Mobile Network Management and Robust Spatial Retreats via Network Dynamics

Ke Ma, Yanyong Zhang, Wade Trappe
Wireless Information Network Laboratory (WINLAB)
Rutgers University, 73 Brett Rd., Piscataway, NJ 08854.

***Abstract*— The mobility provided by mobile ad hoc and sensor networks will facilitate new mobility-oriented services. Recent work has demonstrated that, for many issues, mobility is advantageous to network operations. This paper proposes that the need for mobility may be captured by formulating the movement of nodes as a classical dynamical system. Motivated by classical mechanics, we propose the notion of network dynamics, where the position and movement of mobile devices evolve according to forces arising from system potential functions that capture the operational goals of the network. We argue that, in the context of moving communicating nodes, the equations of motion should be formulated as a steepest descent minimization of the system potential energy. Further, since global information is not practical in sensor networks, we introduce distributed algorithms that yield more practical implementations of network dynamics. The resulting algorithms are generic, and may be applied to produce balanced network configurations for different initial network deployments. As a second application of network dynamics, we examine the problem of adapting a mobile sensor network to the threat of a jammer. We show that the combination of spatial escape strategies with network dynamics prevents network partitioning that might arise from a mobile jammer.**

I. Introduction

Wireless networks are being used for scenarios where a fixed communication infrastructure cannot be suitably employed. Many examples of such networks are gaining popularity, ranging from tactical networks for military communications, to rapidly-deployable mobile ad hoc networks for emergency management, to mobile sensor networks [1–3]. Although these networks naturally involve wireless communications, they are differentiated from many other wireless networks by the inherent mobility of the underlying communication infrastructure. By comparison, in a cellular system, the base station is in a fixed location and mobile devices communicate with this fixed infrastructure. Similarly, sensor networks usually involve static sensors using multihop routing to deliver data [4].

Incorporating mobility into the underlying communication infrastructure allows for new types of mobility-oriented network services. The network becomes a dynamic entity capable of adjusting to its environment [1–3]. This vision of mobile networks requires that we formulate laws governing why and how nodes should be mobile. There may be several objectives that these laws are meant to achieve. For example, a mobile ad hoc sensor network may consist of devices that are initially clustered near each other, and mobility may be used to achieve uniform coverage over a particular region. Or, as another example, a mobile ad hoc network may be subjected to a jamming attack, and the underlying mobility laws may direct the devices to evacuate the jammed area [5].

In this paper, we formulate the laws governing the motion of mobile devices as a dynamical system involving forces acting upon network entities. These forces induce mobility, and adjust the network system as a whole towards a state of minimal conflict, or minimal potential energy. The description of these forces may take many forms, ranging from repulsive forces that disperse tightly clustered radio devices, to attractive forces meant to cause sensor devices to gather near areas of high importance. We begin the discussion of this model, which we call network dynamics, in Section II. We formulate the relationship between network devices as forces arising from a potential energy, and apply Newton's laws of motion to govern the dynamics of the networked system. We show that network dynamics is better formulated as a minimization of a potential energy function via steepest descent. In Section III, we present distributed algorithms that may be used to implement network dynamics in practice. We then examine two applications of distributed network dynamics. The first application, presented in Section IV, focuses on using network dynamics to control the coverage distribution of sensors over a region. In Section V, we apply network dynamics to achieve a robust spatial retreat strategy that maintains desirable connectivity in the presence of a jamming attack. We conclude in Section VI.

II. Network Dynamics

A mobile ad hoc network is a collection of mobile devices, and may be viewed as a dynamical system– the positions and speeds of devices change with time. The dynamics of classical mechanics systems are described via underlying laws of motion and laws of force between objects [6]. The concept of forces, the corresponding notion of potential energy, and the laws of motion may be used to manage the movement of a mobile sensor network. Appropriately defining potential functions and forces, yields a general framework that allows one to optimally use mobility to govern a network's operations.

A. Classical Dynamics and Mobile Networks

Throughout this paper, we shall look at a mobile sensor network as a dynamical system of N devices subject to the laws of classical mechanics. Each device will be able to communicate with its neighbors through some wireless communication protocol. Since the devices are mobile, we will associate with each device j a position vector $\mathbf{p}_j$ and a momentum vector $\mathbf{q}_j$. We will, for simplicity, assume that all devices are located in two-dimensions and that, for each device, both $\mathbf{p}_j$ and $\mathbf{q}_j$ are two-dimensional vectors. For the purpose of our discussion, since we are looking at the system as a mechanical system, we will arbitrarily assign each network device a mass of 1, thus momentum and velocity are equivalent.

N-body dynamical systems appear commonly in physics in order to model classical mechanical systems, such as from gravitational modeling or in electrostatics. In these problems, the dynamical relationship between position and momentum of these N bodies evolves based upon Newton's second law, which describes the motion of a body in the presence of a field

0-7803-9466-6/05/$20.00 ©2005 IEEE

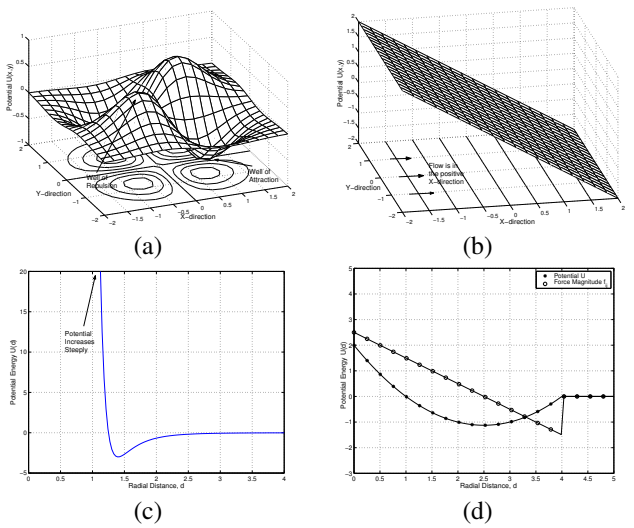


Fig. 1. (a) The potential function U indicating basins of attraction and repulsion, (b) the potential function U encouraging a flow in the positive x direction, (c) the Lennard-Jones potential, and (d) the potential and magnitude of the linear-force model.

of force. The forces acting upon a conservative dynamical system arise as the negative gradient of the potential energy function U. This potential function may be comprised of two components: *external* and *internal*. External potentials arise from externally applied forces, while internal potentials correspond to the attractive or repulsive forces between bodies of the system. Typically, the internal potentials are restricted to two-body interactions, such as the gravitational pull between two objects.

