Towards Mobility as a Network Control Primitive

David K. Goldenberg* [*] dkg6@cs.yale.edu
Jie Lin§ jie.lin@yale.edu
A. Stephen Morse*§ as.morse@yale.edu
Brad E. Rosen* brad.rosen@yale.edu
Y. Richard Yang*[†] yry@cs.yale.edu

Yale University Departments of Computer Science* and Electrical Enginering§
New Haven, CT 06511 USA

ABSTRACT

In the near future, the advent of large-scale networks of mobile agents autonomously performing long-term sensing and communication tasks will be upon us. However, using controlled node mobility to improve communication performance is a capability that the mobile networking community has not yet investigated. In this paper, we study mobility as a network control primitive. More specifically, we present the first mobility control scheme for improving communication performance in such networks. Our scheme is completely distributed, requiring each node to possess only local information. Our scheme is self-adaptive, being able to transparently encompass several modes of operation, each respectively improving power efficiency for one unicast flow, multiple unicast flows, and many-to-one concast flows. We provide extensive evaluations on the feasibility of mobility control, showing that controlled mobility can improve network performance in many scenarios. This work constitutes a novel application of distributed control to networking in which underlying network communication serves as input to local control rules that guide the system toward a global objective.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*wireless communications*

General Terms

Algorithms, Performance, Design

Keywords

Mobility Control, Routing, Self-Configuration in Ad Hoc Networks

1. INTRODUCTION

As technology rapidly progresses, diverse sensing and mobility capabilities will become more readily available to devices. For example, many modern mobile robots are already equipped with various sensing capabilities. As another example, there are presently research activities on low-power robotic insects that move in a variety of ways, including flying (e.g., [2, 1]) and skimming across the surface of water (e.g., [3, 15]). Once mobility becomes feasible, we envision that large systems of such mobile autonomous agents performing various important tasks will be soon to come. Communication will undoubtedly be one of the essential functionalities of these mobile networks. The objective of this paper is to explore the novel capability of these networks to optimize their communications using controlled mobility.

One can envision many settings in which mobility can potentially be used to improve network communications. One such scenario is a long term "bugging" deployment of self-organizing mobile sensors whose purpose is to intercept or record, and then report as much data as possible from a target such as an enemy communication tower or command center. If the sensor nodes are able to move into positions that minimize the energy cost of reporting this stream of data out of the network, the amount of useful information the network can transport would be maximized. Similar arguments for mobility can apply to long-term concast data gathering [9] or to aggregation of large data events in a GHT [23].

One can also imagine mobile networks being uniformly deployed over space with the intention that when a large, geographically dispersed user such as a military division moves in and sets up a base, the network will adapt its configuration in order to best serve the specific communication demands of that user. Such adaptive wireless networks with the capability to autonomously align themselves to fit user needs would be tremendously useful.

In general, long-term deployments which exhibit persistent or habitual communication patterns are prime candidates for the application of mobility to improve network performance. In such settings, the traffic will be regular enough and high enough in volume to warrant nodes expending energy moving in order to more cheaply forward traffic.

There may also exist situations where the power source for mobility is renewable but separate from a non-renewable power source for communications. Such situations could exist in hybrid bio-electronic systems, the simplest example of which is a network of people carrying small radios running on unrechargeable batteries. A more fanciful example is a system of simple living organisms such as insects that are outfitted with radio transmitters and whose motion is controlled by a neuro-electronic interface. In light of the fact that mobility is a capability already perfected by nature, while wireless communication is a human work-in-progress, this type of technological separation of duties might have its merits.

While the previous discussion has motivated some of the potential applications of controlled mobility, there are still few studies in the mobile networking literature on improving communication performance through this capability.

Although mobility has the potential to improve network performance in many settings, there may also exist scenarios in which mobility will be less effective due to various extenuating factors

[*]Supported in part by 2001 NSF Graduate Research Fellowship DGE0202738.
[†]Supported in part by NSF grants ANI-0207399 and ANI-0238038.

including hardware limitations and traffic patterns. The objectives of this paper, therefore, are to 1) analyze when controlled mobility can improve fundamental networking performance metrics such as power efficiency and robustness of communications; and 2) provide initial design for such networks.

One major issue in using mobility is how to effectively control it. Designing mobility control algorithms for communications is challenging, because any scheme that would achieve the apparent potential of the idea should address the following issues.

- First, the precise nature of any effective mobility control will be application dependent. It is clear that nodes will need to move differently under different traffic patterns, e.g., a single source-destination pair (called a single flow), multiple source-destination pairs (called multiflows), or multiple sources and single destination (called concast). Is there a single self-adaptive mobility control scheme that can be applied to such a broad range of scenarios?

- Second, for scalability and robustness purposes, there should not be a central entity who computes the movements of all the nodes. In other words, the mobility-control scheme should be a totally distributed scheme. Is there a mobility control scheme such that although each node makes movement decisions for itself informed by purely local information, the collective system achieves desirable global properties?

- Third, the distributed mobility-control scheme should be able to self-organize the nodes to optimize a performance metric while at the same time satisfying other constraints. For example, although one major objective of a mobility-control algorithm could be to optimize data reporting power efficiency after target detection, it may be important that the network maintain connectivity and/or coverage throughout the operation of the distributed algorithm. Can we design a general mobility-control scheme possessing the flexibility to optimize communication performance while simultaneously conforming to user-imposed connectivity/coverage requirements?

The framework proposed in this paper is the first attempt to design and analyze a system addressing the above issues.

The foundation of our system's self-organizing capability is a distributed descent primitive. One inspiration for this primitive is the distributed averaging algorithm used in [16, 23]. The averaging algorithm of Rao et al. [23] operates in virtual space; our system subsumes such averaging as a special case and operates in physical space. Another inspiration of our primitive is the rendezvous algorithm proposed by Lin, Morse and Anderson [20]. The objective of the rendezvous algorithm is to have all nodes in an arbitrary connected network converge to a single point in space by using uniform, distributed, and locally informed mobility control rules. In this paper, we generalize elements of the two algorithms to design a powerful and self-organizing primitive that can achieve diverse configuration goals and that can be gracefully tuned to ensure desirable network properties such as connectivity, coverage, and power efficiency.

We apply our mobility-control primitive to a broad range of traffic scenarios, under different application requirements. For each scenario, we present and formally prove the correctness of our algorithm. We perform extensive simulations to evaluate the effectiveness of controlled mobility. Our evaluations show that there are many scenarios where mobility control can achieve substantial performance gains. For example, in a random network, we simulate a realistic scenario in which 10 Kbps voice stream data flows over a single 1 Km long greedily routed multihop path of mobile nodes capable of moving at around 0.1 m/s. In under a minute, our mobility control is able to guide the network to its optimal routing configuration in which communication uses as little as 50% of the energy originally required. Taking into account the cost of mobility, total energy savings are realized after five minutes.

The remainder of this paper is organized as follows. In Section 2, we discuss related work and compare our approach with previous approaches. In Sections 3, 4, 5 respectively, we present algorithms for optimization of single source-destination pair, multiple source-destination pair, and many-to-one traffic patterns. In each section, we include extensive simulation results to analyze when controlled mobility is effective and demonstrate the effectiveness of the presented algorithm. We conclude and discuss future work in Section 6.

2. RELATED WORK

Although mobility has been extensively investigated in the mobile networking community, the focus so far has been on random mobility, e.g., [19, 17, 28], instead of controlled mobility. For example, in [10], Grossglauser and Tse have shown that the random movement of users can be used to improve network throughput. In [5] Chakraborty, Yau and Lui have studied algorithms that try to predict user movements to reduce power consumption.

Controlled mobility is an active research area in the control theory community; for example see [4]. In the last few years much progress has been made in designing distributed mobile systems and understanding both natural and artificial mobile systems. The focus of these studies, however, is not on network communications. For example, in [6], Cortes et al. have shown that mobility can be purposefully controlled to implement network coverage; in [18], Ladd et al. have shown that mobility can be used to improve the accuracy of network localization; in DARPA's self-healing minefield project [8], mobility is used to improve and maintain network coverage. However, none of these studies considers routing or power efficiency, two of the fundamental issues in networking and communications.

Some inspiration for this work came from the averaging algorithm, which is used in various settings, e.g., [16, 23]. With the intention of providing coordinates over which to perform geographic routing, in [23] Rao et al. let "virtual positions" of nodes converge to the potential energy minimizing configuration of an equivalent network with edges replaced by springs. In the same way in which the converged virtual configuration of Rao et al. reflects the underlying connectivity of the network, our resulting physical configuration reflects the connectivity of the portion of the network in *active use* i.e. the communicating subgraph. However, there are several major differences between our work. First, our system operates in physical space; thus we must guarantee that connectivity is preserved throughout the actual motions of the nodes. Our nodes also move to a minimum "potential" configuration, but this time with spring potentials assigned only to links *actively* being used for communication. Lastly, the more general potential functions we minimize are equivalent to the communication energy usage of a configuration. Because of this, rather than averaging, we generalize to a *weighted descent* method that optimizes realistic transmission cost models and weights neighbors according to their share of local communication volume. Furthermore, our algorithms make no assumptions about the global traffic pattern, wireless environment, or hardware power usage properties.

