ABSTRACT

We seek to support communications in highly-partitioned mobile wireless networks via controllable data ferries. While existing ferry control techniques assume either stationary nodes or complete ferry observation of node locations, we address the more challenging scenario of highly mobile nodes and partial ferry observations. Using the tool of Partially Observable Markov Decision Processes (POMDP), we develop a comprehensive framework where we expand the solution space from predetermined trajectories to policies that can map ferry observations to navigation actions dynamically. Under this framework, we present an optimal and several efficient heuristic policies. We compare the proposed policies with predetermined control through analysis and simulations with respect to multiple node mobility parameters including speed, locality, activeness, and range of movement. The comparisons show a significant performance gain of up to twice the contact rate in cases of high uncertainty. In cases of low uncertainty, we give a sufficient condition under which predetermined control is optimal.

Categories and Subject Descriptors
C.2.1 [Computer-Communication Networks]: Network Architecture and Design—store and forward networks, wireless communication; G.4 [Mathematical Software]: algorithm design and analysis; I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search—control theory; I.2.9 [Artificial Intelligence]: Robotics—autonomous vehicles

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 2010 ACM 978-1-4503-0183-1/10/09 ...$10.00.

General Terms
Algorithms, Performance, Theory

Keywords
Data ferry control, Partially Observable Markov Decision Processes

1. INTRODUCTION

The emerging application of Mobile Ad Hoc Networks (MANETs) in mission-critical environments brings new challenges. Consider a MANET consisting of highly mobile nodes covering a large, infrastructureless field. Examples of such networks can be found in military tactical operations, search-and-rescue scenarios, as well as mobile sensor networks. Such networks exhibit properties drastically different from traditional networks: low density, high mobility, and poor connectivity. On top of these, the mobility in these networks is usually dictated by their mission needs and physical constraints such as roads and obstacles, which may not provide sufficient contact opportunities. These challenges make it extremely difficult to enable efficient communications via in-network routing and/or replications. Meanwhile, existing work, e.g., [2], has shown that it is feasible to build lightweighted robots that only contain off-the-shelf elements and can be mounted on various types of mobile platforms including unmanned ground or aerial vehicles. The advance in robotic technology and the challenges in mission-critical MANETs inspire our investigation of applying autonomous data ferries in highly-partitioned mobile wireless networks.

Specifically, we propose to enhance these networks with dedicated communication nodes acting as data ferries. We envision a heterogeneous network structure as in [15], where the network consists of regular nodes that passively move according to their inherent mobility patterns and data ferries that actively move according to external control. The regular nodes have communication needs among themselves, while the data ferries only serve as relays in a “load-carry-and-deliver” manner. To fully utilize these ferries, however, effective control of their mobility is needed.

The problem of ferry mobility control has been extensively studied in the case of stationary regular nodes with known locations [1, 5, 15]. The solutions there involve designing ferry routes to cover the nodes and letting the ferries follow these routes deterministically. Little work has, however, been done in the case of mobile nodes, with the only existing solutions to our knowledge [13, 14] assuming complete observation of nodes’ current locations by the ferries. In this paper, we address the cases of highly mobile nodes and partial ferry observations.
practice, this assumption may fail due to obstacles, limited communication range, or poor channel conditions.

We consider the scenario when nodes move stochastically, and data ferries can only observe a subset of the field at a time, referred to as partial observations. Without complete observations, the ferries will have uncertainties about node locations. Our goal is to develop a solution that enables the ferries to navigate themselves intelligently based only on partial observations and statistical knowledge of node mobility.

1.1 Related Work

The idea of using controlled mobility to facilitate communications was first proposed in [7], where nodes deliberately change trajectories to send messages to otherwise disconnected nodes. This solution has limited application in mission-critical networks since regular nodes are often not allowed to change trajectories arbitrarily. For such networks, designated communication nodes are used instead [5, 6, 12–15]. If there are sufficiently many communication nodes, then one can use these nodes to form a connected backbone that covers all the task nodes [12]; however, such solutions will be inefficient in sparse networks. Otherwise, a delay-tolerant scheme is used, where communication nodes physically carry data between task nodes [5, 13–15] or from sensors to base stations [1, 6]. Our work belongs to this category. The existing work is limited by several assumptions: most solutions assume the task nodes to be stationary with known positions [1, 5, 6, 15]; when task nodes are mobile, the solutions assume that ferries know nodes’ current locations through out-of-band communications [14] or location management systems [13]. There have been extensions in terms of channel models, performance metrics, traffic demands, etc., but the main assumption of fully observable network state remains. In contrast, we aim to handle stochastic node movements and incomplete ferry observations.

Technically, our approach belongs to stochastic control and in particular Partially Observable Markov Decision Processes (POMDP) [10]. Although previously applied to agent navigation in robotics [3, 9], its application to data ferry mobility control has not been explored before. Our problem differs from the agent navigation problem in that the subsystem that evolves dynamically (task nodes) is different from the subsystem under control (data ferries). In [6], a related technique of Markov Decision Process (MDP) has been used for ferry control, although it still assumes stationary task nodes and fully observable network state. Another difference is that [6] focuses on node buffer states (also fully observable), whereas we focus on the uncertainty in node locations.