We may collectively refer to the position vector of each of the N bodies by a $2N$-dimensional position vector $\mathbf{p} = [\mathbf{p}_1, \cdots, \mathbf{p}_N]$, and similarly for the momentum vector. Hence, the potential energy may be viewed as $U(\mathbf{p}) = \sum_j U_{ext}(\mathbf{p}_j) + \sum_i \sum_{j>i} U_{int}(\mathbf{p}_i, \mathbf{p}_j)$. Newton's classical equations of motions give $\frac{d}{dt}\mathbf{q}_j = \mathbf{f}_j$ and $\mathbf{f}_j = -\nabla_j U(\mathbf{p})$, where ∇_j is the gradient at the j-th body's position $\mathbf{p}_j$. Applying the relationship between position and velocity, $\frac{d}{dt}\mathbf{p}_j = \mathbf{q}_j$, yields a set of coupled differential equations for the N bodies.

B. Potential Functions for Network Dynamics

Mobility between bodies is governed by the description of the potential function $U(\mathbf{p})$, which in turn yields force and causes motion. We therefore need to define potential functions that suitably capture the need for causing mobile devices to move. We will do this in two parts: first describing possible external potential functions $U_{ext}(\mathbf{p})$, and then describe internal (i.e. pairwise) potential functions $U_{int}(\mathbf{p}_i, \mathbf{p}_j)$.

External potential functions may be viewed as representing factors coming from the environment that should influence the motion of a sensor. For example, suppose that a sensor network has been deployed to perform monitoring functions. It may be that there are regions of the network that deserve more attention than other regions, and therefore we might wish for mobile devices to be attracted to these regions. As another example, it might be desirable for a set of monitoring devices to migrate, perhaps to monitor a moving asset with a known trajectory or perhaps to sweep through a coverage area. We present such potentials in Figure 1 (a) and (b).

Internal potentials typically correspond to either attractive or repulsive pairwise interactions between entities. An attractive potential is the gravitational potential $U(\mathbf{p}_i, \mathbf{p}_j) = \frac{-G}{\|\mathbf{p}_i - \mathbf{p}_j\|}$ where G is the gravitational constant and we have taken the masses to be 1. An example of a repulsive potential is Coulomb's potential from electrostatics, $U(\mathbf{p}_i, \mathbf{p}_j) = \frac{Q_i Q_j}{4\pi\epsilon_0 \|\mathbf{p}_i - \mathbf{p}_j\|}$, where Q_i and Q_j are charges and ϵ_0 is the permittivity of free space. In general, we desire potential functions that are capable of dispersing mobile nodes without causing them to separate too greatly from each other. These potential functions, often known as dispersive potentials, involve repulsive and attractive components. An example of such a potential function is the Lennard-Jones potential from thermophysics [7] $U(\mathbf{p}_i, \mathbf{p}_j) = 4\epsilon \left[\left(\frac{\sigma}{\|\mathbf{p}_i - \mathbf{p}_j\|} \right)^{12} - \left(\frac{\sigma}{\|\mathbf{p}_i - \mathbf{p}_j\|} \right)^{6} \right]$, where σ describes the radial intercept, and ϵ governs the *well depth*, as depicted in Figure 1 (c).

The forces between any two nodes may be calculated via the gradient of the potential function. In order to keep the formulation of the force calculation simple, throughout this paper, we have chosen to use the following linear-force potential function

$$U(d) = \begin{cases} \frac{c}{2}d^2 - cr_0 d + C & d \leq R \\ K & \text{otherwise} \end{cases}, \qquad (1)$$

where $d = \|\mathbf{p}_i - \mathbf{p}_j\|$ is the distance between nodes at $\mathbf{p}_i$ and $\mathbf{p}_j$, c, r_0, R, C and K are preset parameters. In this case, the force $\mathbf{f}_{ij}$ between nodes i and j is linear,

$$\mathbf{f}_{ij} = \begin{cases} c(r_0 - d)\mathbf{u}_{ij} & d \leq R \\ 0 & \text{otherwise} \end{cases}. \qquad (2)$$

Here $\mathbf{u}_{ij}$ is the unit vector in the direction $\mathbf{p}_i - \mathbf{p}_j$. We have depicted an example of the potential and magnitude of $\mathbf{f}_{ij}$ in Figure 1 (d). We may adjust r_0 and R to control the amount of repulsive and attractive forces.

C. Ideal Simulation Framework and Convergence

In order for network dynamics to be used to govern the motion between objects, the information describing the evolution of the potential functions must either be available to a central entity serving as the motion administrator, or must be available to each mobile device. It is not only unreasonable to expect that a continuous-time representation of system potential function to be available, but it is also generally intractable to explicitly solve an arbitrary system of equations that would describe the system's motion. Therefore, we envision that the use of network dynamics will involve discrete time steps. As a result, at every time step, we will assume that a description of the system's potential function is available. We will examine the natural discretization of Newton's equations of motion, and argue that such a formulation leads to mobile devices performing extra mobility. To address this concern, we propose to formulate the equations governing network dynamics using steepest descent optimization methods.

Suppose we have a system of N objects, and consider the evolution of the N objects' position in time n. Here, we will represent time discretely by breaking time into intervals of length T. A typical discretization involves updating the i-th object's position via

$$\mathbf{p}_i(n+1) = \mathbf{p}_i(n) + \mu_p \mathbf{q}_i(n) \qquad (3)$$

$$\mathbf{q}_i(n+1) = \mathbf{q}_i(n+1) + \mu_q \mathbf{f}_i(n), \qquad (4)$$

where μ_p and μ_q are step sizes, and the force $\mathbf{f}_i(n) = -\nabla_i U(\mathbf{p}(n))$ may be determined at each time step. The time-evolution of the network's motion is then determined by letting the above system evolve.