Another inspiration of this work is the rendezvous algorithm of Lin, Morse, and Anderson [20]. They describe distributed local algorithms for guiding a system of multiple nodes to a single point. In this paper, we combine ideas from the rendezvous algorithm with the generalized averaging scheme to design a powerful and flexible tool that can achieve power optimizing configurations and be gracefully tuned to ensure desirable network properties such as connectivity, coverage.

There is a large literature on power-efficient topology control and routing; for example see [25, 22]. A major difference between our scheme and the previous work is that we leverage mobility while the previous work assumed that mobility cannot be controlled by the communication layers of the network. As a precedent to our work, there was also a previous study [26] on the optimal positions of relay nodes for a single source-destination pair. Under a link cost model of $P(d) = a + bd^{\alpha}$, where d is distance, α a constant between 2 and 6, and a and b other constants, Stojmenovic and Lin [26] show that over all multihop paths, straight paths are most energy efficient and further that there is a unique hop count for any distance that minimizes the cost of communications. However, the focus of [26] is not on reaching the optimal configuration using a distributed algorithm. Also, the focus of [26] is on a single source-destination pair while we consider multiple source-destination pairs while maintaining connectivity during mobility control.

3. MOBILITY CONTROL FOR NETWORK WITH A SINGLE FLOW

We first present our mobility control algorithm for a network with a single active flow. This is a simple yet important application scenario.

We make the following assumptions. We assume that a path from the source to the destination consisting of nodes with a mobility capability is discovered using a routing protocol, e.g., a greedy routing protocol or one of the ad hoc routing protocols. We label the nodes from source to destination $0, 1, \ldots, n+1$. We call nodes $1, \ldots, n$ relay nodes. We assume that there is a link between two nodes iff their distance is less than a maximum communication radius r. We also assume that the relay nodes know their positions. This can be achieved through either GPS [12] or some localization methods. Note that the above assumptions can be further relaxed; however, to make our mobility control scheme clear, we do not pursue these relaxations. Finally, our mobility control algorithm is orthogonal to the routing discovery protocol.

3.1 Optimal Configuration of Relay Nodes

The objective of the relay nodes is to move to a new configuration to optimize network performance. We assume that the source and destination do not move as they are not relay nodes. This is reasonable in many application scenarios in which the source is reporting the results of some sensing task or going about some other duties, while the relay nodes are in fact relay nodes because their energy is well spent helping the source to communicate with its destination. We expect that the source or the destination could also be moving, thus requiring mobile tracking. For this case, we expect that our mobility scheme is still guaranteed to maintain a multihop communication link between them as long as the moving speed is below a threshold.

Without connectivity/coverage constraints, the optimal configuration of the relay nodes depends on the cost model of communications. One way to derive communication cost as a function of link distance is to use a link loss model, e.g., [7, 29]. If a node transmits to another node at distance d away, taking into account the loss rate of the link and minimizing the expected energy cost to send one message, we have that the transmission power function is $P(d) = \min_{\omega}\{E[\omega/S(\omega, d)]\}$, where $S(\omega, d)$ is the success rate associated with transmitting a message at power ω over a distance d. We assume that a message is successfully received precisely in the case that the signal-to-noise ratio at the receiver is higher than a certain threshold. Under various realistic probability distributions on noise, we can prove that the power function $P(d)$ is a non-decreasing convex function of d. As a result, the following theorem becomes applicable:

THEOREM 1. *Assume that the energy cost function $P(d)$ is a non-decreasing convex function. Then the optimal positions of the relay nodes must lie entirely on the line between the source and destination. Furthermore the relay nodes must be evenly spaced along the line.*

PROOF. Let d_i be the distance from node i to node $i+1$, where $i = 0, \ldots, n$. Let D denote the direct line distance from the source to the destination. Since $P(d)$ is a non-decreasing convex function, we have $\sum_{i=0}^{n} P(d_i) \geq (n+1)P(\frac{\sum_{i=0}^{n} d_i}{n+1}) \geq (n+1)P(\frac{D}{n+1})$, where the first inequality is due to the convexity of $P(d)$, and the second one holds because $P(d)$ is non-decreasing. □

3.2 Mobility Control to Reach Optimality: the Synchronous Scheme

The previous subsection has established that the optimal configuration of the relay nodes is lying evenly on the line from the source to the destination. We now introduce a uniform distributed algorithm that allows the relay nodes to move to their optimal positions.

```
▷ $x_i$: current position of node $i$.
▷ $x_{i-1}$ and $x_{i+1}$: positions of nodes $i-1$ and $i+1$.
▷ $g \in (0, 1]$: damping factor.

repeat
    send $x_i$ to neighbors $i-1$ and $i+1$
    receive $x_{i-1}$ and $x_{i+1}$
    set $x'_i = (x_{i-1} + x_{i+1})/2$
    move to $x_i + g \cdot (x'_i - x_i)$
until (convergence)
```

Figure 1: The distributed, synchronous mobility-control algorithm at relay node i. Node $i-1$ and $i+1$ are its neighbors on the routing path.

Figure 1 shows a distributed mobility control algorithm. The algorithm proceeds in globally synchronous rounds of maneuvering alternating with quiescence. The key ingredient of the algorithm is the simple averaging step, which we will extend for more complex scenarios and call the target point primitive. Note that although a node computes the average of its two neighbors, the node only moves toward this point, instead of reaching it in one step. In other words, the movement is damped. In some configurations, without this damping, oscillations can occur that inflate the total distance traveled by the nodes before convergence. Damping is also useful as a tool for avoiding node overreaction to ephemeral traffic by setting the time scale over which convergence takes place to be sufficiently large.

Next, we prove that our mobility control algorithm has the essential property that connectivity between communicating neighbors is never broken. This property ensures that throughout maneuvering, the communication functionality of the path is never compromised and that each neighbor always has contact with its two neighbors necessary for computation of a target point. Furthermore, this property can avoid the cost of re-routing, which can be a major source of overhead for many routing protocols in mobile networks.

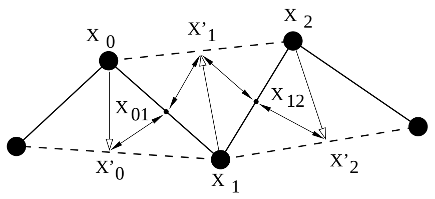


Figure 2: Illustration of Theorem 2.

THEOREM 2. *Connectivity between communicating neighbors is not lost in the synchronous algorithm.*

PROOF. Without loss of generality, suppose that node 1 at position x_1 has as communicating neighbors nodes 0 and 2 at positions x_0 and x_2. Figure 2 shows the nodes before all nodes on the path move. Let x_{ij} denote the midpoint $\frac{1}{2}(x_i + x_j)$ of the line between node i and node j. Node 1 then moves to position $x'_1 = x_{02}$. We see that

$$|x'_1 - x_{01}| = |\frac{1}{2}(x_1 - x_2)| \le \frac{1}{2}r,$$

where r is the communication radius. The new position of node 1 is within half a communication radius away from x_{01} and similarly, x_{12}. Analogous statements must hold for its neighbors as well: $|x'_0 - x_{01}| \le r/2$, and $|x'_2 - x_{12}| \le r/2$, so all new positions are within a distance of r from each other. Since nodes move along straight paths to their target points, we have shown that connectivity is guaranteed throughout the maneuvering period. □

We next establish the convergence of our algorithm; that is, we prove our algorithm always terminates. Here we define termination as arrival at a configuration in which all relay nodes are evenly spaced on the line between the source and destination.

THEOREM 3. *Using the distributed synchronous algorithm, all nodes will eventually be evenly distributed on the line between the source and destination.*

PROOF. Let $x_i(k)$ denote the position of node i after k steps of the algorithm have completed. We will assume unless otherwise stated that $i \in \{1, 2, \ldots, n\}$. The update equation for x_i is given by

$$x_i(k+1) = x_i(k) + g[\frac{x_{i-1}(k) + x_{i+1}(k)}{2} - x_i(k)]$$

for $k \in \{0, 1, \ldots\}$ and damping factor $g \in (0, 1]$.

We let $\bar{x}_i = x_0 + \frac{i}{n+1}(x_{n+1} - x_0)$ be the i-th evenly spaced point on the line between x_0 and x_{n+1}. We will show that this is the location to which node i converges.

