event. The problem for (multi-)point events is quite different. In this case, the definition reduces to minimizing the number of actors that address all the point events in the network (see Figure 3 (b)). For example, consider an intruder-detection application, where the sensors are image sensors and actors are poisonous gas actors. In this case, the event is occurrence of intruders, which are point events. Thus, the mutual exclusion problem reduces to minimizing the actors that cover all these point events based on the benefit function.
- *Event Dynamics*: This challenge occurs when there are multiple rounds of operation to address an event. When the event area has increased in size, it is necessary that mutual exclusion be provided to the new event area as shown in Figure 3 (c). In the fire extinguisher application, if suppose there is a fire in a region initially and if the fire has spread to a larger region with time, the mutual exclusion approach should adapt itself to address the problem for the new event area. Similarly, when an event that had occurred in a certain region has now decreased in size, it is necessary that the mutual exclusion approach is only executed in the current event region (see 3 (d)). In the fire extinguisher example, if suppose there is a fire in a region and if the intensity of fire is maximum at the center of the region, it is likely that the fire is first extinguished in the peripheral regions of the event area. After this time, mutual exclusion is required only in the new event area.

D. Goals

Further, any mutual exclusion approach should also be efficient with respect to the following important goals:

- *Overhead*: The overhead in providing mutual exclusion should be small. In other words, the mutual exclusion approach should be able to address the problem by having low communication overhead, in terms of the total number of bytes transmitted.
- *Correctness*: Another critical goal is that any approach that provides mutual exclusion is able to cover the entire event region (correctness). Correctness is defined by the percentage of area covered by the actor cover set in comparison with the total event region.

III. Centralized Approach

In this section, we present a greedy centralized approach for providing mutual exclusion that is efficient for all four types of mutual exclusion. We make the following simplifying assumptions in order to make the problem more tractable:

- *Network Model*: We consider the case, where sensors and actors are both *static*, and are randomly distributed[1] in sensor/actor field.
- *Location Information*: We assume that sensors and actors can determine their location through localization algorithms [3].
- *Sensing, Acting and Communication Ranges*: We assume that sensing range (R_s), the communication range for sensors (T_s) and actors (T_a) and acting range (R_a) to be the same[2].
- *Routing Model*: We assume that there is an underlying reliable routing protocol for delivering directives and gathering responses and delivering notifications to any sensor or actor [4], [5].

A. Centralized Approach

In this section, we present a greedy, centralized algorithm to construct the actor cover set for an event region to alleviate the mutual exclusion problem. The selection criteria at each iteration of the greedy approach is based on the benefit function of actors.

1) Mechanism: To describe briefly, the greedy algorithm works by selecting, at each stage, the actor with the maximum benefit function, where the benefit function is defined by the type of mutual exclusion as explained in Section II. The selected actor is added to the already selected set of actors at that stage. The benefit function is updated for all the actors that have overlapping acting regions based on the new values of R_M, r_i, n_i and o_i as defined in Section II. The algorithm terminates when the selected set of actors cover the complete event region.

We will now describe the approach in detail using Figure 4. Let us assume that M is the set of actors already selected for inclusion in the actor cover set. Initially, M is an empty set. The algorithm starts by including an arbitrary actor that completely lies in the event region (maximizes the benefit function). At each stage, the greedy algorithm selects an actor, MAX_ACTOR that has the maximum benefit function, VM_{MAX_ACTOR} (lines 15-18 in Figure 4). The actor is added to the actor cover set, M, and the R_M value is updated by adding the non-overlapped region of this actor, $MAX\_BENEFIT$. Also, the set of actors that remain in consideration, ω is updated by removing this actor (lines 20-22 of Figure 4). Any actor, i, that has an overlapping acting range with this actor, updates the non-overlapping region, r_i, new overlapped region, n_i, and the old overlapped region, o_i (lines

[1]We assume that the network is connected through these sensors and actors.

[2]Note that this assumption is not central to our approach and can be easily extended to different sensing, acting and communication ranges. We do not present the details of this aspect due to lack of space.

Input

a_1,a_2,...,a_n: Actors in the WSAN, S: Sink,
C: Command sent to region of interest,
R_C: Region of interest for the command,
Ω: Set of actors whose acting region intersects with R_C,
R_i: Acting range of actor i, and
T_i: Communication range of node i.