1.2 Our Approach and Results

We take the approach of stochastic control, where in contrast to predetermined control in stationary networks, we show that the optimal control in mobile networks needs to be dynamic in nature. Specifically, our contributions are:

- A comprehensive control framework: We model the entire network of nodes and ferries as an interactive system represented by a Partially Observable Markov Decision Process (POMDP), which seamlessly combines the tasks of information collection and decision making. The framework enables a systematic solution to handle uncertainties due to stochastic node movements and partial ferry observations.

- Optimal and efficient heuristic policies: Based on the proposed framework, we develop both optimal and heuristic policies by applying POMDP techniques. We develop a recursive algorithm to compute the optimal policy and a set of efficient heuristic algorithms due to the complexity of the optimal policy. Moreover, we show that our problem has a unique structure that allows an explicit policy representation linked back to ferry trajectories.

Analysis and simulations: We conduct rigorous analysis of the proposed solution verified by simulations. In particular, since dynamic control generally requires higher costs than predetermined control, we want to characterize their performance gap and how it is influenced by external parameters to provide general guidance on system design. Using ferry-node inter-contact time as the performance measure, we derive both upper bounds (achievability) and lower bounds (converse). Our achievability results show promising performance of the proposed dynamic control in that it outperforms predetermined control the most when improvement is most needed: the field is large (or the sensing range is small) and the prior knowledge is minimal (nodes’ steady-state distributions are close to uniform). In this case, our solution can reduce the inter-contact time by at least \( \Omega(\sqrt{r}) \) for a domain of size \( n \), and even \( \Theta(n) \) if nodes are constrained to 1-D routes. The converse shows that this reduction is at most half, achievable by our policy under symmetric and infrequent mobility. On the other hand, when nodes’ steady-state distributions are localized, we also give a sufficient condition under which predetermined control is optimal.

The rest of the paper is organized as follows. Section 2 formulates the problem of dynamic ferry control. Section 3 presents a control framework via POMDP. Section 4 develops the optimal and heuristic policies, whose performance is analyzed in Section 5. Section 6 extends the solution to more general settings. Section 7 evaluates the solution through simulations, and Section 8 concludes the paper.

2. PROBLEM FORMULATION

2.1 Network Model

As illustrated in Fig. 1, we assume that the network of regular (i.e., non-ferry) nodes is divided into \( N \) partitions, each occupying a disjoint region called a domain. Within each domain, at least one node is selected as the gateway for inter-domain communications (assuming sufficient intra-domain contacts). We make this distinction because selected nodes may need special equipments (e.g., high-power antennas) to communicate with ferries, although for our purpose it suffices to model only these gateways, which will simply be called “nodes” in the sequel (the density and mobility of non-gateway nodes are irrelevant to our problem of data ferry control). Each node is equipped with a radio of range \( r \) and moves within its domain according to some stochastic mobility model to be specified later.

Assume that each data ferry is equipped with a radio of the same range as the nodes\(^1\). In addition, it has a buffer to store messages and a controller to specify its movements according to a control policy \( \pi \) (defined later). Assume the ferry self-navigates periodically with a predetermined interval (called slot length), and at the end of each slot \( t \), it senses node’s presence (e.g., by broadcasting beacons) to obtain a binary observation \( z_t \) (\( z_t = 1 \) for contact and 0 for miss). Assume the sensing range to be equal to the radio range. Upon

\(^1\)Equivalently, \( r \) denotes the smaller of ferry/node ranges.
and the number of domains
consider the case when there is one (gateway) node per domain

3. DESIGN FRAMEWORK

sophisticated solutions involve optimization of certain performance metrics. Since the domains are fixed, we can view an entire domain as a stationary “node” and apply existing solutions for ferry route design [15] to order the domains.

Global control: The global control determines which domain to go to. An intuitive solution is to iterate among the domains when there is a miss, which leads to oscillating movements and can be shown to yield a contact rate of $2(p + q)/(2 + p + q - 2(1 - p - q)(1 - pq))$. Another intuitive solution is to let the ferry wait in one cell, whose contact rate is $2(p + q)/p^2 + q^2 + 2pq(p + q)$. None of these policies can guarantee optimality, e.g., the waiting policy is better if $p = q > 0.5$, and the oscillating policy is better if $p = q < 0.5$. A smarter design is perhaps to keep chasing the node to its most likely cell, but even this policy is not always optimal (e.g., it is worse than the oscillating policy for $p = 0.25, q = 0.5$). In the sequel, we will introduce a systematic approach to achieve optimality.

3.1 A Simple Example

To illustrate the difficulty of this problem, let us consider a simple example. Suppose that each domain is only partitioned into 2 cells, labeled cell 1 and cell 2. In each slot, a node will move from cell 1 to 2 with probability $p$ and from cell 2 to 1 with probability $q$. Suppose we want to maximize the contact rate (average number of contacts per slot).