The classical equations of motion are second-order in time, meaning that the coupled equations tie in position, velocity and acceleration. This is suitable for mechanics, but results in undesirable properties from the point-of-view of network dynamics. Specifically, the coupling between position, velocity and acceleration implies that it is quite likely that, even when the system has reached a configuration with minimal potential energy, the N bodies will still have velocity, and hence kinetic energy. As a result, the system will escape its desirable configuration and have to compensate later by applying forces in the opposite direction. To visualize this, simply consider the classical pendulum, in which the pendulum achieves its minimal potential at the base of the trajectory, the momentum causes the pendulum to swing past and increase potential energy. The increased potential causes the pendulum to reverse direction and oscillate around the point of minimal potential.

This behavior is undesirable as the mobile device wastes movement, and hence power, compensating for momentum. We therefore, would like to modify the equations of motion to remove the effect of momentum. This may be accomplished by making the equations first-order, and have the force affect the distance traveled. For example, we may simply create an iterative system $\mathbf{p}(n+1) = \mathbf{p}(n) - v\nabla U(\mathbf{p}(n))$. We see that we are simply making a step of size v in the direction of steepest descent to minimize the potential function U. Now, once the nodes have moved into a configuration with minimal potential U, they stop moving and don't move unless a disruption is introduced that necessitates the relocation of devices. For suitable choice of v, convergence will follow from the convergence of steepest descent methods. In the following sections, where we discuss distributed versions, we shall use our steepest descent formulation and will discuss the selection of v.

III. Distributed Implementations of Network Dynamics

In this section, we discuss the mapping of network dynamics on an actual network and its implementation in a completely distributed fashion with realistic constraints.

A. Overview of Distributed Network Dynamics

In Section II, we discussed the discretization of network dynamics. The straight-forward way to formulate network dynamics involves a centralized controller with knowledge of each device's position and the ability to communicate its directives to each mobile device. In practice, however, such a formulation is unrealistic as sensor networks, by their inherently ad hoc nature, do not have a centralized infrastructure. Consequently, we must devise a set of distributed algorithms whereby each node makes decisions based on its local environment in an attempt to achieve the minimum system potential energy. Although there are different ways of implementing distributed network dynamics, there are some common features amongst these different schemes. A node can only "feel" the forces from the nodes that are within its radio range. Every node j periodically calculates the total force $\mathbf{f}_j$ from all its neighbors. If the magnitude of the total force is above a threshold, i.e. $\|\mathbf{f}_j\| > \delta$, then node j will start moving in the direction of $\mathbf{f}_j$. While moving, the node will broadcast its location information and receive location updates from its neighbors if necessary, thereby allowing each node to periodically calculate its new force and adjust its movement during the next time slice. The system dynamics proceeds until each node's net force below the threshold δ.

B. System Model and Performance Metrics

We now describe the system setup and the performance metrics that we will use in formulating and evaluating our network dynamics algorithms.

System Model: We have made the following assumptions regarding the sensor network: The network is deployed on a 2D plane. Nodes have mobility, and there is a maximum velocity associated with a mobile node. Every node knows its own location. This can be achieved by devices such as GPS [8], or through various wireless localization algorithms [9].

Performance Metrics: We propose the following metrics to evaluate the proposed distributed algorithms:

- *Movement efficiency:* The average ratio of the Euclidean distance between a node's initial position (x, y) and its final location (x', y') to the distance a node actually travels. If a network dynamics algorithm makes a node travel a total of d to reach its final location, then that node's movement efficiency may be calculated as $\sqrt{(x-x')^2 + (y-y')^2}/d$. The closer this ratio is to 1, the less extra distance a node must travel before it reaches the real destination. We measure the total system's performance by averaging the movement efficiency over all nodes.
- *Convergence time:* When a network dynamics algorithm converges, every node within the network will have a force which is below the specified threshold. At the same time, the system potential energy U will have reached its minimum. In general, a shorter convergence time is preferred.
- *Messaging overhead:* Without a centralized controller, a distributed algorithm learns external information through exchanging messages. These messages are costly as they consume energy.

C. Sequential Movement Algorithm

We will describe two distributed algorithms. The first algorithm is a sequential algorithm, which we refer to as Sequential Distributed Network Dynamics (SDND). SDND is a straightforward implementation of network dynamics in which the nodes within a neighborhood of each other allow only a single node to move. While this node is moving, the other neighbor nodes in its radio range must remain stationary.

Suppose node i receives a force from its neighbors, and we have $\|\mathbf{f}_i\| > \delta$. Therefore, it will try to move along the direction of $\mathbf{f}_i$ so that the potential energy U can decrease. Before moving it must ensure that all the neighbors are not moving. We employ the following protocol to guarantee this. It first broadcasts a $ReqMove$ message to all its neighbors and waits for their replies. It stays in the $wait$ state until all the reply messages arrive. As soon as the $ReqMove$ message reaches a neighbor node, the neighbor node replies with a $GrantMove$ message if it is not in the wait state itself; otherwise, the neighbor node replies with a $NoMove$ message. The neighbor node enters the $blocked$ state upon sending out the $GrantMove$ message, and each blocked node maintains a $wake$ list, including those nodes that it has sent $GrantMove$ messages to and that will wake it up later. Any node that is in the blocked state cannot move until it is woken up by every one from its wake list. The node that intends to move, i.e. node i, can only start movement after it receives a $GrantMove$ message from each of its neighbors. If no reply is received from a neighbor node within a time window, we assume that the neighbor node has either moved out of the range or failed, and simply ignore that neighbor. This

```
Algorithm: The Node that Intends to Move
wait(random_delay);
send(ReqMove);
state = WAIT;
while (received(NoMove) == TRUE) do
    send(DoneMove);
    wait(random_delay);
    send(ReqMove);
end
state = MOVE;
f = calculateForce();
while (||f|| > δ) do
    move(f, v, T);
    f = calculateForce();
end
send(DoneMove);
state = NORMAL;
```

Algorithm 1: Sequential Distributed Network Dynamics Algorithm for the node that intends to move.

```
Algorithm: The Neighbor Node
while (1) do
    msg = waitForMsg();
    if (msg == ReqMove) then
        if (state == WAIT) then
            send(NoMove);
        else
            send(GrantMove);
            state = BLOCKED;
        end
    end
    if (msg == DoneMove) then
        state = NORMAL;
    end
end
```

Algorithm 2: Sequential Distributed Network Dynamics Algorithm for neighboring nodes.

algorithm is deadlock-free: if two nodes send out $ReqMove$ messages at the same time, neither of them can proceed because they will not receive $GrantMove$ messages from each other. In this scenario, it is possible that some neighbor nodes have already replied with $GrantMove$ to one of them and entered blocked state. Hence, these two competing nodes, after receiving $NoMove$ from each other, should send $DoneMove$ messages immediately to wake them up. After this collision, these two nodes can each back off for a random delay, and re-initiate the entire process.