Define the error at step k as $e_i(k) = x_i(k) - \bar{x}_i$. Observing that $\bar{x}_i = \frac{1}{2}(\bar{x}_{i-1} + \bar{x}_{i+1}) = (1-g)\bar{x}_i + \frac{g}{2}(\bar{x}_{i-1} + \bar{x}_{i+1})$, we have that for $i \in \{2, 3, \ldots, n-1\}$,

$$\begin{aligned} e_i(k+1) &= x_i(k+1) - \bar{x}_i \\ &= (1-g)x_i(k) + \frac{g}{2}(x_{i-1}(k) + x_{i+1}(k)) - \bar{x}_i \\ &= (1-g)e_i(k) + \frac{g}{2}(e_{i-1}(k) + e_{i+1}(k)). \end{aligned}$$

As for e_1 and e_n, a simple calculation reveals that

$$e_1(k+1) = (1-g)e_1(k) + \frac{g}{2}e_2(k)$$

and

$$e_n(k+1) = (1-g)e_n(k) + \frac{g}{2}e_{n-1}(k).$$

Define the error vector $e = (e_1, e_2, \cdots, e_n)^T$. It follows that $e(k+1) = Te(k) = T^{k+1}e(0)$, where

$$T = \begin{pmatrix} 1-g & g/2 & 0 & 0 & \cdots & 0 \\ g/2 & 1-g & g/2 & 0 & \cdots & 0 \\ . & . & . & . & . & . \\ 0 & \cdots & 0 & g/2 & 1-g & g/2 \\ 0 & \cdots & 0 & 0 & g/2 & 1-g \end{pmatrix}_{n\times n}$$

and can be rewritten as $I + gM$, where

$$M = \begin{pmatrix} -1 & 1/2 & 0 & 0 & \cdots & 0 \\ 1/2 & -1 & 1/2 & 0 & \cdots & 0 \\ . & . & . & . & . & . \\ 0 & \cdots & 0 & 1/2 & -1 & 1/2 \\ 0 & \cdots & 0 & 0 & 1/2 & -1 \end{pmatrix}_{n\times n} .$$

Because M is symmetric, its eigenvalues are real. From the Gerschgorin Circle Theorem, $-2 \le \rho(M) \le 0$, where $\rho(M)$ is the largest eigenvalue of M. Simple calculation reveals that neither 0 nor -2 is an eigenvalue of M; thus $-2 < \rho(M) < 0$. It follows from this and the fact that $g \le 1$ that $-1 \le 1 - 2g < \rho(T) < 1 - 0g \le 1$, i.e $|\rho(T)| < 1$. It follows from a standard result in the theory of matrix products [13] that $\lim_{k\to\infty} T^k = 0$. This implies that $\lim_{k\to\infty} e(k) = 0$, thereby establishing that the algorithm converges to a configuration of nodes evenly spaced on the line between x_0 and x_{n+1}. □

3.3 Mobility Control to Reach Optimality: the Asynchronous Scheme

While the simplicity and functionality of the synchronous algorithm is appealing, the globally synchronous mode of operation is at odds with the need for distributed algorithms that do not require any global information. To remedy this violation of the localized design requirement, we present an asynchronous algorithm, shown in Figure 3, which uses no global information and requires only that each node eventually reach its target point in bounded time.

```
▷ $x_i$: current position of node $i$.
▷ $x_{i-1}$ and $x_{i+1}$: positions of nodes $i-1$ and $i+1$.
▷ $g \in (0,1]$: damping factor.

repeat
    send $x_i$ to neighbors $i-1$ and $i+1$
    repeat listen() until ($L \wedge R == True$)
    send $moving_i$ to neighbors $i-1$ and $i+1$
    set $L := False$, $R := False$
    set $x'_i := (x_{i-1} + x_{i+1})/2$
    move toward $x_i + g \cdot (x'_i - x_i)$
    repeat listen() until (arrive in bounded delay)
until (convergence)

subroutine listen():
        upon receive $x_{i-1}$ do $L := True$
        upon receive $x_{i+1}$ do $R := True$
        upon receive $moving_{i-1}$ do $L := False$
        upon receive $moving_{i+1}$ do $R := False$

▷ $L$ and $R$: internal boolean state variables.
▷ $moving_i$: message signaling node $i$ starting to move.
```

Figure 3: The distributed, asynchronous mobility-control algorithm at node i, where $i = 2, \ldots, n-1$.

The algorithm outlined in Figure 3 defines the operation of all relay nodes other than the two nodes 1 and n respectively connected to the source and destination. Node 1 has its state variable L permanently set to $True$ and node n has R permanently set to $True$. Other than this, the operation of nodes 1 and n is identical to that of all the other relays.

A state transition diagram describing the asynchronous algorithm for nodes 2 through $n-1$ is shown in Figure 4. We omit the slightly different but straightforward diagram for nodes 1 and n. The system starts in the state in which nodes are stationary and informed of the positions of their neighbors. The state variables M, L, and R respectively represent the state of moving and of having fresh position information for the left and right neighbors. In the diagram, we abuse notation a bit and represent by x_j the reception of a message containing the position of node j, signaling that node j has stopped moving; by m_j the reception of a message indicating that node j is moving; and by $\backslash y$ the action of sending message y.

Since this is an asynchronous protocol, one potential concern is that it could cause deadlock. The proposition below shows that if messages can be reliably transmitted, then our asynchronous protocol is deadlock free. If messages can be dropped, then we can use a reliable transport protocol to guarantee reliability; or if each node in quiescent mode periodically transmits its position and a transmission arrives after finite number of retransmissions, there will still be no deadlock.

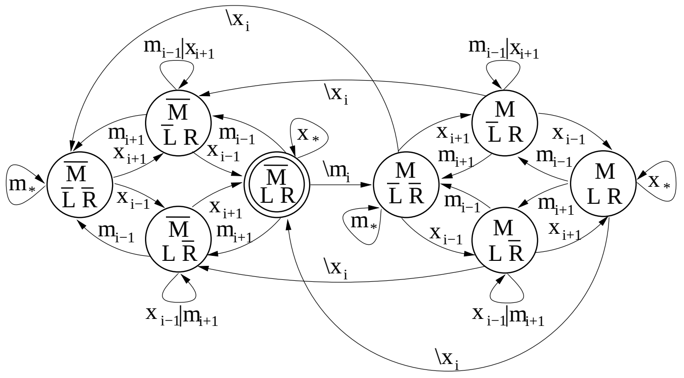


Figure 4: State transition diagram for the asynchronous algorithm: $i = 2, \ldots, n-1$.

PROPOSITION 1. *The asynchronous protocol is deadlock free, if messages are not dropped.*

PROOF. Assume the system is deadlocked. Then there is some node i that is the last to stop moving, say by time t_i. By assumption, no nodes are moving at time t_i. We will denote the state of a node j by S_j and refer to nodes by their index. Without loss of generality, upon node i stopping, $R \in S_{i-1}$, where $i-1$ is the left neighbor of i. It must be the case then that $S_{i-1} = \{\bar{L}, R\}$ or else $i-1$ would be able to move, violating our assumption of deadlock. Node $i-1$'s left neighbor $i-2$ must have stopped moving strictly before $i-1$ did, or else $L \in S_{i-1}$. Since $i-1$ stops moving after $i-2$, $R \in S_{i-2}$. But by the same argument as before, $S_{i-2} = \{\bar{L}, R\}$. As this process continues as shown in Figure 5, we reach the node $i-k$ whose left neighbor is the source and conclude that $S_{i-k} = \{\bar{L}, R\}$. But by our algorithm design, $L \in S_{i-k}$ and we have a contradiction. Hence, the system cannot be deadlocked. □

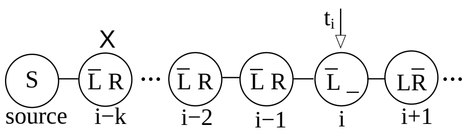


Figure 5: Illustration of deadlock proof contradiction. Node i must have at least one of L and R as $False$.

We next prove that no node loses communicating neighbors.

THEOREM 4. *Connectivity between communicating neighbors is not lost in the asynchronous algorithm.*

PROOF. At the instant that node i starts moving towards the average of its neighbors, its neighbors must be stationary, or else it would have invalidated at least one of its state bits preventing itself from leaving. As node i is moving, neither of its neighbors can move, since they invalidated a state bit upon i's departure. At the moment node i stops, it is clear that it will be within distance $r/2$ from its neighbors. □

Finally we prove that the asynchronous algorithm will also converge to an evenly spaced straight configuration. This proof uses a new proof technique similar to that used in [20]. The translation of the algorithm into an manageable mathematical model depends crucially on the invalidation of a state bit upon receiving a *moving* signal from a neighbor. Once the model is set up, the convergence result is a direct consequence of the assumptions that the source and destination are fixed and that nodes reach their targets in bounded time.

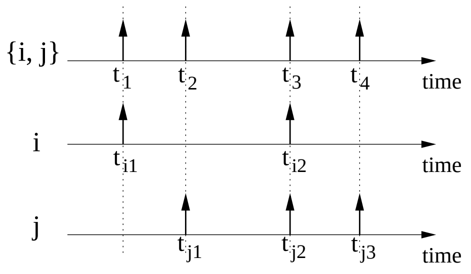


Figure 6: Illustration of event time.

Before we begin our proof, we first establish a preliminary concept. We define an *event time* to be any real time $\bar{t}_{ik}$ at which some node i begins to move, where k indicates that it is the k-th time that node i starts to move. Deleting duplicates, we now arrange the set of all event times $\{\bar{t}_{ik} : i \in \{1, 2, \ldots, n\}, k \geq 1\}$, in increasing order and label the ordered times by t_s, $s \in \{1, 2, \ldots\}$. For $i \in \{1, 2, \ldots\}$, let $s_i(k)$ denote the value of s for which $t_s = \bar{t}_{ik}$ i.e., the index of the event time at which node i moves for the k-th time. Because s corresponds to an event time at which node i starts moving, s is said to be in the *image* of s_i, $Im\{s_i\}$. This is illustrated in Figure 6 where $s_i(1) = 1, s_i(2) = 3, s_j(1) = 2, s_j(2) = 3, s_j(3) = 4, Im\{s_i\} = \{1, 3\}$, and $Im\{s_j\} = \{2, 3, 4\}$.