Simple as it seems, much thought is required to achieve optimality. A baseline policy of random movements will give a contact rate of 0.5. An intuitive alternative is to check the other cell whenever the ferry has a miss, which leads to oscillating movements and can be shown to yield a contact rate of $2(p + q)/(2 + p + q - 2(1 - p - q)(1 - pq))$. Another intuitive solution is to let the ferry wait in one cell, whose contact rate is $2(p + q)/p^2 + q^2 + 2pq(p + q)$. None of these policies can guarantee optimality, e.g., the waiting policy is better if $p = q > 0.5$, and the oscillating policy is better if $p = q < 0.5$. A smarter design is perhaps to keep chasing the node to its most likely cell, but even this policy is not always optimal (e.g., it is worse than the oscillating policy for $p = 0.25, q = 0.5$). In the sequel, we will introduce a systematic approach to achieve optimality.

3.2 A Systematic Solution via POMDP

The preceding “simple” example turns out to require a sophisticated solution [8]. In general, the problem is even trickier since the ferry will not know the exact location of the node before contact. In this section, we introduce a systematic design framework using the POMDP theory [10].

POMDP models a control problem where the goal is to select control actions based on run-time observations reflecting the state of the controlled system to optimize reward. A POMDP problem is represented by the following tuple $< \mathcal{S}, \mathcal{U}, \mathcal{Z}, \mathcal{P}, \mathcal{P}_z, r, \gamma, b_0 >$ [3]:

1. $\mathcal{S}$: the state space, defined as the set of states of the controlled system; here the state $s_t$ denotes the cell of the current node at the beginning of slot $t$ and the state space $\mathcal{S} := \{1, 2, \ldots, n\}$ is the set of all cells;
2. $\mathcal{U}$: the action space, defined as the set of control options; here the action $u_t$ denotes the cell the ferry will move to in slot $t$, either in the current domain ($u_t \in \mathcal{U}_t := \{1, \ldots, n\}$ means to follow the current node to cell $u_t$) or in a new domain ($u_t \in \mathcal{U}_n := \{1', \ldots, n'\}$ means to switch to cell $u_t$ of the next domain specified by the global control), and $\mathcal{U} = \mathcal{U}_t \cup \mathcal{U}_n$;
3. $\mathcal{Z}$: the observation space, defined as the set of possible observations; here $\mathcal{Z} = \{0, 1\}$, with 1 denoting “contact” and 0 “miss” ($z_t \in \mathcal{Z}$ is taken at the end of slot $t$);
4. $\mathcal{P}$: the state transition model specifying the transitions of the system state; here it is the transition matrix for node mobility (assuming Markovian mobility), where $P(t, j) = \Pr(s_{t+1} = j | s_t = i)$ is the probability for the node to move from cell $i$ to cell $j$ in slot $t$;
5. $\mathcal{P}_z$: the observation model specifying how observations are related to states and actions; here $P_z(s, u, z) := \Pr(z_t = z | s_{t+1} = s, u_t = u)$ is the probability of
sensing \( z \) if the node moves to cell \( s \) and the ferry to cell \( u \); we assume \( P_t(s, u, z) = 1 \) if and only if \( z = l_{s(u)} \); 
6. \( r(z, u) \): the payoff function specifying the reward of taking action \( u \) and observing \( z \); here we use \( r(z, u) = z \) to give a unit reward for each contact; 
7. \( \gamma \in (0, 1] \): the discount factor specifying how much delayed rewards will be discounted; the total reward is given by \( ^{\gamma}R(z_t, u_t^t) = \sum_{t=1}^{\infty} \gamma^t r(z_t, u_t) \); 
8. \( b_0 \): the initial belief specifying the initial distribution of the system state; here we assume it is the node’s steady-state distribution \( b_0 \) (assuming it exists).

Many of the above elements are design choices and can be extended (see Section 6 and 8). The main assumption is the Markovianess of node mobility. While noting it as a limiting assumption, we point out that most mobility models, e.g., random walk, random waypoint, and their variations, can be mapped to Markovian models under proper quantization.

4. OPTIMAL AND HEURISTIC POLICIES

We now develop policies for the local control under the framework in Section 3.2 (extension to joint control will be presented later in Section 6.2). We will first develop a mathematical representation of the optimal policy and then investigate the algorithms. For clarification, we assume nodes in different domains move i.i.d. with transition matrix \( P \) and steady-state distribution \( b_0 \). The non-i.i.d. case can be handled by plugging in different matrices and distributions.

4.1 Optimal Policy

It is known in POMDP theory that a sufficient statistic is the belief vector \( b_t \), defined as the posterior distribution of the node (at the beginning of slot \( t \)) given all past actions and observations, i.e., \( b_t(s) = P_T(s_t = s | s_0, u^{t-1}, z^{t-1}) \). It thus suffices to design policy based on \( b_t \). We introduce an auxiliary function \( V_T \) called the value function, where \( V_T(b) \) denotes the total discounted reward gained by policy \( \pi \) over a horizon \( T \) starting from belief \( b \). It is known that the optimal value function must satisfy the following equation known as the value iteration:

\[
V_T(b) = \gamma \max_{u \in U} \left[ r(b, u) + \sum_{z \in Z} P_b(b, u, z)V_{T-1}(f(b, u, z)) \right],
\]

where \( r(b, u) \) is the expected reward in the current slot, \( P_b(b, u, z) \) the probability of observing \( z \), and \( f(b, u, z) \) the belief after taking action \( u \) and observation \( z \). The optimal policy is the solution to (2), i.e., \( \pi_T(b) \) is the action \( u_T \) achieving \( V_T(b) \).