Once all the $GrantMove$ messages reach node i, it starts to move at a constant velocity of v. Its movement is broken down into time slices with duration T. At the beginning of time slice n, it calculates the force $\mathbf{f_i}(n)$, and during that time slice, it moves in the direction of $\mathbf{f_i}(n)$. We note that it is not necessary to exchange messages to update the location information of the neighbors because the neighbors are static during i's movement. The combination of time slice duration T and velocity v can have a significant impact on the convergence performance. For a particular T, a large v can cause a node to quickly approach the destination, but may also cause the node to overshoot its destination, which may prevent convergence. Consequently, a conservative value of v should be chosen. The node keeps moving until it reaches a position $\mathbf{p}_i$ where the force it receives is below the threshold, i.e. $\|\mathbf{f}_i\| < \delta$. At this time, i stops movement and wakes up all the neighbors that are blocked by sending the $DoneMove$ message.

The algorithms for the node that intends to move, and the rest of the neighboring nodes are presented in Algorithms 1 and 2.

D. Parallel Movement Algorithm

The strategy behind the sequential movement algorithm is rather straightforward and easy to implement. However, the SDND algorithm suffers from inefficiencies due to its sequential nature. In particular, SDND may lead to high messaging overheads because a large number of messages must be exchanged to serialize the movements. Additionally, since SDND limits the amount of devices that may move at any time, SDND can experience slow convergence. In order to address these issues, we next propose a parallel version of network dynamics algorithm, which we call the Parallel Distributed Network Dynamics (PDND) algorithm.

In PDND, any node that intends to move can start movement immediately, and does not need to get approval from its neighbors. A moving node advertises its location every T time, while a stationary node updates its location information at a much coarser granularity. Every node maintains a neighbor table which records each neighbor's location. Note that this information may not be up-to-date. Based on the neighbor table, each node calculates the total force using its neighbors' positions. If a node's force is greater than the threshold, then that node will move along the direction of the force for T time. After T time, it sends out its new location, and re-calculates the force, and determines whether it needs to move in the next time slice, and the direction if it needs to. It stops moving when the total force it receives is below the threshold δ. Every node repeats this process iteratively until all the nodes reach steady state.

Unlike the case in the sequential algorithm, where the movement of a node involved a fixed velocity in the direction of the node's net force, for our parallel movement algorithm we propose the use of an adaptive velocity. The motivation for using an adaptive velocity is that it will allow the dynamics of the network to converge to steady-state quicker. In particular, we would like to control the velocity of each node based on the magnitude of the force. To accomplish this, we propose to use the following strategy to determine the velocity: If $\mathbf{f_i}(n)$ is the force at the ith node at the beginning of the nth time slice, and $\mathbf{f}_i(n+1)$ is the force at the beginning of the $(n+1)$th time slice, then the velocity during the $(n+1)$th time slice is $v_i(n+1) = \frac{\|\mathbf{f}_i(n+1)\|}{\|\mathbf{f_i}(n+1)-\mathbf{f_i}(n)\|} v^{max}$.. The rationale behind this strategy is that, if the current force is still large compared to the change in the force from the last time slice, then this indicates that the node is still far from its destination, and it can move at a faster speed. On the other hand, if the current force is small compared to the difference, the node needs to slow down because it is already close to the convergence point. When $\mathbf{f}_i(n)$ and $\mathbf{f}_i(n+1)$ have positive projections on each other, the node is still moving in roughly the same direction. Intuitively, it is safe for it to speed up since, in this case, $\|\mathbf{f}_i(n+1) - \mathbf{f}_i(n)\|$ is a relatively small number. When $\mathbf{f}_i(n)$ and $\mathbf{f}_i(n+1)$ have negative projections on each other, the node has overshot and it needs to move backwards to converge. This node needs to lower its velocity because the convergence point is rather close. The velocity is made small because $\|\mathbf{f}_i(n+1) - \mathbf{f}_i(n)\|$ is large.

The PDND algorithm is summarized in Algorithm 3. This is a completely distributed implementation, wherein the nodes do not need to synchronize with each other. For this algorithm, the location update interval T is an important parameter. A smaller T can reduce the oscillation of the movement, thus resulting in a faster convergence. At the same time, a smaller T, however, also incurs more overhead because more control messages (location update messages) will be exchanged.

Algorithm: *Parallel Movement Algorithm*

```
old_f = f;
f = calculateForce(my_location, neighbor_location);
while (‖f‖ > δ) do
    new_f = f;
    v = calculateVelocity(old_f, new_f);
    move(f, v, T);
    send(my_location);
    old_f = f;
    f = calculateForce(my_location, neighbor_location);
end
```

Algorithm 3: Parallel Distributed Network Dynamics Algorithm.

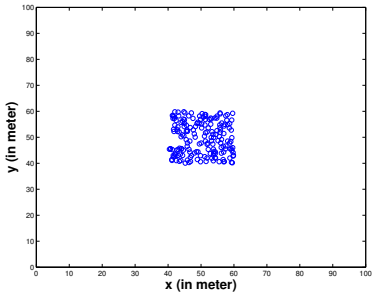


Fig. 2. All the nodes are clustered together in the initial topology.