THEOREM 5. *Using the asynchronous algorithm, all nodes will eventually be evenly spaced on the line between source and destination, given that there is an upper bound on the time it takes for a node to move to its target point.*

PROOF. *(sketch)* For the sake of brevity, we will not include the complete proof but only a sketch. We set the damping constant $g = 1$ for clarity.

We will first define a discrete model that describes the position of each mobile node indexed by event time.

At the event time indexed by s, every node i is either starting to move (if $s \in Im\{s_i\}$), in the process of moving, or stationary. We define $x_i(s)$ as the next resting position of node i after t_s in the following way. If at time t_s, i is starting to move or in the process of moving, $x_i(s)$ represents its target point. If i is stationary at t_s, we define $x_i(s)$ to represent the position at which it is resting. This state variable $x_i(s)$ was carefully designed in conjunction with the algorithm to guarantee accurate representation of algorithm behavior as well as convergence.

If node i does not start moving at event time s, then $x_i(s) = x_i(s-1)$: The next resting location of i is the target of its last motion. Otherwise, the update equation for $x_i(s)$ is given by

$$x_1(s) = \frac{1}{2}(x_0 + x_2(s-1)),$$

$$x_n(s) = \frac{1}{2}(x_{n+1} + x_{n-1}(s-1)),$$

and

$$x_i(s) = \frac{1}{2}(x_{i-1}(s-1) + x_{i+1}(s-1)),$$

for $i \in \{2, 3, \ldots, n-1\}$.

Note that the design of our algorithm ensures that if $s \in Im\{s_i\}$, nodes $i-1$ and $i+1$ may not be moving at time t_s. Assuming $s \in Im\{s_i\}$, then without loss of generality, if node $i-1$ was moving at time t_{s-1}, then by design, node $i-1$ must have come to rest at position $x_{i-1}(s-1)$ by time t_s. If node $i-1$ was stationary at time t_{s-1}, then at time t_s, its position will be $x_{i-1}(s-1)$, as by the definition of event time, it could not have moved in-between consecutive event times. We can see now that $x_i(s)$, which is the target of node i's incipient motion, is the average of its neighbor's positions at the instant it departs. This representation is possible only because the algorithm dictates that each node invalidate the position of either of its neighbors who starts moving.

Next,we normalize the distance between source and destination to be one, and define our equilibrium positions $\bar{x}_i = \frac{i}{n+1}$. We define the error as $e_i(s) = x_i(s) - \bar{x}_i$, and set up the error equation evolving on the sequence of event times as we did in the convergence proof of the synchronous algorithm. As before, we can put the error update equations into matrix form $e(s) = M(s)e(s-1)$, where $e = [e_1, e_2, \ldots, e_n]$. In this case however, instead of the update matrix having a single form, $M(s)$ is now a matrix defined at an event time in a way which depends on exactly which nodes begin to move at that event time. This update matrix has the following form. If node $i \in [1, \ldots, n]$ does not start moving at event time s i.e., if $i \notin \text{Im}\{s_i\}$, row i appears as it does in the $n \times n$ identity matrix. If node i does in fact move, row i appears as it does in the following $n \times n$ matrix:

$$\begin{pmatrix} 0 & 1/2 & 0 & 0 & \cdots & 0 \\ 1/2 & 0 & 1/2 & 0 & \cdots & 0 \\ . & . & . & . & . & . \\ 0 & \cdots & 0 & 1/2 & 0 & 1/2 \\ 0 & \cdots & 0 & 0 & 1/2 & 0 \end{pmatrix}_{n \times n} .$$

At event time s, the error is given by the product of s matrices of type $M(s)$ applied to the initial error vector $e(0)$. In our full proof, we show that after a finite time, the maximal row sum or reduced infinity norm of this product has an upper bound strictly less than one *precisely* because the source and destination are fixed *and* every node reaches its destination in finite time. Consequently, $e(s) \to 0$ as $s \to \infty$ and the state variables and therefore the positions of all nodes converge to their desired locations evenly spaced on a line between source and destination. $\square$

3.4 Maintaining Connectivity for Non-Communicating Neighbors

In the previous two subsections, we have shown that both the synchronous and asynchronous algorithms guarantee that communicating neighbors are never disconnected. However, it is possible that as a communicating node moves towards its optimal location, it becomes disconnected from some of its non-communicating neighbors.

In networks where preserving all connectivity is important, we can introduce a simple constraint on the motion of nodes to guarantee permanent connectivity to all nodes connected to it, either communicating or non-communicating. We call the mobility control algorithm without this constraint *unconstrained mobility control*, and the algorithm with this constraint *constrained mobility control*. Specifically, the constraint is that a communicating node does not move beyond the maximum communication range away from any of its non-communicating neighbors. This means that the communicating node moves to the point closest to its target point that satisfies all constraints imposed by non-communicating neighbors.

3.5 Evaluations

We now evaluate the performance of the mobility control algorithm.

3.5.1 *Simulation Setup*

We have implemented a simulator to evaluate the performance of our mobility-control algorithms. The simulator generates nodes uniformly at random, and then randomly chooses a source and a destination. Next, it runs the greedy geographic routing protocol to locate a routing path. The nodes on this routing path then move to decrease the energy usage of the path using our synchronous mobility control protocol. Our statistics regarding network power consumption are obtained by running our algorithm on 50 different random network instantiations for each combination of parameters. It is worth noting that in uniformly random networks, the performance improvements through mobility are minimized because the paths chosen by greedy routing already tend to approximate their optimal straight configuration. Mobility in anisotropic networks or networks with geographic routing holes will yield greater performance improvements than we see here, so this study serves to delineate the baseline of the potential performance enhancements offered by mobility.

A key ingredient of the simulator is the communication cost model. We assume that the cost of transmission of a single bit over a distance d is $P(d) = a + bd^{\alpha}$, where α is between 2 and 6, and a and b are constants. This is a commonly used power function [24] where the values of a and b depend on the hardware and algorithms used for transmission, reception, decoding, and encoding. Typical values, which we adapt for use here, are $a = 100$ nJ and $b = 0.1$ nJ/m^2 [11] for a path loss model of $\alpha = 2$. For a general path loss model of $\alpha \in [2, 6]$, in order to achieve the same receiver signal-to-noise ratio as for $\alpha = 2$, we must transmit at a proportionately higher power $P' \propto d^{\alpha}$. Therefore, we use parameters $a = 100$ nJ, $b = 0.1$ nJ/m$^{\alpha}$, and $\alpha \in [2, 6]$.

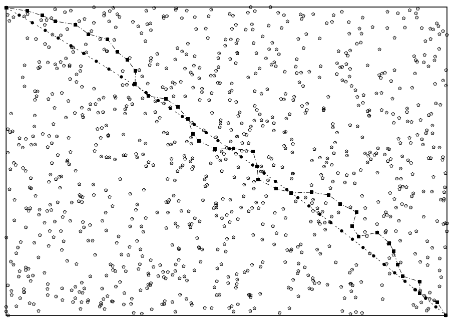
By the results of [26], communication under this cost model is achieved with least power expenditure in a multihop fashion with hop length $(1000/(\alpha - 1))^{1/\alpha}$m. Hence, in order to interpret our simulation results realistically, we scale our simulation distances so that the hop lengths typically used are of this order of magnitude.

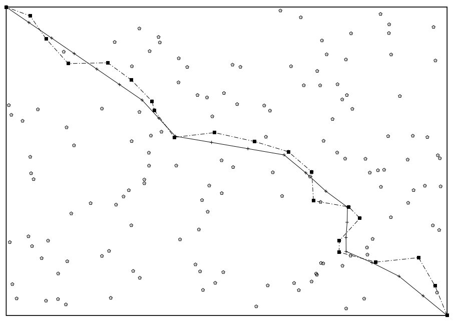
Another key ingredient of our simulation setup is the cost of mobility. We choose to use a distance proportional cost model $P_m(d) = kd$. A distance proportional cost model is reasonable for wheeled vehicles, where the energy used to accelerate can typically be recovered upon braking, neglecting losses due to friction. It is possible that flying, floating, and swimming vehicles may have to overcome larger fixed energy costs to initiate motion and less to maintain it, but we do not consider these details here and abstract to the distance proportional cost model. So as not to overestimate the potential benefit of mobility, the values of k that we consider are kept conservatively large; ranging from 0.1 J/m to 1 J/m. A one kilogram wheeled vehicle with rubber tires moving on concrete must overcome a 0.10 N force of dynamic friction, or expend 0.10 J/m [27], so an energy cost of 1 J/m does not seem unrealistic.

3.5.2 *Simulation Results*

Before we report quantitative results, we first present several figures to visually illustrate the effectiveness of the mobility control algorithm. Figure 7 shows network configurations before and after mobility control. In this experiment, we use both greedy routing and "stingy" routing to find an initial routing path. Stingy routing is a form of routing that picks the neighbor which makes the least forward progress [14]. The proved convergence of our mobility control algorithm to the straight and evenly spaced line is corroborated by these simulations. Since our algorithm converges from all initial configurations and requires only local information, it is robust against both increases in network size and highly irregular paths such as those produced by stingy routing.