We now translate the above definitions to our problem. In our problem, the belief vector experiences two updates per slot: a belief transition and a Bayesian update. The belief transition takes place after the ferry takes an action \( u_t \) at the beginning of slot \( t \) and before it takes an observation \( z_t \) at the end of slot \( t \) to model node movements during this slot:

\[
b_t' = f_t(b_t, u_t) \left\{ \begin{array}{ll}
P^t_{b_t}b_t & \text{if } u_t \in U_f, \\
b_0 & \text{if } u_t \in U_s,
\end{array} \right.
\]

where the belief is reset if the ferry switches to a new domain. The Bayesian update occurs after the ferry observes \( z_t \):

\[
b_{t+1} = f_2(b'_t, u_t, z_t) \left\{ \begin{array}{ll}
b_0 & \text{if } z_t = 1, \\
(b_{t}(u_t) & \text{if } z_t = 0,
\end{array} \right.
\]

and the action achieving (5) gives the optimal policy. Physically, \( V_T(b) \) is the maximum discounted number of contacts over \( T \) slots starting from a node distribution \( b \). Although \( \gamma \) does not appear in the physical problem, it is needed to ensure a unique optimal solution in the long run.

FACT 4.1 ([11]). There exists a unique stationary policy \( \pi_\infty \) that optimizes the total reward as \( T \to \infty \) if \( \gamma < 1 \), and its value function \( V_\infty \) is invariant under (2).

4.2 Policy-Computing Algorithms

Although mathematically sound, equation (5) cannot be directly used to compute the policy as there are infinitely many belief vectors. In this section, we explore algorithms that can compute the policy in a finite number of steps. Note that since we currently focus on local control in the 2-tiered control hierarchy (see Section 3), the policies and their complexities are independent of the number of domains (see Section 6.2 for extension).

4.2.1 Hardness of Optimal Policy

Sondik [10] first showed the possibility of computing the optimal policy by utilizing the piece-wise linear and convex property of the optimal value function: \( V_T(b) = \max_{\alpha \in \Gamma_f} \langle b \cdot \alpha \rangle \).

where \( \alpha \) is a coefficient vector, called the value vector, that represents a linear segment of \( V_T \). The problem is then reduced to finding the set of value vectors \( \Gamma_f \subseteq \mathbb{R}^n \). Our problem differs from the classic POMDP in that we have two belief updates per slot (3-4) that are interleaving with the action. Consequently, our optimal value function has a unique form as follows (see [4] for proofs).

PROPOSITION 4.2. The optimal value function has the form

\[
V_T(b) = \max_{\alpha \in \Gamma_f} \left( \max_{\alpha' \in \Gamma'} \max_{\alpha \in \Gamma_f} \langle b \cdot \alpha \rangle \right).
\]

Moreover, \( \Gamma_f \) and \( \Gamma' \) (\( T \geq 1 \)) satisfy the recursion\(^6\):

\[
\Gamma_{f+1} = \tilde{\Gamma}_{f+1} \cap \{ \alpha : \alpha' \in \Gamma' \},
\]

where \( \tilde{\Gamma}_{f+1} := \{ \tilde{\alpha} \alpha' : \alpha \cdot \tilde{\alpha} = \alpha' \} \), and \( \tilde{\Gamma}_{f+1} := \{ \tilde{\alpha} u, \tilde{\alpha}' u : \forall u \in S, \alpha' \in \Gamma_{f+1} \} \) for

\[
\tilde{\alpha} u, \alpha'(s) = \left\{ \begin{array}{ll}
(1 + c_1) \gamma(s) & \text{if } s = u, \\
\gamma(s) & \text{otherwise},
\end{array} \right.
\]

\(^5\)The notation \( b' = b_{1, u} \) means \( b'(u) = 0 \) and \( b'(s) = b(s)/(1 - b(u)) \) for \( s \neq u \).
\[ \alpha^u(s) = \begin{cases} \gamma(1 + c_1) & \text{if } s = u, \\ \gamma c_2 & \text{o.w.,} \end{cases} \]

(c_1 := \max_{\alpha' \in \Gamma_T^{-1} \cup \Gamma_T^{-1}} b_0 \cdot \alpha', \text{ and } c_2 := \max_{\alpha' \in \Gamma_T^{-1}} b_0 \cdot \alpha'.)

The above result also gives us an explicit algorithm to compute the optimal policy. We start with \( \Gamma_T^0 = \Gamma_T^0 = 0 \) and iteratively apply (7) for \( T = 1, 2, \ldots \) to compute \( \Gamma_T^T \) and \( \Gamma_T^{-1} \) for any given \( T \). When applying the policy, we select the action that maximizes the consolidated functions on the right-hand side of (5), where \( V_{T-1}(b_0) \) and \( V_{T-1}(b_{u'}) \) are computed from \( \Gamma_T^{-1} \) and \( \Gamma_T^{-1} \) by (6).