Output

The minimal actor cover set for command C that covers R_C

Compute the minimal actor cover for region R_C

M: The set of actors selected as part of actor cover at any given stage
R_M: Region covered by M
$\omega = \Omega$
while ($R_C \not\subseteq R_M$)
    $MAX\_BENEFIT$ = 0 (or) -$\alpha \cdot \pi \cdot R_i^2$ (or) -$\beta \cdot \pi \cdot R_i^2$ (or) -$\beta_{I_i} \cdot I_i \cdot \pi \cdot R_i^2$
    for each $a_i \in \omega$
        VM_i = Benefit function of a_i
        if ($VM_i > MAX\_BENEFIT$)
            $MAX\_BENEFIT = r_i$
            $MAX\_ACTOR = i$
        end if;
    end for;
    $M = M \bigcup MAX\_ACTOR$
    $R_M = R_M + MAX\_BENEFIT$
    $\omega = \omega - MAX\_ACTOR$
    for each ($a_i \in \omega$) and (($R_i \bigcap R_{MAX\_ACTOR}) \neq 0$)
        Update r_i, o_i, n_i
    end for;
end while;
return M

Fig. 4. Centralized Algorithm pseudo-code

23-25 in Figure 4). The algorithm will eventually terminate when the event region is contained in R_M (line 11 in Figure 4).

2) Optimality of the Approach: It has been shown in [6], [7], [8] that the computation of the optimal cover for the generic set cover problem for a given network graph is NP-hard. The authors in [7], [8] also established the upper bound of the competitive ratio of a greedy approach to the optimal solution to be directly proportional to the radius of the network and logarithmically proportional to the number of nodes. These results can be extended to the greedy centralized approach for the mutual exclusion problem. The competitive ratio reduces to an upper bound of $O(r_C \times log(\Omega))$ to the optimal, where r_C represents the radius of the event region, and Ω is the set of actors in the event region.

IV. Distributed Approach

In this section, we present a distributed and fully localized approach called the Neighborhood Back-off (NB) approach and show that it addresses the challenges identified in Section II. We first provide a high-level overview of the NB approach and present the actions performed by the sink, sensors and actors.

A. Overview

In this section, we identify the key design components of the NB approach. We first identify the notion of a dependency region for a sensor or an actor (which we refer by the common term *entity*). For a given *entity*, D_x, the maximum region within which another entity can have an impact on its *execution*[3] range is defined to be the *dependency region*. The dependency region of a sensor is given by the region with radius equal to the sum of sensing and acting range ($Sensing\ Range + Acting\ Range$), while that of an actor is given by the region with radius as twice the acting range ($2 \cdot ActingRange$).

Once the dependency regions for all the actors in the event region have been determined, the basic operations in the NB approach are as follows:

- The determination of initial benefit function for each actor based on the directives issued by the sensors to the actors in its dependency region.
- The emulation of the greedy centralized strategy at each actor by waiting for an amount of time that is inversely proportional to benefit function of that actor. If the benefit function of a particular actor is large (close to its acting range) the waiting time will be relatively small, whereas, when the benefit function is very small the waiting time will be relatively large.
- The updating of the benefit functions (and hence the waiting time for execution) for all actors within the dependency region of an actor that has acted on a specific directive.

The approach trades-off the communication overhead in the construction of a centralized greedy actor-cover set by appropriately waiting for suitable amount of time at each actor that is inversely proportional to the benefit function of the actor. When there is no strict delay bound (δ), the distributed approach will construct an actor-cover set that is equivalent to the greedy centralized approach. This is because at each stage only actors with the best benefit function will be chosen to act in the region, where the benefit function is captured by waiting for the appropriate amount of time. If the benefit function is large, the waiting time will be relatively small ensuring that the actor is selected as part of the actor cover set. On the other hand, when the benefit function is small, the actors wait for a longer amount of time. Thus, the approach allows for a distributed realization of the centralized strategy and allows for automatic updates to benefit functions of all entities within each dependency region.

B. The Neighborhood Back-off Approach

As mentioned in the overview, the neighborhood back-off (NB) approach is a distributed realization of the centralized approach, where the benefit function is captured by a waiting time after which an actor can act. We now describe the con-

[3]We refer to sensing range for sensors and acting range of actors with the common term execution range.

struction of dependency regions, and the operations performed at each entity, i.e., sensors and actors.

1) Construction of Dependency Regions: We assume as part of the initial set up of the network that there is an underlying 2-hop neighbor discovery mechanism so that each actor can advertise their node locations to its 2-hop region. In [9], the authors discuss an approach for local broadcast of beacons in order to transmit location information of the node as a basic step to ensure sensor coverage. This technique can be extended to a 2-hop neighborhood in order to transmit the location information. The neighbor discovery mechanism will allow each sensor and actor in the network to learn about all the other actors within the dependency region. This is a one-time discovery process to determine the set of actors within the dependency region of a sensor or an actor.