The converged configurations and Theorems 3, 5 have only shown that our mobility control algorithm will move the relay nodes to the optimal configurations. However, a mobility control algorithm could move the nodes along arbitrarily long curves, thus consuming much energy for mobility. We define blowup as the ratio between the distance that a node actually travels between its initial and final positions and the straight line distance. We observe in Figure 8 that the path blowup for greedy path optimization is small, indicating that our algorithm does not suffer from large oscillations. In Figure 8, MaxMove is a parameter that expresses the maximum speed of node mobility by imposing an upper limit on distance traveled by a node per round. As one would expect, the path blowup of unconstrained optimization is greater than it is in constrained optimization; however, the value is still small. Overall, the small blowup factor of our algorithm indicates that our algorithm consumes close


(a) unconstrained; greedy


(b) constrained; greedy

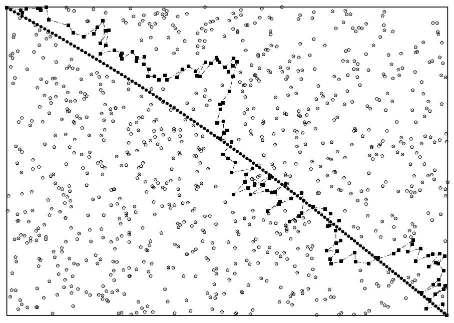

(c) unconstrained; stingy

Figure 7: Network configurations before and after mobility control. The first term of each subcaption indicates whether mobility control is constrained or unconstrained. The second term indicates the routing protocol used to find an initial routing path.

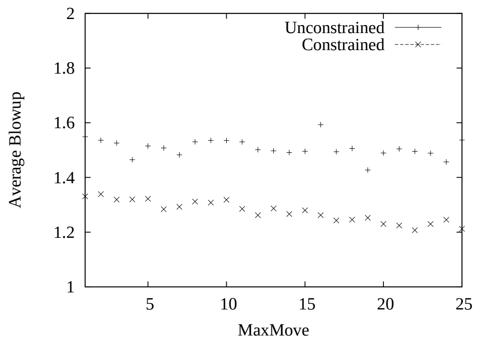
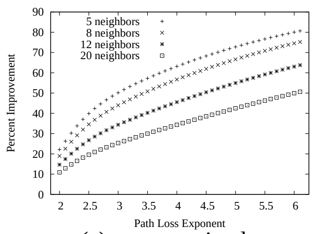


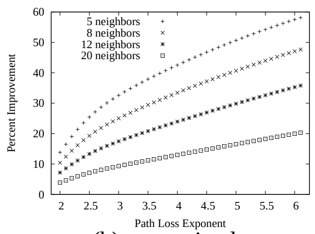
Figure 8: Blowup of network path.

to optimal energy on mobility.

Having established the convergence and small blowup factor of our algorithm, now we evaluate whether mobility control can improve the power efficiency of a routing path. We evaluate this under the cost model [11] $P(d) = 10^{-7} + 10^{-10}d^3$ and $P_m(d) = kd$, in Joules, where d is in meters, scaled from our simulation as described earlier. Note that we are comparing the power usage of greedy routing paths before and after mobility throughout. Situations in which greedy routing paths could not be found were discarded in order to produce a baseline evaluation of mobility, despite the fact that it is in precisely those discarded cases that mobility will perform the best.



(a) unconstrained



(b) constrained

Figure 9: Performance improvement of mobility control under different α values.

We first evaluate the effectiveness of mobility control for a wide range of communication environments. We control this effect by varying the value of α, the exponent of path loss and power dependency on distance. Figure 9 shows the performance improvements of unconstrained and constrained mobility control. The x-axis is the exponent while the y-axis is the percentage improvement computed as $100*(E-E_m)/E$, where E is the energy cost of the routing path before mobility control and E_m is the energy cost of the path after mobility control. For constrained mobility control, as we increase α from 2 to 6, the improvement is increased from around 10% to around 50%, translating into a potential improvement in lifetime of from 10% to 100%. The performance improvement when there is no connectivity constraint is even higher (note that the connectivity of communicating neighbors is always maintained). We observe that the typical performance improvement when there is no connectivity constraint almost doubles that with the connectivity constraint. One conclusion we can draw is that mobility control will be more effective in improving power efficiency for larger values of α.

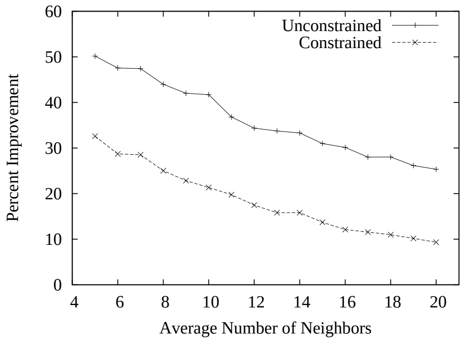
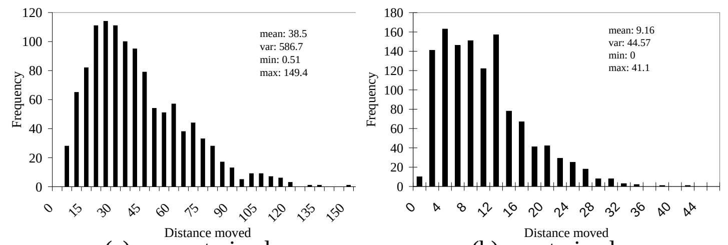


Figure 10: Effect of node density (number of neighbors) on performance improvement.

Network density, expressed through average number of neighbors, also plays a role in the effectiveness of mobility control. Figure 10 shows the result for $\alpha = 3$. We observe that with increasing density, the performance improvement decreases. This is as expected given that as density increases, greedy routing finds paths that more closely approximate the straight line. Thus one conclusion we can draw is that mobility control will increase in effectiveness as the density and regularity of networks decreases. As network density increases, it is likely that the requirement that relay nodes not lose connectivity with their static neighbors will become less essential. This may justify the use of unconstrained mobility, which produces robust energy savings even for highly dense networks.

The previous results evaluate only the total energy consumption of a path. However, a path becomes disconnected if any one of the nodes runs out of battery. From this perspective, mobility control has the further advantage that since one of its functions is to produce paths with equal hop length, the problem of premature path disconnection due to imbalanced power usage along a path is reduced. A potential problem with this claim however, is that nodes may move varying total distances, thus consuming unequal amounts of mobility energy. In other words, it could be possible that an un-



(a) unconstrained (b) constrained

Figure 11: Distribution of distance moved.

equal communication burden be reduced at the expense of unequal mobility energy cost. Figure 11 shows that the distances traveled by mobile nodes are quite balanced. We do not observe the heavy tails that would indicate some nodes spending an inordinate amount of energy on mobility relative to average.

Next, we will summarize the trade-off between mobility energy cost and communication energy cost. We assume that nodes communicate with their neighbors during mobility periods as if they are aware of the maximum distance to their neighbor over the period and send at the exact power required. In a realistic setting however, this is not possible and nodes may do one of two things. First, they may send traffic to a neighbor during mobility periods as if it is $r/2 + d_i$ units away, where d_i is the separation from the neighbor during the previous quiescent period. The alternative is that nodes send the coordinate of their target point before they begin moving. In this way, every pair of neighbors can determine the maximum potential separation between them over each mobility period and send at the appropriate minimum power level. As we will see, the transient period before the network converges is relatively short, so such details will not affect the salient properties of our simulation results.

The total energy usage was calculated using our previously described transmission power model with a path loss exponent of $\alpha = 3$ for a 10 Kbps flow[1] along a path of mobile relays approximately 1 Km long. Each maneuvering period lasts 10 seconds and during these periods, a node typically moves no more than a meter, resulting in reasonable speeds of about 0.1 m/s. We ran a trace of the evolution of the system under unconstrained mobility and another under constrained mobility. These were distinct network instantiations, resulting in the disparate total power usage.

Our results are shown in Figure 12. The slope of the lines is the energy used per MB. We can see that the slope of the line corresponding to the static network is always greater than the slope of the lines corresponding to the mobile networks, as expected. Note that also consistent with earlier simulation results, the slope of the constrained mobile traces shown in part (b) exhibits proportionately less decrease from the static case than the unconstrained mobile traces shown in (a). The higher slopes of the mobility traces close to the origin are due to the energy used on movement before convergence. We can see that the energy usage of the mobile network is substantially higher than that of the static network in the early stages of its development, meaning that mobility can incur high energy penalties if a flow is short-lived.

However, there is a number of bits sent after which the energy usage of the static network permanently outstrips that of the mobile network. For flows sending more than this number of bits, mobility saves energy. We plot this *crossing point* in part (c). Clearly, the value of the crossing point depends on the cost of mobility. As mobility becomes more expensive, the crossing point becomes larger. Another feature to notice is that the crossing point is always lower for constrained mobility than it is for unconstrained mobility. This is due to the fact that the nodes move less distance and converge faster to their final positions. So we can see distinct advantages to constrained mobility besides the intended functionality of preservation of path connectivity with static neighbors. First, a performance improvement is achieved for a smaller amount of traffic sent than in unconstrained mobility. Second, a smaller maximum energy penalty for prematurely ending flows is incurred than in unconstrained mobility. The disadvantage to constrained mobility is of course that the energy savings to be gained are more modest than they are through unconstrained mobility.