The drawback of the above algorithm is that the number of value vectors grows fast. It can be shown that \( |\Gamma_T^T| = |S|(|S|^T - 1)/(|S| - 1) = O(|S|^T) \) (recall: \( |S| = n \); see Section 3.2) and \( |\Gamma_T^{-1}| = 1 \), and thus the worst-case complexity grows exponentially with \( T (O(|S|^{T+2})) \). This is not specific to our algorithm, but due to the intrinsic hardness of POMDP [3].

### 4.2.2 Efficient Heuristic Policies

Due to the high complexity of the optimal policy, we turn to heuristic policies. Most heuristic policies seek to approximate the optimal value function. For example, we can approximate the exact value iteration (5) by the Interpolation-Extrapolation method [3], which quantizes the belief space into a finite set of representative belief vectors and runs value iterations only on these beliefs; the values at other beliefs are computed via interpolation. Alternatively, we can approximate the value-vector-based algorithm in Section 4.2.1 by the Point-Based Value Iteration method [9], which limits the set of value vectors to those achieving the maximum value in (6) for at least one of the representative beliefs. Other heuristic methods include reducing the problem to a fully observable case (MDP and QMDP approximations), directly approximating the value function by curve fitting (Least-Squares Fit), etc.; see [3,9] and references therein.

An extreme case is when we ignore the value function altogether, which leads to a much simpler policy, called the myopic policy, that greedily optimizes the probability of immediate contact:

\[ \pi_1(b) = \arg \max_{u \in U} b'(u). \] (10)

Unlike the above sophisticated policies, this policy does not require any offline computation and has very short online response time. On the other hand, it may suffer more performance loss due to greediness. In certain cases, however, we show that the myopic policy is optimal (proof in [4]).

**Theorem 4.3.** At horizon \( T \) and belief state \( b \), the myopic policy is optimal if

\[ 1 - \gamma \geq \frac{1 - b^{(1)}_1}{1 + b^{(2)}_0(1 - \gamma^{T-1})/(1 - \gamma)}, \] (11)

where \( b^{(1)}_1 = \max_{u \in U} b'(u) \) (achieved at \( u_1 \)) is the largest immediate reward, \( b^{(2)}_2 = \max_{u \in U \cup \{u_1\}} b'(u) \) the second largest immediate reward, and \( b^{(1)}_0 = \max_{u \in U} b_0(u) \).

This theorem gives a sufficient condition for optimality of the myopic policy. The condition depends on two parameters: the number of remaining slots \( T \) and the current belief \( b \). We can safely skip value iterations and follow the myopic policy in subsets of the belief space where (11) holds.

### 4.2.3 Explicit Policy Representation

Different from general POMDPs, our problem has a special property as follows (proof in [4]).

**Proposition 4.4.** Each belief-based dynamic policy \( \pi \) corresponds to a unique trajectory \( u^\pi := (u^\pi(t))_{t=1}^\infty \) (\( d \) can be \( \infty \)) within one domain such that \( \pi \) is equivalent to a policy of following \( u^\pi \) and repeating it in different domains until contact, after which this process is repeated in a new domain.

This is because the ferry’s observations are binary and the controller restarts (in a new domain) after each contact, which reduces a policy to a sequence of actions \( (u^\pi) \) if the observations are consecutive misses. This property allows us to represent the policy explicitly by \( u^\pi \). Note that this is different from predetermined ferry trajectories because instead of following \( u^\pi \) deterministically, the ferry will stop and switch domains immediately after a contact, and its actual trajectory will be stochastic. To distinguish from the actual trajectory, we will call \( u^\pi \) the policy trajectory of policy \( \pi \).

### 5. PERFORMANCE ANALYSIS

We analyze the performance of the proposed policies using the measure of average inter-contact time \( E[K] \), \( i.e., \) the mean number of slots between two consecutive ferry-node contacts \((1/E[K] \) is the contact rate). Our goal is twofold. First, we want to compare dynamic policies with predetermined policies. Since dynamic policies are more expensive to implement, the comparison will provide guidance on performance-cost tradeoff. Moreover, we want to see how the performance is influenced by node mobility models. Instead of limiting to specific models, we focus on generic parameters such as speed, locality, activeness, and the size and the shape of the field to draw insights on when dynamic control is worthwhile and when predetermined control suffices.

#### 5.1 Dynamic vs. Predetermined Control

To analyze the fundamental performance gain, we need to compare the optimal dynamic policy with the optimal predetermined counterpart. We begin with the following observation (proof in [4]).

**Claim 5.1.** The optimal predetermined policy is to keep switching among the most likely cells of different domains (called the switching policy).

Therefore, it suffices to compare dynamic policies with the switching policy. In the sequel, we use the notation that \( (b^{(i)}_0)_{i=1}^\infty \) is an ordered sequence of the elements in \( b_0 \) such that \( b^{(1)}_0 \geq b^{(2)}_0 \geq \ldots \). Let \( K^{SW} \) denote the inter-contact time of the switching policy. It is easy to see that \( E[K^{SW}] = 1/b^{(1)}_0 \). For dynamic policies, an exact analysis is intractable as we cannot even write down the optimal policy explicitly (see Section 4.2.1). Instead, we seek to bound its performance from both above and below.