IV. Applying Network Dynamics to Managing Mobile Sensor Coverage and Topology

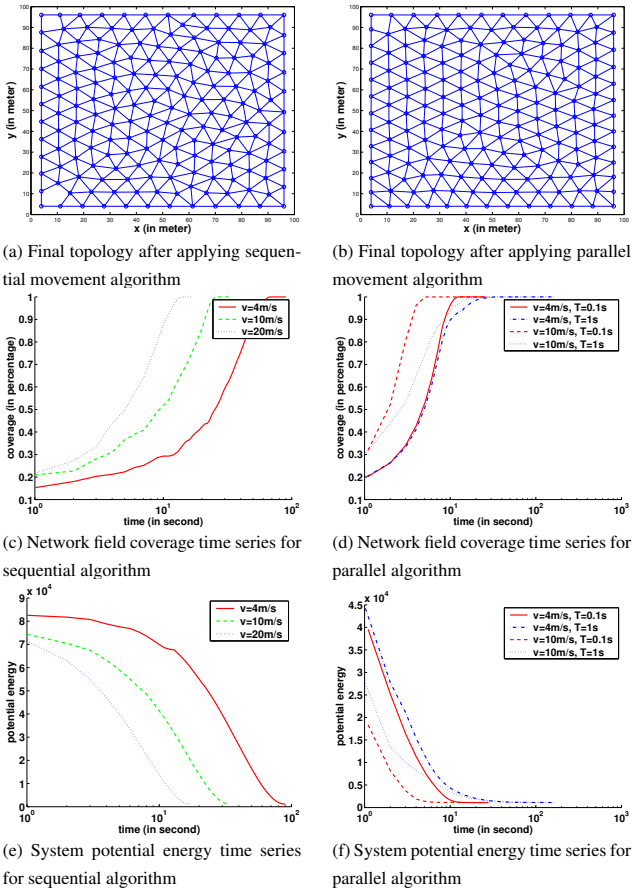
We will now explore an application of network dynamics to the problem of managing the coverage area provided by a deployment of sensor nodes. To set up the problem, suppose that a set of sensor nodes are deployed to monitor a particular region. In many applications, it is difficult to initially deploy the sensors to have a uniformly random coverage over the whole region, especially when the region is large. Rather, since it is often easier to deploy the sensors over a smaller region, or randomly over a larger region, it is desirable to then have the nodes adjust themselves to provide full coverage over the entire area. We may employ the proposed distributed network dynamics algorithms in conjunction with suitable dispersive potential function to help the nodes evenly cover the network field. We will begin our discussion of such strategies by first looking at the model for the forces that we use, and then present the results of both sequential and parallel algorithms.

Force Model: In this application, the forces come from two sources: internal interaction between nodes, and repulsive forces around the boundary of the sensor region. We describe each type of force below:

- The force between two nodes i and j. In this case study, we use the linear-force model to calculate the force between neighboring nodes i and j:

$$\mathbf{f}_{ij} = \begin{cases} c(r_0 - d_{ij})\mathbf{u}_{ij} & \text{if } d_{ij} \leq R \\ 0 & \text{otherwise} \end{cases}, \qquad (5)$$

where d_{ij} is the Euclidean distance between i and j, $c > 0$ and $r_0 > 0$ are two pre-set constants, R is the radio-range, and $\mathbf{u}_{ij}$ is the unit vector starting at j and pointing to i. r_0 denotes the balanced distance between two nodes where the resulting force is zero. If $r_0 < R$, then $\mathbf{f}_{ij}$ is an attractive force when $r_0 < d_{ij} < R$, and a repulsive force otherwise; if $r_0 \geq R$, $\mathbf{f}_{ij}$ is always a repulsive force. The total force exerted on i is $\mathbf{f}_i = \sum_{j \in N(i)} \mathbf{f}_{ij}$, where $N(i)$ is the set of neighboring nodes of i.



(a) Final topology after applying sequential movement algorithm

(b) Final topology after applying parallel movement algorithm

(c) Network field coverage time series for sequential algorithm

(d) Network field coverage time series for parallel algorithm

(e) System potential energy time series for sequential algorithm

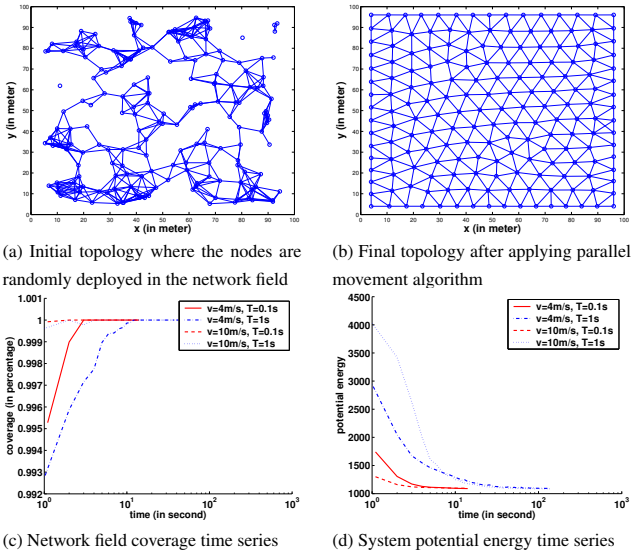
(f) System potential energy time series for parallel algorithm

Fig. 3. Distributed network dynamics algorithms can make a clustered initial topology (shown in Figure 2) become a uniform topology.) The time axis is in log-scale.

The parameter r_0 is a parameter that may be used to govern the distance between two adjacent nodes in the final network configuration. In particular, the area of the coverage region, and the number of mobile node affect the choice of r_0. It should be noted that if there are not enough nodes to sufficiently cover the region, and $r_0 > R$, then the network will ultimately end up partitioned into many small subnetworks.

- *The force from the network boundary:* In our experiments, we model the effect of the boundary as a region of repulsive force of width d_0 around the region's boundary. If the distance between a node and the boundary is less than a pre-set value d_0, and the node's net force is towards the boundary, then the force exerted on that node is a repulsive force that cancels the component of the net force perpendicular to the boundary. The net result is that the boundary potential cancels potentials that would force the node to invade this boundary region. We assume that each node is already pre-programmed with the boundary's location prior to deployment.