[1] 10 Kbps is a rate appropriate for minimal voice streaming.

4. MOBILITY CONTROL FOR NETWORK WITH MULTIPLE FLOWS

Controlled mobility can also be applied to a network consisting of multiple flows. For nodes that are on the path of only one flow, the averaging algorithm described in the previous section is still valid; that is, in each step a relay node moves to the average position of its two neighbors. However, in a network with multiple flows, some relay nodes will be on the routing paths of multiple flows; we call such nodes junction nodes. Applying the averaging algorithm from the previous section in the presence of junction nodes can raise several issues.

4.1 Issues of Multiple Flows

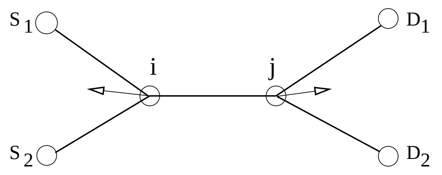


Figure 13: Illustration that moving to average can cause junction nodes i and j to disconnect, where i and j are on the paths from S_1 to D_1 and S_2 to D_2.

The first issue that arises in the application of averaging to multi-flow networks is that junction nodes may become disconnected from their communicating neighbors. An example of this is shown in Figure 13. Nodes i and j start out separated by the maximum communication radius. As soon as one of them moves toward the average of its neighbors, connectivity between them is lost.

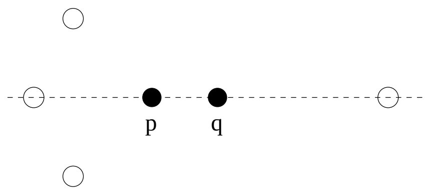
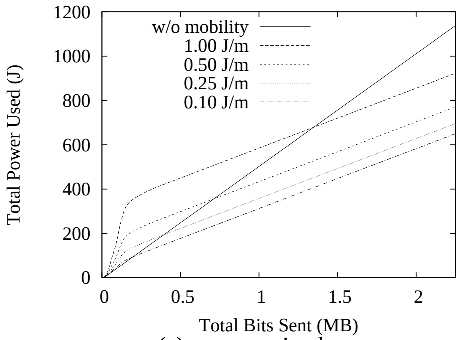
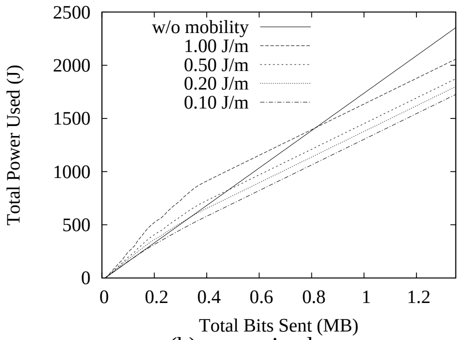


Figure 14: Effect of α on the optimal configuration; the point p is the average of the four empty circles, while the point q is the center of the minimum enclosing circle of the four empty circles.

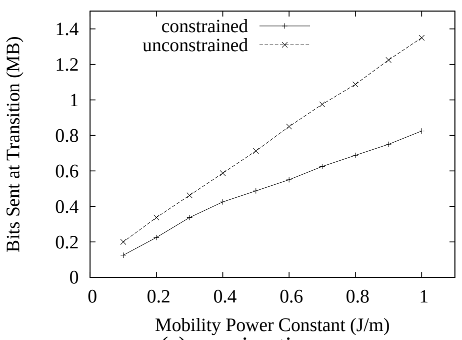
The next issue is that the optimal position of a junction node may not be the average of its neighbors. This is the case because the optimal position of a node serving as a relay between more than two neighbors depends on the power model. Consider the power model $P(d) = d^\alpha$. For $\alpha = 2$, we can show that the energy minimizing position for a relay node between multiple sources and destinations carrying equal amounts of traffic is always the average of the positions of its neighbors. However, this is only a special case. For $\alpha > 2$, the average is no longer always an optimal solution. Figure 14 illustrates this point with an example of placing a single relay between four nodes. To minimize total energy usage under a cost function quadratic in distance, the optimal point is p; but as $\alpha \to \infty$ the optimal point approaches the point q which minimizes the distance of the maximum distance node. This point q lies at the



(a) unconstrained



(b) constrained



(c) crossing time

Figure 12: Trade-off between mobility energy cost and communication energy cost.

center of the smallest circle which contains all the nodes i.e., the point minimizing the l_∞ norm. Keep in mind however, that for non-junction nodes, the average of the neighbors' positions is the energy minimizing point for all convex energy cost functions, as we showed in the previous section.

Another reason that the average is not in general the optimal target point is that a single link to a neighbor may be on any number of paths, and further, there may simply be different amounts of traffic along different flows. We need to weight the importance of neighbors by the amount of communication done with them. Note that in a line topology however, the weights of neighbors of a relay are equal due to flow conservation. Thus, in order to define a general scheme capable of adapting to these varying optimality conditions, we must devise an algorithm that yields an optimal configuration under varying power usage environments and which allows flexible weighting.

4.2 Mobility Control for Multiple Flows

To address the above two issues with multiple flows, we generalize the averaging algorithm in two ways. First, for a general power cost function and number of neighbors, instead of having nodes move toward a known minimizer, as we were doing in the line topology by moving toward the neighbors' average, nodes must now descend along a local-minimization direction in order to decrease communication cost. Our descent direction algorithm provably converges, but we omit the proof here. For a large class of power functions, the descent direction can be computed easily using only local information; even further, such algorithms can be extended to be adaptive to channel conditions such as multipath and interference.

More precisely, assume that the link cost function of a junction node to its i-th neighbor is $P_i(d_i)$. Assume that the relative amount of traffic to and from neighbor i is w_i. For example, in Figure 13, if each source sends the same amount of traffic to its destination, the weight of link ij is 2 while the weight of the other links is 1. Then the descent direction can be computed as $\Delta x = x_n - x_o$, where x_o is the current position of the junction node, and x_n can be computed as follows. Let $x^{(k)}$ denote the k-th dimension of the vector x. We have

$$x_n^{(k)} = \frac{\sum_{i=1}^n \frac{w_i}{d_i} \frac{\partial P_i}{\partial d_i} x_i^{(k)}}{\sum_{i=1}^n \frac{w_i}{d_i} \frac{\partial P_i}{\partial d_i}},$$

where x_i is the position of the i-th neighbor.

For the particular power function $P(d) = a + bd^\alpha$, we have

$$x_n^{(k)} = \frac{\sum_{i=1}^n w_i d_i^{\alpha-2} x_i^{(k)}}{\sum_{i=1}^n w_i d_i^{\alpha-2}}.$$

Note that when $\alpha = 2$ and all neighbors have the same amount of traffic, this descent direction is a constant pointing directly to the average of the neighbors' positions. In a line topology with any $\alpha > 2$, the optimal configuration of relays is at the average of their neighbors but the descent direction is no longer a constant. This results in a curved path leading to the same eventual destination as averaging. Thus the previous averaging algorithm is just a special case of this more general descent algorithm. For $\alpha > 2$, nodes must locally follow the descent direction with sufficiently small step size to guarantee convergence. For line topologies however, a small optimization is found in simply having nodes move toward the average of their neighbors regardless of the value of α, resulting in their moving along a shorter path before convergence than using the general descent direction.

To address the disconnection issue, we impose a *pairwise constraint* on every moving node. The pairwise constraint, shown in Figure 15, holds between two potentially mobile nodes that are neighbors. It makes explicit the intuition behind the disconnection impossibility proofs for the single path case by dictating that in order to stay connected to a neighbor, a node must not move outside the disk of radius $r/2$ centered at the midpoint of the line between itself and that neighbor. A separate pairwise constraint is enforced for every mobile neighbor, thereby restricting a node's motion to the intersection of a number of disks. The less restrictive *static constraint* that a node must stay within communication range r of all of its non-mobile neighbors may also be enforced. A node is potentially mobile if and only if it is actively communicating.

This pairwise constraint in conjunction with the static constraint guarantees that all pre-existing connectivity is preserved throughout any mobility.

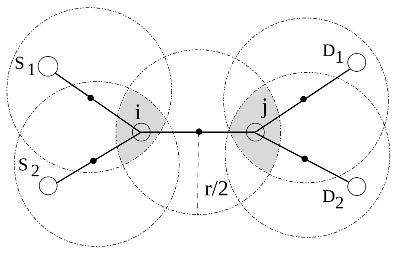


Figure 15: Illustration of pairwise constraint: nodes i and j may not move outside of their respective shaded regions.

THEOREM 6. *If the pairwise constraint is satisfied, communicating neighbors will not become disconnected.*

As an illustration of the generality of this updated scheme, note that in the case of a single flow, merely moving to the average of one's neighbors implicitly satisfies the pairwise constraint.

Since moving to the average for the general case may not satisfy the pairwise constraint for junction nodes (nodes that are on multiple flows), our algorithm has every junction node move as far as it can within its allowable region; namely the intersection of the allowed disks defined by pairwise constraints with every mobile neighbor and static constraints due to every non-communicating neighbor.

One potential concern in the use of the pairwise constraint is that it may hamper a node from reaching its optimal position. However, our observations show that the pairwise constraints are almost never active at system convergence. Thus, in practice, both the non-junction nodes and junction nodes are not held back by the pairwise constraints; as a result our multiflow algorithm leads in most cases to a globally optimal configuration of relay nodes.