For the upper bound, we need to analyze a good but simple dynamic policy. Specifically, consider the myopic policy with policy trajectory \( u = (u(t))_{t=1}^\infty \), and let \( p_t \) denote the conditional probability of contact in step \( t \) given that there has been no contact in the previous \( t-1 \) steps. Then \( p_t \) and \( u(t) \) can be computed recursively by

\[ p_t = \max_s b_t^*(s), \quad u(t) = \arg \max_s b_t^*(s), \] (12)
where $b_{t+1} = P^T(b_t)\gamma(t)$ and $b_t := b_0$; $d$ is the smallest $t$ such that $p_{t+1} < b_t^{(1)}$.

For the lower bound, we need to modify the problem into one that allows superior performance. Consider a hypothetical scenario where all nodes are stationary with distribution $b_0$ such that the ferry never needs to visit a cell twice. The policy trajectory in this case is the sequence of cells corresponding to $b_0^{(1)}, b_0^{(2)}, \ldots$. The conditional contact probability $p_t$ is given by $p_t = b_t^{(1)} / \left(1 - \sum b_t^{(i)}\right)$, and the trajectory length $d'$ is the smallest $t$ such that $p_{t+1} < b_t^{(1)}$. With these notations, we present the following bounds (proof in [4]).

**Theorem 5.2.** The average inter-contact time $E[K^{dy}]$ of the optimal dynamic policy satisfies

$$K_0' + d'(1 - p_0)/p_0 \leq E[K^{dy}] \leq K_0 + d(1 - p_0)/p_0,$$

where $p_0 := 1 - \prod_{t=1}^d (1 - p_t)$, $K_0 := d\sum_{t=1}^d p_t \prod_{i=1}^{t-1} (1 - p_i)$, and $p_0, K_0'$ are similarly defined with $p_t, d$ replaced by $p_t', d'$.

This theorem provides both achievability and converse results on dynamic ferry control. The achievability result (the upper bound) gives a guaranteed performance gain in comparison with predetermined control ($E[K^{sw}]$). The converse result (the lower bound) establishes a fundamental limit that no control policy can outperform, which can serve as a reference in evaluating the effectiveness of specific policies.

### 5.2 Impact of Domain Size & Node Speed

We now apply Theorem 5.2 to analyze the impact of node mobility parameters. We first examine the impact of domain size, measured by the number of cells $n$ per domain, and node speed, measured by the number of cells $m$ a node can move across per slot (i.e., it can move to locations at most $m-1$ cells away). In this subsection, we consider a symmetric mobility model where nodes are uniformly distributed in the steady state to only focus on $n$ and $m$; asymmetric mobility models will be treated separately in Section 5.3.

Intuitively, the ferry starts with little knowledge about node location, but will learn more as it visits cells and makes its belief vector more concentrated. However, due to node movements, if the ferry misses the node in one cell, its probabilities of seeing the node in neighboring cells will also drop in the next slot, and this effect will propagate to further neighbors over time. To compensate for this effect, we make the ferry fallow a dominating trajectory, defined as a sequence of cells $c(1), c(2), \ldots$ such that no $c(k)$ can be reached by a node starting from cell $c(j) k-j$ slots ago. The idea is to let the ferry scan the field by visiting only one cell per neighborhood to best utilize the contact opportunities; see Fig. 2 for illustrations. Using this policy, we obtain the following bounds (proof in [4]).

**Corollary 5.3.** If $b_0$ is uniform, then $E[K^{sw}] = n$, and

$$\frac{n}{2} \leq E[K^{dy}] \leq n - \frac{d(n)}{2}, \quad (14)$$

where $d(n)$ is the length of the longest dominating trajectory.

The above gives closed-form bounds on control performance in terms of domain size and the length of dominating trajectory. Intuitively, the more stretched the domain is and the slower the node moves, the longer the dominating trajectory, and the better dynamic policies will perform. For example, for random walks (i.e., $m = 2$), $d(n) \approx n/2$ if the domain is 1-D (trajectory 1, 3, \ldots) and $d(n) \approx \sqrt{n}$ if 2-D (see Fig. 2). In general, it can be shown that $d(n) = \Theta(n/m)$ in 1-D and $d(n) = \Omega(\sqrt{n}/m)$ in 2-D. This result is consistent with the intuition that it is easier to catch nodes that move slowly or in constrained domains. Meanwhile, since a predetermined controller only uses the steady-state distribution, its performance is independent of the above parameters.

As the domain size $n$ increases (for fixed $m$), the absolute performance gain $E[K^{sw}] - E[K^{dy}]$ will always increase, at least as $\Omega(\sqrt{n})$. Moreover, if nodes are constrained to some routes (1-D mobility), then even the relative gain $(E[K^{sw}] - E[K^{dy}])/E[K^{sw}]$ will be positive, i.e., there will be a positive-fraction improvement in the contact rate. This is significant given the fact that the contact rate can at most be doubled as seen from the lower bound. Although not proved for 2-D mobility, simulations show that the upper bound is loose in this case and dynamic control still provides significant improvement in contact rate (see Section 7).