2) Operations at the Sensors: When a sensor detects the presence of an event, it first reports the sensed information to the sink. In the NB approach, every sensor in the event region also constructs a shortest path tree [10] to every actor within its dependency region[4] and uses a hop-to-hop reliability mechanism in the case when there are losses. Once the two-hop routing tree is constructed, it issues a request directive ($REQUEST()$) to all the actors in its dependency region.

Each sensor, S_i, piggybacks the following information along with the request directive: $REQUEST(Dir\_id, X_{S_i}, Y_{S_i})$. Here, Dir_id refers to the directive identifier based on the query sent by the sink, and (X_{S_i}, Y_{S_i}) indicates the location of the current sensor. The $REQUEST()$ directive is similar to command directive issued by the sink and is required to act on the environment. Additionally, the sensors may also choose to send a $CANCEL()$ directive with the same fields, if a request needs to be annulled. We describe this later in this section in the context of an event decrease. The actors wait for a time proportional to twice the transmission delay, T_d, to ensure reception of all $REQUEST()$ and $CANCEL()$ directives within its dependency region, prior to initiating an action.

3) Operations at the Actors: Once the event has been reported to the sink, the sink issues a command directive to the actors in the network[5]. An actor can determine the event region in its acting range based on the $REQUEST()$ directive received from the sensors within its dependency region and the sensing range. Also, every actor in the event region constructs a shortest path tree [10] to all other actors within its dependency region as mentioned before. This allows the actors to communicate with any or all of the actors within its dependency region. We now explain the operations performed at the actor for a single directive using Figure 5.

- When an actor has received a $REQUEST()$ directive from a sensor, it first determines the additional event area covered by the sensor and adds that region to already existing event area as indicated in lines 13-14 in Figure 5. Further, it determines the intersection of this area with the acting range and determines this to be the virtual metric. This virtual metric is used to determine the wait time for this actor, which is scaled appropriately to have an upper bound of δ (lines 15-16 in Figure 5). If the wait time is determined to be less than or equal to zero, which implies that benefit of this actor is high enough to act immediately, a $NOTIFY()$ transmission is scheduled to indicate that this actor will be executing the directive shortly. If on the other hand, the wait function is greater than zero, the actor remains in $Wait()$ state.
- If the message received is a $NOTIFY()$ message, the actor first checks if it is already scheduled to act by checking the $Flag()$. If the $Flag()$ is set, which indicates that the current actor is also scheduled to act, the relative benefit function of both actors is compared and if the current actor has a higher benefit function, or, a lower node identifier with the same benefit function, it returns to $Transmit()$ state to execute the directive (lines 26-29 in Figure 5). If on the other hand, the $Flag()$ is not set, or if the benefit function of the current actor is lower, or if the node identifier is higher with the same benefit function, the actor recomputes its benefit function and waiting time inferring that the other actor is scheduled to act. The $Flag()$ is set to $FALSE$ indicating that the actor is no longer in the execution phase and remains in $Wait()$ state if the wait time is greater than zero (lines 30-36 in Figure 5).
- If the wait time has reached zero, the actor goes into the transmit phase, where the actor will transmit the $NOTIFY()$ message prior to execution. The actor will first send the $NOTIFY()$ message to all actors in its dependency region using the routing structure already constructed and waits for a time corresponding to two-hop delay, $2 \cdot T_d$. If in the meantime, a $NOTIFY()$ message is received, the procedure mentioned in lines 26-36 in Figure 5 is followed as explained above.

In this fashion, an actor can update its benefit function based on the $NOTIFY()$ messages received prior to its wait time. This update process is similar to that in the centralized scheme. However, unlike in the greedy centralized approach where only one actor is selected in each iteration, in the NB approach, it is possible that several actors are selected to act in one iteration (or at the same time) as long as the actors are not within the dependency region of each other. This allows for faster completion of the actor selection albeit at the cost of a slight increase in the number of actors selected as part of the actor cover, as we show later in Section V.

C. Mechanisms for Addressing Challenges

We will now show how the proposed scheme can address the different challenges described in Section II.

[4]The nodes in the periphery of the event region will construct the routing structure for the actors (within the dependency region) that are part of the event region.

[5]The event region estimated by the sink based on the sensor responses may not be accurate as it will require that all the responses reported by the sensors arrive at around the same time.