Before we proceed to empirically evaluate the performance of the multiflow algorithm, we observe that our descent algorithm has a desirable self-adaptive property. That is, although this algorithm is completely uniform, namely each node runs the same algorithm, in regions of space containing multiple flows, all nodes move as far as they can along the descent direction defined by the positions of their neighbors and weights determined by traffic volume while never violating their pairwise constraints; on the other hand, in regions of space containing only one flow, nodes move to the same target points as they did in the single flow algorithm as if they are unconstrained and their neighbors unweighted. This self-adaptive property means that there need be no global coordination such as ordering a global mode change from multiflow mode to single flow mode. Such local self-adaptation means that our algorithm can respond to changes in the environment, evolution of the network, and traffic events automatically, be it the path loss exponent changing as a node moves, a node changing from a junction node to a non-junction node, or large fluctuations in traffic volume.

4.3 Evaluations

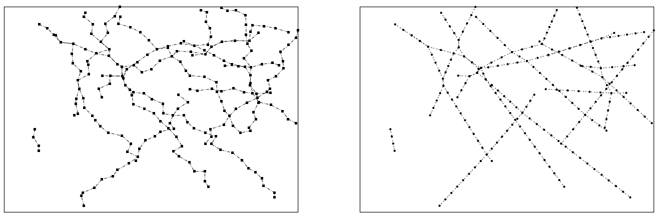

(a) initial configuration (b) final configuration

Figure 16: Comparison of network configurations before and after mobility control.

In this subsection we evaluate the mobility control algorithm for multiple source-destination pairs. The settings of these evaluations are similar to those of the previous section, except that now we have multiple communicating source-destination pairs. As a simplifying assumption, all flows carry the same amount of traffic. Thus, in our algorithm, the number of flows passing through both a node and one of its neighbors determines the weight assigned by the node to that particular neighbor.

We first present the network configurations before and after relay nodes move to their new positions. In these figures, only communicating nodes are shown. The network in fact has a uniform density of nodes. Figure 16 (a) shows the initial configuration of a network with 10 randomly chosen source-destination pairs, and Figure 16 (b) shows the configuration after the mobility algorithm converges. We can clearly see that the mobility control algorithm straightens the lines of communication. For a flow sharing no nodes with other flows (see the flow in the lower left corner), the behavior of the multiflow algorithm is the same as that of a single flow.

While the descent algorithm theoretically performs better than averaging method, we have found empirically that the method results in at best scant power savings over the far simpler averaging rule. This is because there are few junction nodes, these junction nodes typically have at most three neighbors, and our power exponents used are relatively small (3 to 6). In these cases, the target point computed with averaging is very close to the target point reached through the descent direction. For this reason, we use averaging throughout our multiflow evaluations. In nonuniform networks where certain nodes may be choke points for many flows from many asymmetrically located neighbors, we expect the performance improvement from the descent method to be more marked.

We now quantitatively evaluate the performance gain of mobility control for multiple flows. Figures 17 (a) and (b) show the percentage gain in energy efficiency of mobility control of a network with 10 source-destination pairs, under constrained and unconstrained mobility. We observe that even under constrained mobility, the performance improvement is still substantial. The performance improvement is measured as the percent reduction from the energy cost required to send one bit between each source-destination pair in the original network configuration to the cost in the converged post-mobility network. Note that we test here on relatively dense networks communicating over greedy paths. The performance improvements will be substantially higher on sparser networks and on non-greedily determined paths.

Finally, we evaluate the trade-off between mobility energy cost and communication energy cost in networks with multiple source-destination pairs. Figure 17 (c) shows the results for a 15 source-destination pair network. The system traces for the multiflow network look very similar to Figures 12 (a) and (b) so we omit them here. The crossing times show similar behavior to the single flow case in that the cost of mobility is paid off with a small amount of traffic. As before, the constrained rule realizes its performance gains earlier than the unconstrained rule realizes its comparatively larger gains. The crossing times are larger than in the single flow case because there are now more flows, but still correspond to similar numbers of bits per flow before mobility results in performance gains.

5. MOBILITY CONTROL FOR CONCAST NETWORKS

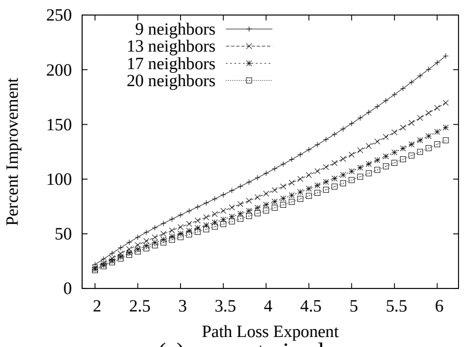
In this section we apply our mobility control algorithm to the multiflow scenario called *concast* in which the destination of all flows is a single *sink* node. Concast matches many notable sensornet applications. For example, concast describes traffic patterns of applications that involve data aggregation or reporting from distributed sensors to a central sink, e.g., [9]. For sensor networks running a geographic hash table [23], all data events of a particular type will be stored at a single location in the network. The traffic pattern we will see in this case is a multisink concast.

5.1 Mobility Control in Concast Network

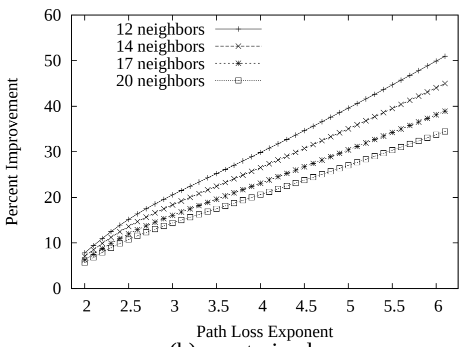
Concast is simpler than the general multiple source-destination pair case because of the single direction of traffic flow. In this case, we can more easily determine the weight of each link. A junction node need only keep track of the amount of traffic sent to it from each of its upstream neighbors and assign these flow rates directly to its upstream weights. The weight of the downstream neighbor is then set to be the sum of all the upstream weights. It is clear that the optimal position of concast junctions sags toward its sink side.

Below, we evaluate the performance of mobility control for concast. The settings of the evaluation in this section are similar to those of previous sections. We again send the same amount of traffic along each flow so that the weight of an upstream neighbor is simply the total number of flows passing through it. Accordingly, the weight of the downstream neighbor is the sum of all upstream weights.

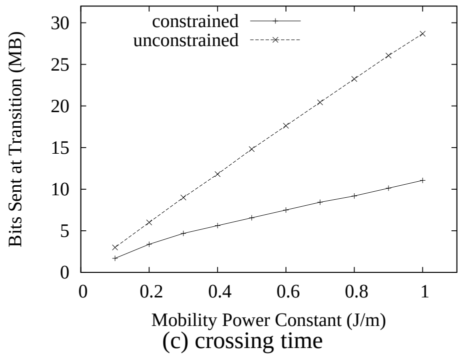
We first evaluate the scenario of a single sink which can be a good model for a surveillance network with a global sink. Figure 18 shows the network configurations before and after applying concast mobility control. The network has a single concast session with 50



(a) unconstrained



(b) constrained



(c) crossing time

Figure 17: Trade-off between mobility energy cost and communication energy cost for multiple unicast flows.

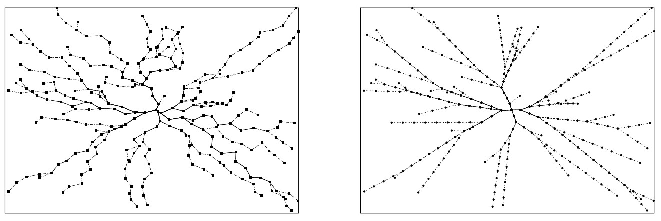

(a) initial configuration (b) final configuration

Figure 18: Network configurations before and after applying concast mobility control: a single concast session with 50 sources.

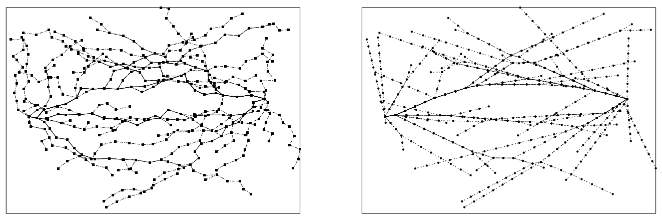

(a) initial configuration (b) final configuration

Figure 20: Network configurations before and after mobility: multisink concast without connectivity constraint: each concast has 30 sources.

sources. We observe that the initially highly irregular concast tree converges to a sharp and regular tree with nodes evenly spaced along its branches.

The effect of the weighting in our algorithm can be seen at the node connected directly to the left of the sink in the final configuration. This node aggregates three flows into one outgoing link. Without weighting, the node would converge to a position much closer to its three upstream neighbors than to the sink. However, because the weight of the downstream neighbor is equal to the sum of the weights of all upstream neighbors, the node converges instead to a central location. Another example of the effect of weighting can be seen in the node connected directly above the sink. This node has two upstream neighbors: the one at left aggregating 10 flows and the one at right aggregating 1. We can see that the converged position of the node is only slightly affected by the presence of its right upstream neighbor.