Simulation Results: In our experiments, we first created an initial random deployment involving 180 nodes clustered about the center of the network field, as shown in Figure 2. The network field is $100 \times 100\ m^2$, and all the nodes are initially deployed in a small area of size $20 \times 20\ m^2$. Each node's radio range is 10m, and r_0 in Equation 5 is set to 10m. Usually, if there are N nodes over a network field of area A, then the value of r_0 should be governed by the equation: $\sqrt{A/N\pi} < r_0 \leq R$



(a) Initial topology where the nodes are randomly deployed in the network field

(b) Final topology after applying parallel movement algorithm

(c) Network field coverage time series

(d) System potential energy time series

Fig. 4. Applying parallel PDND algorithm to an initial network topology where the nodes are randomly distributed. The time axis is in log-scale.

where R is the radio range. Following the initial deployment, we applied both the sequential algorithm and the parallel algorithm with the linear-force model, with $\delta = 0.1$.

Figures 3(a) and (b) show that both algorithms can successfully make the 180 nodes uniformly distributed across the network field. The edges in these two figures denote the radio connection between any two neighboring nodes. Careful inspection of Figures 3(a) and (b) reveals that there are differences between the final configurations produced by the sequential and parallel algorithm. Compared to the initial topology, the advantage of the uniform topology is obvious. In order to witness the convergence process, we also recorded the network field coverage time series, as well as how the system potential energy evolves with time. Figures 3(c) and (d) present the time series of the network coverage ratio for the sequential and parallel algorithms respectively. The network coverage ratio is the percentage of the network field that is covered by the wireless nodes. For the sequential algorithm, we present results for different velocities, while for the parallel algorithm we examine both different velocities as well as time slice durations T. We applied Monte Carlo integration methods to calculate the coverage ratio. Suppose the network field is a rectangle, whose upper-left coordinate is (x_0, y_0), and bottom-right coordinate (x_1, y_1). We may randomly generate N locations (x, y) within the field (i.e. x is a uniformly random number between $[x_0, x_1]$, and y is a uniformly random number between $[y_0, y_1]$). Among these N locations, suppose that n locations are covered by the mobile nodes. Then the corresponding coverage ratio is n/N. We chose a value of N sufficiently large to guarantee our estimate was stable. Irrespective of the algorithm and the parameters, the overall trend is that the coverage ratio starts from a poor value, and becomes 100% by the end of the algorithm. More importantly, we observe that the coverage ratio keeps increasing until it reaches 100%. The time series of the system potential energy are presented in Figures 3(e) and (f). Similar trends are also observed here: as time progresses, the potential energy keeps increasing, until reaching the minimum.

Sometimes it is more convenient to randomly place sensor nodes over the network field, and the corresponding initial topology will look like the one shown in Figure 4(a). We have applied both sequential and parallel algorithms to this topology, and only show the results for the parallel case for space considerations. Figures 4(b)-(d) show the final topology after the PDND algorithm converges, the network field coverage ratio time series in the course of convergence, and how the system potential energy evolves.

V. Applying Network Dynamics to Spatial Retreats

In this section, we discuss a second application of network dynamics that involves repairing mobile sensor networks subjected to attacks of radio interference.

A. Jamming attacks and Spatial Retreats

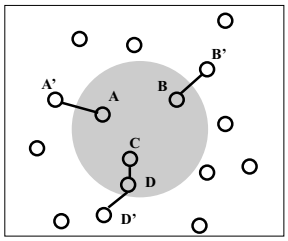
The shared nature of the wireless medium makes wireless networks susceptible to a broad array of security threats. In particular, one class of powerful attacks that has received attention recently is jamming [5, 10–12]. Adversaries may easily jam legitimate wireless communications by either continuously emitting RF signals, or by interrupting the transmission and reception of legitimate packets [12]. Either way, the net result is that legitimate traffic will be interfered with. Jamming attacks can have a particularly deleterious effect as the presence of a jammer may block whole regions of the network.

To defend against jamming attacks, two strategies were recently proposed in [5], namely channel surfing and spatial retreats. In this paper, we are interested in spatial retreats, in which jammed nodes try to evacuate from jammed regions. As such, spatial retreats are suitable for mobile networks. In the remainder of this section, we discuss our proposed spatial retreat strategy, and show how network dynamics may be incorporated into spatial retreats to achieve robustness in network communications.

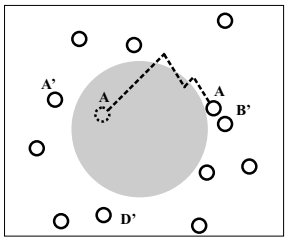
The rationale behind spatial retreats is that when mobile nodes are interfered with, they should simply move to a safe location. Assuming each mobile node can detect jamming attacks in a timely fashion [12], the key to the success of this strategy is to decide where the mobile nodes should escape to and how these nodes should coordinate their escapes. Merely escaping from a jammed region is not a sufficient defense mechanism, however, as a mobile adversary can move through the coverage area and cause large swaths of the network to relocate. By doing so, an adversary can cause the network to become unevenly distributed, or even partitioned, thereby severing network communications.

Therefore, spatial retreat strategies should be robust to mobile jammers. In order to achieve this robustness, our spatial retreat strategy has two phases: (1) *Escape phase*. In this phase, the nodes that are located within the jammed area move to "safe" regions, and stay connected with the rest of the network. (2) *Reconstruction phase*. After the jammed nodes escape from the jammed area, a distributed network dynamics algorithm is applied to re-adjust the network deployment to be more uniformly covered. In particular, after the jammer has moved on, the jammed area will leave a hole in the network coverage, and network dynamics will serve to quickly fill in the hole and restore network coverage.

We begin by discussing the escape strategy that we have developed. Suppose the network is connected before the jamming attack, i.e. every node within the jammed area is connected with nodes outside via one-hop or through multiple hops. In the example shown in Figure 5(a), before the jamming attack, node A was directly connected with A', node B was directly connected with B', node D was directly connected with D', and C was

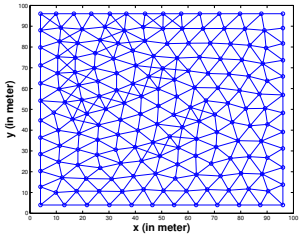


(a) The network topology when the jamming attack occurs. The jammed area is highlighted by the gray area.