Default State:
0 *Wait(A_i)*
Variables:
1 A_i: Actor Node id, $D(i)$: Directive ID,
2 $S_1 \ldots S_k$: Array of Requests from Sensors at A_i,
3 $R_{A_i}(k)$: Region enclosed by requests from k sensors at A_i,
4 $VM(A_i)$: Benefit fn. of A_i, $VM(A_i)_{init}$: Initial Benefit fn.,
5 $WT(A_i)$: Waiting time before execution for A_i,
6 $Ar(S_i)$: Maximum area covered by sensor S_i,
7 $Ar(A_i)$: Maximum area covered by actor A_i,
8 $T_{curr}(A_i)$: Current Time at actor, A_i,
9 $T_{init}(A_i)$: Time at actor, A_i when first request was received,
10 $Flag(A_i)$: Flag at A_i to denote $NOTIFY()$ sent or not,
11 $T_{note}(A_i)$: Time at actor, A_i when $NOTIFY()$ was sent,
Receive(A_i)
12 If $(MSG_{RX} == (REQUEST() or CANCEL())$
13 If $(MSG_{RX} == REQUEST(Dir\_id, X_{S_j}, Y_{S_j})$
14 $R_{A_i}(k+1) = R_{A_i}(k) \bigcup Ar(S_j)$
15 $VM(A_i) = R_{A_i}(k+1) \bigcap Ar(A_i)$
16 $WT(A_i) = \delta \cdot \frac{(Ar(A_i) - VM(A_i))}{(Ar(A_i) - VM(A_i)_{init})}$ + $T_{init}(A_i)$ - $T_{curr}(A_i)$
17 If $WT(A_i) \leq 0$
18 *Transmit(A_i)*
19 If $(MSG_{RX} == CANCEL(Dir\_id, X_{S_k}, Y_{S_k})$
20 $R_{A_i}(k-1) = Ar(S_1) \bigcup Ar(S_2) \bigcup \ldots \bigcup Ar(S_{k-1})$
21 $VM(A_i) = R_{A_i}(k-1) \bigcap Ar(A_i)$
22 If $VM(A_i) > 0$
23 $WT(A_i) = \delta \cdot \frac{(Ar(A_i) - VM(A_i))}{(Ar(A_i) - VM(A_i)_{init})}$ + $T_{init}(A_i)$ - $T_{curr}(A_i)$
24 Else If $VM(A_i) \leq 0$
25 $WT(A_i)$ = $NULL$
26 If $(MSG_{RX} == NOTIFY(Dir\_id, VM(A_j), X_{A_j}, Y_{A_j})$
27 If $(Flag(A_i) == TRUE)$
28 If $(VM(A_i) > VM(A_j)) or ((VM(A_i) == VM(A_j)) and (i < j))$
29 Return to *Transmit(A_i)*
30 Else
31 Update r_i, o_i, n_i
32 Update $VM(A_i)$ based on benefit function
33 $WT(A_i)$ = $\delta \cdot \frac{(Ar(A_i) - VM(A_i))}{(Ar(A_i) - VM(A_i)_{init})}$ + $T_{init}(A_i)$ - $T_{curr}(A_i)$
34 $Flag(A_i) == FALSE$
35 If $WT(A_i) \leq 0$ then *Transmit(A_i)*
36 Return to *Wait(A_i)*
Transmit(A_i)
37 $Flag(A_i) = TRUE$
38 Send $NOTIFY(Dir\_id, VM(A_i), X_{A_i}, Y_{A_i})$ to 2-hop actors
39 $T_0 = T_{curr}(A_i)$
40 Do
41 If (MSG_{RX})
42 *Receive(A_i)*
43 While !($T_0 == 2 \cdot T_d$)
44 *Execute(Dir_id)*
Wait(A_i)
45 $VM(A_i)_{init} = 0$ (or) $-\alpha \cdot Ar(A_i)$ (or) $-\beta \cdot Ar(A_i)$ (or) $-\beta_{I_i} \cdot I_i \cdot Ar(A_i)$
46 Do
47 If (MSG_{RX})
48 *Receive(A_i)*
49 While !($WT(A_i) == 0$)
50 *Transmit(A_i)*
51 $Flag(A_i) = FALSE$
52 $WT(A_i)$ = $NULL$

Fig. 5. The NB Approach at Each Actor for One Directive

1) Handling Varying Event Intensities: This challenge pertains to adapting the actor cover algorithm based on the difference in the intensity across the event region.