### 5.3 Impact of Node Locality

As node mobility becomes asymmetric, the steady-state distribution $b_0$ will deviate from the uniform distribution. After a certain point, $b_0$ alone can already give a good prediction of node location, and dynamic control becomes unnecessary. We confirm the above intuition by the following result (proof in [4]).

**Corollary 5.4.** If the node mobility is sufficiently biased such that $b_0^{(1)} > b_0^{(2)}/(1 - b_0^{(1)})$, then $E[K^{dy}] = E[K^{sw}]$, i.e., the switching policy is optimal.

This result gives a sufficient condition for optimality of the switching policy. If this condition holds, it is optimal to hop among the most probable cells of different domains deterministically, and no dynamic control is needed.

### 5.4 Impact of Node Activeness

Intuitively, there are two aspects of the level of mobility: how far a node can move in one slot and with what probability. The first aspect reflects node speed and has been discussed in Section 5.2; the second reflects how active the nodes are and will be the focus of this subsection.

To model node activeness, we start from a base transition matrix $P$ and scale all the off-diagonal entries by a parameter $\beta > 0$, called the activeness parameter. We then modify the diagonal entries to make it a valid transition matrix again, i.e., the new matrix $P_\beta$ is given by $(P_\beta)_{ii} = \beta P_{ij}$ for $i \neq j$ and $(P_\beta)_{ii} = 1 - \beta \sum_{j \neq i} P_{ij}$, where
0 < \beta < 1/[\max_{j \neq i} P_{ij}]. Intuitively, \beta models how frequently nodes cross cell boundaries during one slot. It is orthogonal to the steady-state distribution (proof in [4]).

**Lemma 5.5.** Let \( \mathbf{P} \) be a base transition matrix and \( \mathbf{P}_\beta \) its scaled version according to the activeness parameter \( \beta \). Then their associated steady-state distribution \( \mathbf{b}_0 \) and \( \mathbf{b}_0 \) are equal for all \( \beta \in (0, 1/(1 - \min_i P_{ii})) \).

Since \( \mathbf{b}_0 \) is independent of \( \beta \), the previous results that only depend on \( \mathbf{b}_0 \), e.g., the performance of the switching policy and the lower bound in (13), are also independent of \( \beta \). Although the general upper bound in (13) still holds here (with \( \mathbf{P} \) replaced by \( \mathbf{P}_\beta \)), we can narrow the gap by explicitly including \( \beta \) as follows (proof in [4]).

**Theorem 5.6.** If we scale the transition matrix in Theorem 5.2 by the activeness parameter \( \beta \), then besides the bounds in (13), \( \mathbb{E}[K^{DY}] \) also satisfies \( \mathbb{E}[K^{DY}] \leq \hat{K}_0 + d'(1 - \hat{\rho}_0)/\hat{\rho}_0 \), where

\[
\hat{\rho}_0 := \sum_{t=1}^{d'} \hat{b}_0^{(t)} - \frac{\beta}{2} \sum_{t=1}^{d'} \hat{b}_0^{(t)}(d' - t)(d' - t + 1),
\]

\[
\hat{K}_0 := \frac{1}{\rho_0} \sum_{t=1}^{d'} t \left( \hat{b}_0^{(t)} - \frac{\beta}{2} \sum_{i=1}^{t-1} \hat{b}_0^{(i)}(t - i) \right),
\]

for \( \delta = \max_{i \neq j} P_{ij} \) and \( d' \) as defined in Theorem 5.2. As \( \beta \to 0 \), \( \mathbb{E}[K^{DY}] \) converges to the lower bound at \( O(\beta) \).

Besides verifying the intuition that as nodes become less active, it will be easier to infer node locations from past observations and the control performance will improve, the above result also guarantees that the convergence rate is (at least) linear in \( \beta \), i.e., an \( x\% \) decrease in node activeness will generate at least \( x\% \) decrease in the performance gap compared with a ferry in stationary networks.

### 6. MODEL EXTENSIONS

Having studied the problem under the specific framework proposed in Section 3, we now revisit our initial assumptions and extend some of them for broader applicability.

#### 6.1 Multiple Nodes per Domain

The previous solution assumes that each domain contains only one (gateway) node, while in general there may be multiple nodes per domain. First, we note that if these nodes stay close and follow group mobility (assuming the group falls into the same cell with high probability), then the ferry’s control will remain the same as before. If, however, the nodes move independently, then the solution will need some adjustments as follows. Specifically, consider the case of \( L \) nodes per domain, following i.i.d. mobility characterized by a transition matrix \( \mathbf{P} \) as defined in Section 3.2.

We first revisit the policy. Assume that to contact a domain, the ferry can contact any node in that domain. An immediate observation is that since all nodes start from the same steady state and undergo the same actions and observations, their beliefs at the ferry are identical, and thus the ferry can keep only one belief vector as before. The probability of contact will, however, increase from \( b'(u) \) to \( 1 - (1 - b'(u))^L \), and thus the value iteration in (5) becomes

\[
V_T(b) = \gamma \max_{u \in U} \left[ 1 - (1 - b'(u))^L \right] \left[ (1 - (1 - b'(u))^L) \left[ \sum_{i=1}^{L-1} (1 - b'(u))^{L-i} b_{i-1}(u) \right] \right]
\]

Accordingly, the optimal policy will also change with \( L \). An exception is the myopic policy, where since maximizing \( 1 - (1 - b'(u))^L \) is the same as maximizing \( b'(u) \), the policy is independent of \( L \).