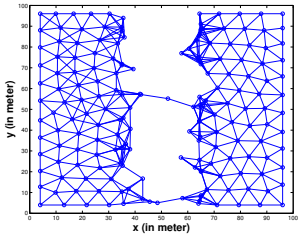


(b) The dashed line marks the trace through which node A escapes from the jammed area and re-connects to the rest of the network.

Fig. 5. Escaping from the jammed area.



(a) The topology before the jamming attack



(b) The topology after the jammer moves from the bottom to the top.

Fig. 6. A mobile jammer may partition the network.

connected with D' via D. After the jamming attack is detected, the nodes within the jammed area choose a random direction to evacuate. While moving, each node continuously runs the jamming detection algorithm until it leaves the jammed area. As soon as it leaves the jammed area, it tests whether there are some nodes within its radio range. If not, it moves along the boundary of the jammed area until it re-connects to the rest of the network. In Figure 5(a), if node A moves along the boundary, it will eventually arrive at a location which is between the location of A' and the original location of A, where it can re-connect to A'.

In order to make sure a node moves along the boundary of the jammed area, the node must continually run the jamming detection algorithm. Following the hull-tracing strategy in [5], it can use the simple strategy: whenever it feels the jammer's power is increasing, it makes a 90 degree left turn; whenever it feels the jammer's power is decreasing, it makes a 90 degree right turn. After it has moved a total r distance, where r is its radio range, the node will probe to see whether there is another node in its radio range (probing can also occur at a finer granularity). If not, it will continue moving along the boundary. Figure 5(b) illustrates how node A chooses a random direction to escape from the jammed area, and how it re-connects to the rest of the network using the simple policy.

We next turn to the reconstruction phase. Although our simple escape strategy guarantees that every jammed node can escape from the jammed area and successfully connect to the rest of the network, a serious problem remains. As we noted earlier, if the jammer is mobile, its movement may cause the network to become severely unbalanced, or even partitioned. As an example, in Figure 6, we depict an initial network configuration (Figure 6(a)), and then introduce a jammer that moves in the y-direction through the middle of the network. The result is a network that is severely partitioned (Figure 6(b)).

In order to address this problem, we propose to apply the network dynamics to continuously repair the network topology, regardless of the jammer movement. In this scenario, there are three types of forces in the network field: the forces between the nodes, the force from the boundary of the region, and the force from the boundary of the jammed area. We used the same model for the internode forces and region boundary forces as we used in Section IV. The force that we used for the boundary of the jammed area is similar to the force for the boundary of the network field. Nonetheless, we cannot pre-program the jammed area boundary as in the case of network field boundary. Instead, jammed area mapping techniques such as the one proposed in [10] should be incorporated here.

We now examine robust spatial retreats by looking at an experiment involving a mobile jammer cutting across the network coverage area. Figure 7 illustrates the evolution of the network's topology as the jammer moves through the network, and the robust spatial retreat algorithm not only evacuates the jammed area but also repairs regions left empty by the mobile jammer. In this experiment, we used PDND. Overall, the benefit of applying distributed network dynamics is two-fold: (1) as soon as the victim nodes escape from the jammed area, instead of gathering around the boundary, the nodes redistribute to cover the rest of the network field more evenly; and (2) as soon as the jammer leaves the current location, the resulting "hole" can be quickly repaired.

sectionRelated Literature

The mobility of the communication infrastructure is a distinguishing feature of mobile ad hoc and sensor networks. It has become a recent trend in the area of mobile communications to look for scenarios where mobility can improve network performance. Kansal et al. propose that mobility can address topology adaptivity, increase network and energy capacity, and also help a network redistribute resources [13]. Goldenberg et al. [1] examined cases where mobility improves network performance. They presented one of the first algorithms for controlling node mobility to improve network performance. This distributed algorithm adjusts the locations of mobile devices according to local information to achieve global communication objectives.

Another application of mobility addresses the coverage area provided by a sensor network deployment [2, 3, 14–17]. These papers use controlled mobility to enhance sensor coverage following a non-ideal initial deployment of sensors. Along a similar vein, a technique that used mobility to repair networks in the presence or faults due to energy consumption (depleted batteries) was presented in [18].

The work of Howard et al. [16] is one of the first papers to look at mobile devices as particles that exert forces upon each other, as well as introduce the notion that repulsive forces may be used to demarcate boundaries and objects. They used the traditional equations of motion as their guidelines for governing motion. The authors did not address important issues, such as the waste of energy as a result of momentum causing minimal potential configurations to be overshot. Their techniques were adopted in [2, 14, 19]. Wang et al. presented the linear-force model that we used in our work [2]. Poduri et al. follow the construction of Howard et al. and incorporate modifications that attempt to maintain a level of connectivity between nodes [19]. Nonlinear force models were employed in [14]. In [14], the authors present an off-line algorithm that may be run at cluster heads to determine where to move sensors in order to maximize sensor field coverage. The drawback of such a scheme is the cluster head must receive messages from mobile sensors, compute positions, and then issue commands to sensor nodes before the coverage is improved. By contrast, our distributed algorithms are online and adjust to needs to change the configuration, and exhibit improved coverage as the system