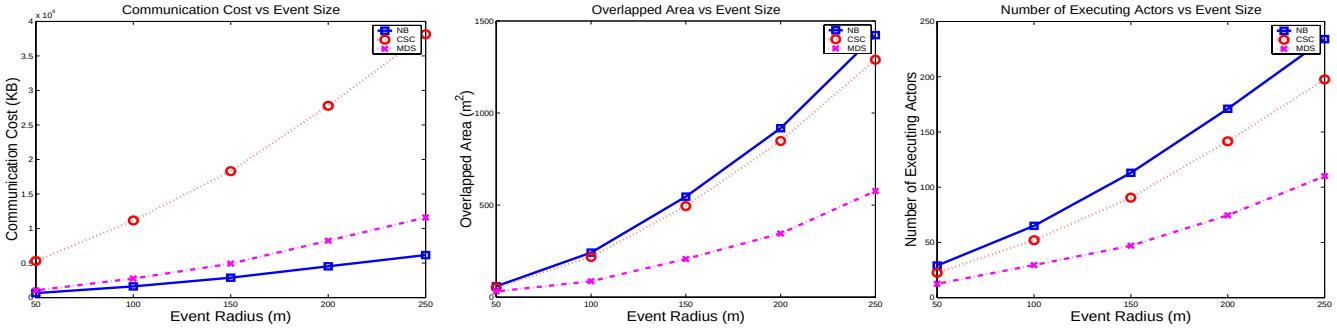
In the NB approach, the intensity is piggybacked along with the $REQUEST()$ directive sent by each sensor. Each actor would accommodate the intensity metric in the calculation of the benefit function, $VM()$ and wait time, $WT()$. This information would also be sent in the $NOTIFY()$ message sent by the actors. Thus, an actor that is scheduled to act based on the waiting time, will act multiple times corresponding to the intensity desired. The actors within its dependency region, would also decrease their $VM()$ by a factor corresponding to both intensity and the benefit function of the actor that has acted.

2) Handling Point Events: Thus far, we have only discussed the NB approach in the context of regional events. This challenge corresponds to the case where the events are confined to points in the network. In such a case, the NB approach should still be able to select a minimum set of actors that covers all the point events without any overlap.

In the NB approach, the sensors send the point event location along with the $REQUEST()$ directive. Thus, it is possible for the actors to determine how many of these point events fall within its acting range. Based on this information, the benefit function can be computed as the sum total of all point events that an actor can cover, and wait times are chosen accordingly. When an actor receives a $NOTIFY()$ message, the benefit function, $VM()$, will be decreased by a quantity proportional to the point events covered by the other actor. This will accordingly increase the wait time proportional to the number of point events not yet covered. In this fashion, it is still possible to maintain the low overhead operation achievable by using a distributed approach, while maintaining the basic operations of a greedy approach.

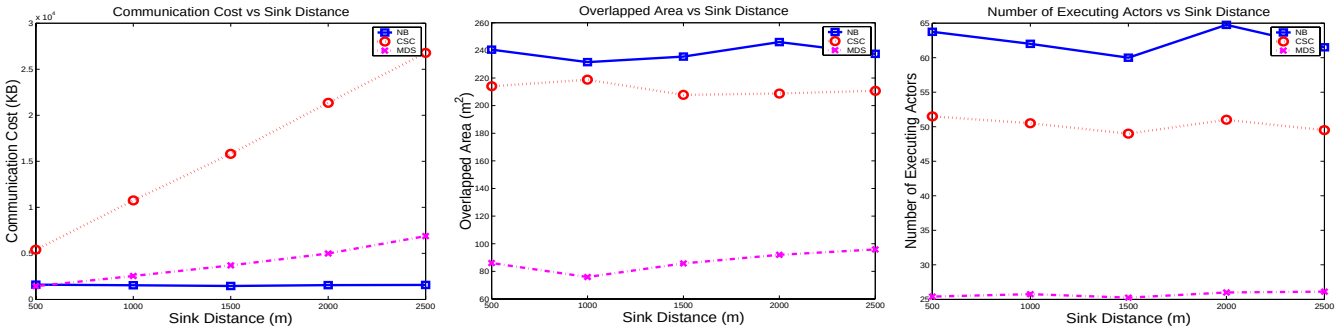
3) Handling Event Dynamics: This challenge pertains to the increase or decrease in the event area during the course of multiple rounds of operation to address an event.

In the NB approach, event increase can be readily accommodated if the sensors that are part of the new event region send $REQUEST()$ directives to actors within its dependency region. This will force the actors that are beyond the old event region to participate in the actor cover set selection procedure ensuring that the entire region is covered. To address event decrease, sensors that are not part of the event region anymore direct actors waiting to act in the region to stop, by using a $CANCEL()$ control message. Once an actor receives a $CANCEL()$ directive, it removes the non-overlapping area corresponding to this sensor in the calculation of the benefit function and computes the new wait time. If the benefit function is zero, it implies that the actor need not act on this



(a) Total Traffic (b) Overlapped Area (c) Resource Cost

Fig. 6. Performance under Different Event Size



(a) Total Traffic (b) Overlapped Area (c) Resource Cost

Fig. 7. Performance under Different Distance from the Sink

directive and hence the wait time and benefit function is set to $NULL$ values. This procedure is shown in lines 19-25 in Figure 5.