Figures 19 (a) and (b) quantify the performance gain of mobility control on concast networks. We observe that compared with random source-destination pairs, concast has slightly lower performance gain, due to concentrated traffic patterns. However, the performance gain is still substantial. This substantial improvement in communication cost translates into a lax design requirement for low mobility cost. Figure 19 (c) shows the trade-off between mobility energy cost and communication cost. We observe similar behaviors as those of multiple source-destination pairs. A notable difference however, is that performance gains are realized through mobility after relatively little traffic has been sent. This concast consists of 50 flows, yet exhibits crossing times as little as 10 times those of the single flow scenario. This makes sense if one considers the fact that in concast, many nodes are junction nodes, and their movement simultaneously helps many flows. Hence, in concast, small amounts of mobility tend to result in strikingly large performance gains.

Next, we evaluate the performance improvement of mobility control when there are multiple concast sinks. This can be a good model for many scenarios, for example GHT [23]. Figure 20 shows how configuration changes through mobility and Figure 21 charts the crossing times. We observe similar behaviors to those of a single concast sink.

Overall, these results show that our algorithm is very robust against diverse network environments, achieving performance gains in a wide range of scenarios.

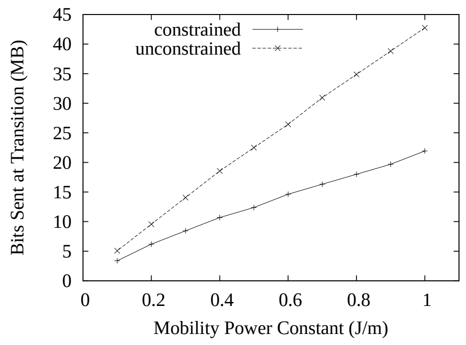
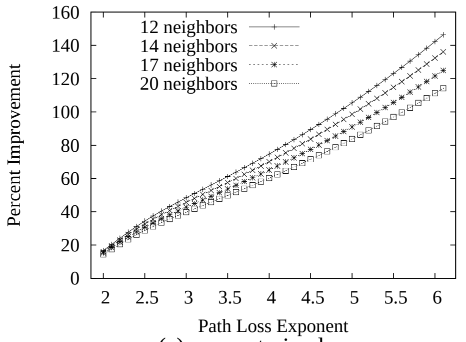


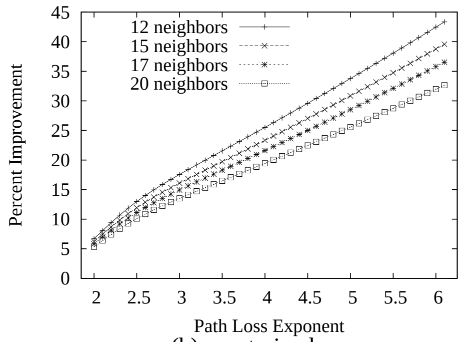
Figure 21: Crossing time: 3 concast sessions with 15 sources in each session.

6. CONCLUSION AND FUTURE WORK

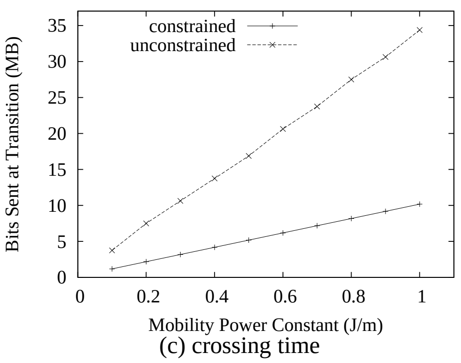
In this paper, we presented the first mobility control scheme for improving communication performance in wireless networks. Our full mobility scheme is completely distributed, requiring each node to possess only local information. The scheme is self-adaptive, being able to transparently encompass several modes of operation. In particular, we showed how our single scheme improves power efficiency for one flow, multiple flows, and many-to-one concast flows. In addition to empirical verification, we have also formally proved the correctness and convergence of our scheme under mild conditions. We also provided evaluations of the feasibility of mobility control and showed that there are many scenarios where controlled mobility can provide substantial performance improvement.



(a) unconstrained



(b) constrained



(c) crossing time

Figure 19: Performance of concast: using averaging for networks of a single concast session, with the session having 50 sources.

Although the full strength of controlled mobility can only be demonstrated by completely functional prototypes, the initial design and analysis in this paper show that mobility can be used as one of the effective control primitives to improve network performance.

The focus of this paper is on power efficiency but one can envision that controlled mobility can be also used to improve many other aspects of network performance. The proposed algorithmic framework in this paper can serve as starting point for such further explorations. For example, one possibility with good potential is to use controlled mobility to improve wireless network capacity. Looking beyond communication, we imagine that our mobility algorithms could be used in conjunction with trajectory-based routing [21] or other routing methods as a *communication-driven* approach to distributed formation of arbitrary spatial layouts of nodes.

In closing, we have elucidated some of the potential uses of controlled mobility in improving network communications. This interface between networking and control theory has been little explored until now and is sure to be a promising and important area for future exploration.

7. REFERENCES

[1] NASA project: Planetary exploration using biomimetics. http://www.oai.org/pages/PlanX.html.

[2] BBC News: Robotic insect takes to the air. http://news.bbc.co.uk/1/hi/sci/tech/1270306.stm, Apr 2001.

[3] BBC News: Robot insect walks on water. http://news.bbc.co.uk/2/hi/science/nature/3126299.stm, Aug 2003.

[4] *Block Island Workshop on Cooperative Control*, Lecture Notes in Control and Information Sciences. Springer Verlag, 2003.

[5] S. Chakraborty, D. K. Y. Yau, and J. C. S. Lui. On the effectiveness of movement prediction to reduce energy consumption in wireless communication (extended abstract). In *Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, San Diego, CA, June 2003.

[6] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks: variations on a theme. In *Med. Conf. on Control and Automation*, Lisbon, Portugal, July 9-13 2002.

[7] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of The Ninth International Conference on Mobile Computing and Networking (Mobicom)*, San Diego, CA, Sept 2003.

[8] DARPA. Self-healing minefi eld. http://www.darpa.mil/ato/programs/SHM/.

[9] D. Estrin, J. Heidemann, R. Govindan, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of The Fifth International Conference on Mobile Computing and Networking (Mobicom)*, Seattle, WA, Nov. 1999.

[10] M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad-hoc wireless networks. In *Proceedings of IEEE INFOCOM '00*, pages 1360–1369, Tel Aviv, Israel, Mar. 2001.

[11] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-effi cient communication protocols for wireless microsensor networks. In *Proceedings of Hawaaian International Conference on Systems Science*, January 2000.

[12] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice, Fourth Edition*. Springer-Verlag, 1997.

[13] R. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, New York. NY, 1995.

[14] T.-C. Hou and V. O. Li. Transmission range control in multihop packet radio networks. *IEEE Transactions on Communications*, 34(1):38–44, Jan 1986.

[15] D. L. Hu, B. Chan, and J. W. M. Bush. The hydrodynamics of water strider locomotion. *Nature*, 427(7):663–667, August 2003.

[16] A. Jadbabaie, J. Lin, and A. Morse. Coordination of groups of autonomous mobile agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[17] A. Jardosh, E. Belding-Royer, K. Almeroth, and S. Suri. Towards realistic mobility models for mobile ad hoc networks. In *Proceedings of The Ninth International Conference on Mobile Computing and Networking (Mobicom)*, San Diego, CA, Sept 2003.

[18] A. M. Ladd, K. E. Bekris, A. Rudys, L. E. Kavraki, D. S. Wallach, and G. Marceau. Robotics-based location sensing using wireless Ethernet. In *Proceedings of The Eighth International Conference on Mobile Computing and Networking (Mobicom)*, Atlanta, GA, Nov. 2002.

[19] Q. Li and D. Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *Proceedings of The Sixth International Conference on Mobile Computing and Networking (Mobicom)*, Boston, MA, Aug. 2000.

[20] J. Lin, A. Morse, and B. Anderson. Multi-agent rendezvous problem. In *Proceedings of the 42nd IEEE CDC*, Dec 2003.

[21] D. Niculescu and B. Nath. Trajectory-based forwarding and its applications. In *Proceedings of The Ninth International Conference on Mobile Computing and Networking (Mobicom)*, San Diego, CA, Sept 2003.

[22] C. Perkins. *Ad Hoc Networking*. Addison-Wesley, 2000.

[23] A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proceedings of The Ninth International Conference on Mobile Computing and Networking (Mobicom)*, pages 96–108, San Diego, CA, Sept 2003.

[24] V. Rodoplu and T. Meng. Minimum energy mobile wireless networks. In *Proceedings of IEEE ICC*, Atlanta, GA, June 1998.

[25] E. M. Royer and C. K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, pages 46–55, April 1999.

[26] I. Stojmenovic and X. Lin. Power-aware localized routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(11):1122–1133, November 2001.

[27] P. A. Tipler. *Physics For Scientists and Engineers*. Worth Publishers, 3rd edition, 1991.

[28] J. Yoon, M. Liu, and B. Noble. Sound mobility models. In *Proceedings of The Ninth International Conference on Mobile Computing and Networking (Mobicom)*, San Diego, CA, Sept 2003.

[29] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of The First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, Los Angles, CA, Nov. 2003.