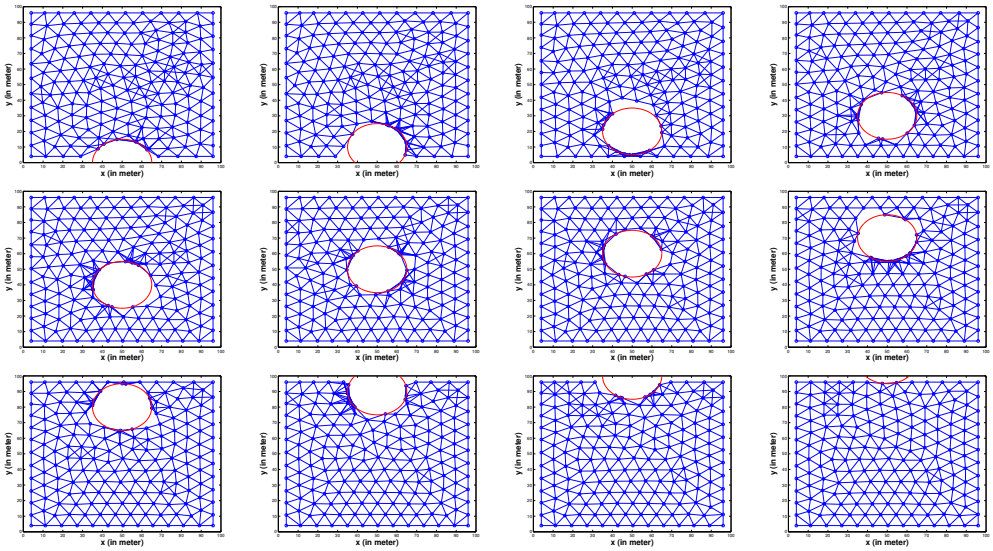


Fig. 7. Robust spatial retreats repairs the effect of a jammer passing through the coverage region using the PDND algorithm.

reconfigures step-by-step towards convergence.

The issue of jamming detection was briefly studied by Wood and Stankovic in [10] in the context of sensor networks. Several different models for jammers, along with multimodal detection mechanisms were presented in [12]. Countermeasures for coping with jammed regions in wireless networks has been studied in [5]. In [5], two countermeasures are presented for coping with jamming. The first method, channel surfing, involves a form of on-demand link-layer frequency hopping, where valid participants change the channel they are communicating on when a denial of service attack occurs. The second method, spatial retreats, focused on the case of a stationary jammer, and did not deal with the pernicious effects of a mobile jammer.

VI. Concluding Remarks

This paper presented a framework for managing the mobility in a mobile ad hoc network by defining suitable potential energy functions. The design of these potential functions captures the operational goals of the network, and can encourage nodes to gather, disperse, or migrate. We argued that the equations of motions should not directly follow those of classical mechanics, and instead proposed that the equations of motion should follow a steepest descent formulation to minimize potential energy. Since centralized control of motion is not practical, we devised several distributed algorithms, whereby devices make movement decisions based on the position information of other devices within their radio range. The performance of network dynamics may be evaluated by measuring convergence time, the efficiency of the node movement, messaging overhead, as well as by monitoring the evolution of the system potential energy with time. Using these performance metrics, we evaluated our distributed algorithms on the problem of dispersing an initially clustered deployment of sensors to achieve roughly uniform coverage. We observed that our algorithms successfully adjust the topology and, during convergence, monotonically increase network coverage. We then examined the application of network dynamics to the problem of managing a MANET's topology in the presence of a jamming attack. In particular, simple spatial retreat strategies, whereby jammed nodes seek to escape the jammed area, can lead to fractured network topologies. By employing network dynamics in conjunction with a jamming escape algorithm, we developed a robust spatial retreat strategy capable of repairing network partitioning as a jamming device cuts through the MANET.

References

[1] D. Goldenberg, J. Lin, and A. Morse, "Towards mobility as a network control primitive," in *Proc. ACM MobiHoc'04*, May 2004, pp. 163–174.

[2] G. Wang, G. Cao, and T. Porta, "Movement-assisted sensor deployment," in *Proc. IEEE INFOCOM'04*, Mar. 2004, pp. 2469–2479.

[3] G. Wang, G. Cao, T. Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *Proc. IEEE INFOCOM'05*, Mar. 2005.

[4] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[5] W. Xu, T. Wood, W. Trappe, and Y. Zhang, "Channel surfing and spatial retreats: defenses against wireless denial of service," in *Proceedings of the 2004 ACM workshop on Wireless security*, 2004, pp. 80 – 89.

[6] R. P. Feynman, *Feynman Lectures On Physics*, Addison Wesley, 1970.

[7] C.G. Gray and K. E. Gubbins, *The Theory of Molecular Fluids 1: Fundamentals*, Clarence Press, Oxford, 1984.

[8] P. Enge and P. Misra, *Global Positioning System: Signals, Measurements and Performance*, Ganga-Jamuna Pr, 2001.

[9] K. Langendoen and N. Reijers, "Distributed localization in wireless sensor networks: a quantitative comparison," *Comput. Networks*, vol. 43, no. 4, pp. 499–518, 2003.

[10] A. Wood, J. Stankovic, and S. Son, "JAM: A jammed-area mapping service for sensor networks," in *24th IEEE Real-Time Systems Symposium*, 2003, pp. 286 – 297.

[11] A. Wood and J. Stankovic, "Denial of service in sensor networks," *IEEE Computer*, vol. 35, no. 10, pp. 54–62, October 2002.

[12] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Proceedings of the 2005 ACM International Symposium on Mobile Ad Hoc Networking and Computing*.

[13] A. Kansal, M. Rahimi, D. Estrin, W. Kaiser, G. Pottie, and M. Srivastava, "Controlled mobility for sustainable wireless sensor networks," in *Proc. IEEE SECON'04*, Oct. 2004, pp. 1–6.

[14] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," in *Proc. IEEE INFOCOM'03*, Mar. 2003, pp. 1293–1303.

[15] A. Howard, M. Mataric, and G. Sukhatme, "An incremental deployment algorithm for mobile robot teams," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02)*, Sept. 2002, pp. 2849–2854.

[16] A. Howard, M. Mataric, and G. Sukhatme, "Mobile sensor network deployment using potential fields: a distributed, scalable solution to the area coverage problem," in *Proc. The 6th International Symposium on Distributed Autonomous Robotics Systems (DARS'02)*, June 2002.

[17] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. on Robotics and Automations*, pp. 243–255, 2004.

[18] S. Ganeriwal, A. Kansal, and M. B. Srivastava, "Self aware actuation for fault repair in sensor networks," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'04)*, Apr. 2004, pp. 5244–5249.

[19] S. Poduri and G.S. Sukhatme, "Constrained coverage for mobile sensor networks," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004, pp. 165–171.