V. Performance Evaluation

This section evaluates the performance of the NB approach with two strategies: Centralized Set Cover (CSC), which is a realization of the centralized approach described in Section III, and Minimum Dominating Set (MDS) [4]. In the CSC approach, once an event has been detected, the sink sends explicit commands to all actors within the event region to know their location information before constructing the set cover in the greedy fashion described in Section III. The performance metrics considered include, the total communication cost per event processing (in KB), overlapped action area (in m^2), correctness (% of event area covered by actions), and number of actors that have performed the action in the event region (resource cost). For all simulations, the benefit function used corresponds to the first type of mutual exclusion[6].

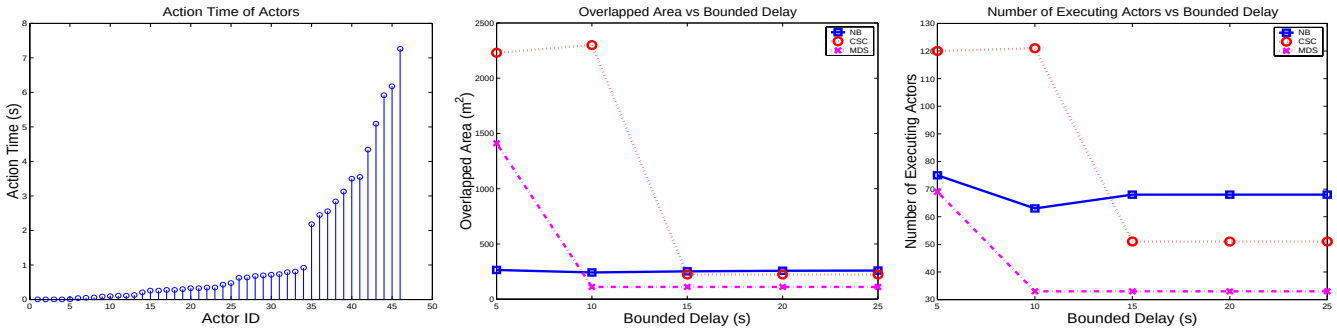
We use a custom built, event-driven simulator written in C. For all simulations, unless otherwise stated, 2000 sensors and 2000 actors are randomly placed on a 3000m$\times$3000m square area to ensure connectivity. The sensing and communication range of sensors is set to be 30m, and the acting and communication range of actors is 30m. The default event radius is set to be 100m, and we vary its value from 100m to 500m. The default distance from the event center to the sink is set to be 1500m, and we also evaluate the performance with distances ranging from 500m to 2500m. The bounded delay is set to be 10 seconds, and the packet size is 1 KB. The other parameters are described along with the results.

In terms of correctness of operation, both NB and CSC approaches can guarantee 100% correctness, and ensure complete coverage of the event area. However, the MDS approach cannot guarantee this property. *In all simulations, we have observed that MDS can only achieve about 70% correctness, implying that about 30% of the event area has not been covered.*

A. Varying the Event Area Size

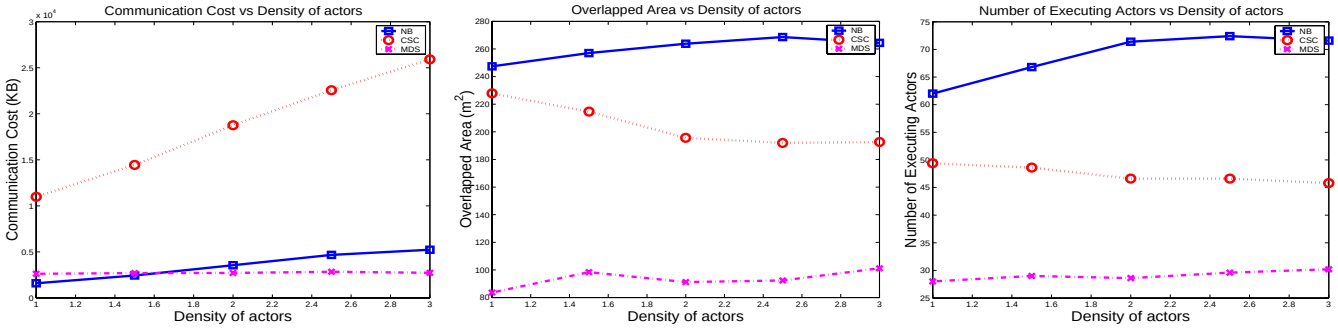
Figures 6 shows the performance results of the three approaches under varying event area size. The NB approach achieves the best performance in terms of communication cost. As shown in Figure 6(a), with increasing event area size, the traffic per event for all three approaches increases. For CSC and MDS approaches, this is mainly because of the super-linear increase in the number of sensors and actors in the event area, which consequently results in a super-linear increase in the data reported to the sink and commands issued by the sink. Since the NB approach is fully localized, each node receives notifications *only* within its dependency region, and consequently the increase is gradual. Although MDS achieves a lower communication cost than CSC, it is only at the expense of correctness. Figures 6(b) and (c) show the results of

[6]We have observed that the trends are similar for other types of mutual exclusion.



(a) Action Times of Acted Actors (δ=10s) (b) Overlapped Area (c) Resource Cost

Fig. 8. Performance under Different Bounded Delay



(a) Total Traffic (b) Overlapped Area (c) Resource Cost

Fig. 9. Performance under Different Actor Densities

overlapped action areas and number of actors that performed the action. The NB approach is only slightly worse than CSC, while they both achieve 100% correctness.

B. Varying the Distance from the Sink to the Event Center

Figure 7 shows the performance of the three approaches for varying sink-to-event distances. We observe similar trends as in the previous set of results: NB has the best performance in terms of communication cost, and has almost similar performance in terms of the overlapped action area and number of actors. A key observation in Figure 7 (a) is that the communication cost of the NB approach does not increase with increasing sink-to-event distance. This is due to the localized operation in NB, where there is minimal coordination between sensors, actors and the sink. For approaches such as CSC and MDS, the communication cost increases with increasing sink-to-event distance due to centralized computation of the actor cover.

C. Varying the Delay Bound

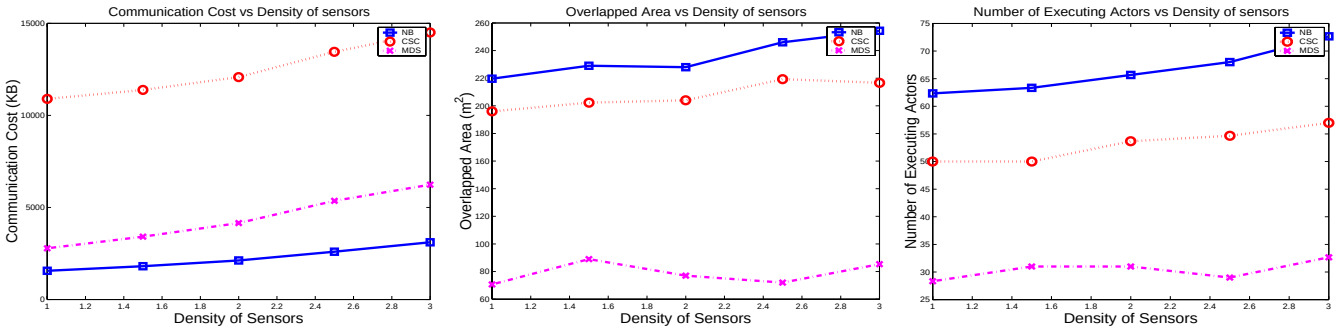
The effect of increasing event processing delay, δ, on the performance is shown in Figure 8. We vary the δ values from $5s$ to $25s$. In Figure 8 (a), we show the action times for the actors that performed in the NB approach when the delay bound is $10s$. From the results, we see that the actors execute the command at different times in the NB approach due to the back-off mechanism. In Figure 8 (b) and (c), we show the results of overlapped area and action resources. We observe that when the delay bounds has little effect on the NB approach as opposed to the other two schemes. The NB approach is a fully localized approach, and so even a nominal delay bound can be satisfied. However, for centralized approaches such as the CSC and the MDS, the resource cost and overlapped area can be very large when the delay bound is small. This is because centralized approaches require a longer time to wait for sensor and actor responses at the sink, based on which the actor cover is computed.

D. Varying the Density of Actors

When the density of actors increases from 1x2000 actors to 3x2000 actors as shown in Figures 9, both NB and CSC incur a slightly larger communication overhead due to the additional overhead to communicate with the extra actors introduced, while the overhead of the MDS is almost constant. CSC approach achieves better performance with respect to the overlapped area and the number of actors (resource cost) for increasing values of density. The reason is intuitive as the sink can determine a better actor cover when there are more actors available. For NB, they perform slightly worse because of the distributed operations as mentioned in Section IV.

E. Varying the Density of Sensors

When the density of sensors increases from 1x2000 sensors to 3x2000, all three approaches have a slightly larger overhead due to the marginal increase in communication cost to report sensor data as shown in Figure 10. We observe a marginal



(a) Total Traffic (b) Overlapped Area (c) Resource Cost

Fig. 10. Performance under Different Sensor Densities

increase for all three approaches in the amount of overlapped area and resource cost with increasing sensor density. For high sensor density, the event area increases marginally, resulting in a slightly larger number of actors executing a particular directive.

VI. Related Work

A. Sensor and Actor Networks

In [11], the authors propose centralized and distributed solutions for determining the minimum connected sensor cover in order to reduce the overall energy consumption of a pure wireless sensor network (WSN). This problem is different to mutual exclusion problem in WSANs, where there is no need for the actors to be connected. Also, there different types of mutual exclusion identified in the context of WSANs, are not required in a pure WSN environment. Further [11], does not try to minimize delay and hence does not incorporate any delay constraints in the design of their approaches. Also, [11] does not address any of the challenges that are unique to a WSAN environment. The authors in [12] have considered a different problem in WSANs pertaining to actor-actor coordination, where the goal is to determine the set of actors to cover an event region when the actors have different power levels. Here, the actor set is optimized to reduce the overall power consumption, which is different from the problem of mutual exclusion.

B. Mutual Exclusion in Ad hoc Networks

The distributed mutual exclusion problem has been identified in the context of ad hoc networks in the context of assignment of channels and shared resources [2], [13], [14]. However, these works do *not* conform to the definition of mutual exclusion in the context of WSANs.

VII. Conclusions

In this paper, we have identified the problem of mutual exclusion in the context of a wireless sensor and actor network, and described the different types and the associated challenges. We have proposed a greedy centralized approach that is near optimal. We have also proposed a localized and fully distributed approach, called the NB approach, that addresses the problem and the associated challenges effectively. Through simulations, we compare the performance of the NB approach with the centralized approach and a baseline strategy.

References

[1] I. H. Kasimoglu I. F. Akyildiz, "Wireless Sensor and Actor Networks: Research Challenges," in *Elsevier Ad Hoc Networks Journal*, 2004.

[2] G. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems: Concepts and Design*, Addison-Wesley, 2001.

[3] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-Less Low Cost Outdoor Localization for Very Small Devices," in *IEEE Personal Communications, Special Issue on Smart Spaces and Environments*, Oct. 2000, pp. 28–34.

[4] S. J. Park, R. Vedantham, R. Sivakumar, and I. F. Akyildiz, "A Scalable Approach for Reliable Downstream Data Delivery in Wireless Sensor Networks," in *Proceedings of the ACM symposium on Mobile Ad Hoc Networking and Computing (ACM MOBIHOC)*, May 2004.

[5] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyilidiz, "ESRT: Event-to-Sink Reliable Transport in wireless sensor networks," in *Proceedings of the ACM symposium on Mobile Ad Hoc Networking and Computing (ACM MOBIHOC)*, June 2003.

[6] R. D. Carr, S. Doddi, G. Konjevod, and M. V. Marathe, "On the Red-Blue Set Cover Problem," in *Symposium on Discrete Algorithms*, 2000, pp. 345–353.

[7] U. Feige, "A Threshold of ln n for Approximating Set Cover," in *28th ACM Symposium on Theory of Computing*, Philadelphia, USA, May 1996, pp. 314–318.

[8] D. Johnson, "Approximation Algorithms for Combinatorial Problems," in *Journal of Computer and System Sciences*, 1974.

[9] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage Problems in Wireless Ad-hoc Sensor Networks," in *Proceedings of the IEEE conference on Computer Communications (IEEE INFOCOM)*, Apr. 2001.

[10] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "Energy-efficient broadcast and multicast trees in wireless networks," *Mobile Networks and Applications*, vol. 7, no. 6, 2002.

[11] H. Gupta, S. Das, and Q. Gu, "Connected Sensor Cover: SelfOrganization of Sensor Networks for Efficient Query Execution," in *Proceedings of the ACM symposium on Mobile Ad Hoc Networking and Computing (ACM MOBIHOC)*, June 2003.

[12] T. Melodia, D. Pompili, V. C. Gungor, and I. F. Akyildiz, "A Distributed Coordination Framework for Wireless Sensor and Actor Networks," in *Proceedings of the ACM symposium on Mobile Ad Hoc Networking and Computing (ACM MOBIHOC)*, May 2005.

[13] I. Katzela and M. Nagshineh, "Channel Assignment Schemes for Cellular Mobile Telcommunication Systems: A Comprehensive Survey," in *IEEE Personal Communications*, June 1996, pp. 10–31.

[14] J. E. Walter, J. Welch, and N. Vaidya, "A Mutual Exclusion Algorithm for Ad hoc Mobile Networks," in *ACM and Wireless Networks Journal Special Issue on Dialm papers*, 2001.