We then revisit the analysis. Since the belief updates remain the same, the upper and lower bounds in Theorem 5.2 are still valid, with the minor change that the conditional contact probability \( p_i \) in (12) will be replaced by \( 1 - (1 - \max_i b_i'(s))^L \) and similarly for \( \bar{p}_i \). The real question, however, is what is the performance impact of having more nodes per domain. This requires an extension of the previous results to a general \( L \) (proof in [4]).

**Corollary 5.6.** Under \( L \geq 1 \) nodes per domain, Corollary 5.4 and 5.3 can be extended as follows: for biased mobility, the switching policy is still optimal if \( b_i'(s) > b_i'/(1 - \hat{b}_0) \), and their inter-contact time will converge to 1 as \( L \) increases at \( O((1 - b_0)/(1 - \hat{b}_0)) \); for symmetric mobility and large domains (i.e., \( n \gg 1 \)), \( \mathbb{E}[K^{SW}] \approx n/L \), and

\[
\frac{n}{L + 1} \leq \mathbb{E}[K^{DY}] \leq \frac{n[1 - (1 - d(n)/n)^{L+1}]}{(L + 1)(1 - (1 - d(n)/n)^{L+1})},
\]

where \( d(n) \) is defined as in Corollary 5.3.

For more explicit results, we note that the upper bound in (18) will converge to \( n/(L + 1) \) if the domain is constrained \( \lim_{n \to \infty} d(n)/n > 0 \), or to \( n/L \) if it is unconstrained \( \lim_{n \to \infty} d(n)/n = 0 \). In the latter case, however, simulations show that the bound is loose and \( \mathbb{E}[K^{DY}] \approx n/(L + 1) \) anyway (see Section 7). From these results, we see that increasing node density will have a significant impact on ferry performance, causing the inter-contact time to decrease inverse proportionally (as \( n/L \) for predetermined control and \( n/(L + 1) \) for dynamic control) and eventually converge to one exponentially. This result leads to the following observation: having \( L \) (gateway) nodes per domain in conjunction with dynamic ferry control is almost equivalent to having \( L + 1 \) nodes per domain with predetermined ferry control, where the approximation is tight for \( L \geq 2 \) (see Fig. 9). Therefore, dynamic control can provide significant improvement over predetermined control at low node density (i.e., small \( L \)), but only diminishing improvement at higher node densities. Intuitively, this is because when the node density is high, the ferry will probably meet a node no matter where it goes, and its mobility strategy becomes less important. Nevertheless, the savings in gateway nodes by using dynamic control will add up when the network becomes highly partitioned (i.e., the number of domains is large).

The above extension assumes fixed and known number of (gateway) nodes per domain. If nodes can move in and out of domain boundaries during the operation, then the number of nodes \( L \) will vary over time. Accordingly, the controller will need to track it by another belief \( b_L(l) := \Pr(L = l) \), and the value iteration in (17) will be modified by replacing the right-hand side by its expectation under \( b_L \), with detailed analysis left to future work.
6.2 Joint Global and Local Control

We have so far only focused on the local control in the 2-tiered structure proposed in Section 3. Ideally, the global and the local decisions should be made jointly, although doing so will only be suitable for small N as explained below.

To achieve joint control, we first need to expand the action space. With a little abuse of notation, denote it by \( \mathcal{U} = \bigcup_{i=1}^{N} \mathcal{U}_{i} \), where \( \mathcal{U}_{i} = \{1, \ldots, n_{i}\} \) is the set of cells in domain \( i \). Accordingly, the belief vector will also be expanded to a set of belief vectors \( \mathbf{B} := (\mathbf{b}_{i})_{i=1}^{N} \), where \( \mathbf{b}_{i} \) is the belief of node \( i \) (still assuming one node per domain). Suppose that nodes in different domains move independently. To navigate globally, the controller needs to keep track of all the beliefs. Specifically, if the ferry visits cell \( u \in \mathcal{U}_{i} \) with observation \( z_{t} \) in slot \( t \), then at the end of the slot, the beliefs will be updated as \( \mathbf{B}_{t+1} = f(\mathbf{B}_{t}, u_{t}, z_{t}) \), where

\[
\mathbf{b}_{i, t+1} = \begin{cases} \mathbf{P}_{i}^{T} \mathbf{e}_{u_{t}} & \text{if } z_{t} = 1, \\ \mathbf{P}_{i}^{T} (\mathbf{b}_{i, t})_{\setminus u_{t}} & \text{if } z_{t} = 0, 
\end{cases}
\]

for the visited domain \( i \), and \( \mathbf{b}_{j, t+1} = \mathbf{P}_{j}^{T} \mathbf{b}_{j, t} \) for any other domain \( j \). The value iteration in (5) now becomes

\[
V_{P}(\mathbf{B}) = \gamma \max_{u \in \mathcal{U}} \left[ r_{i} b_{i}(u) + b_{i}(u) V_{T-1}(f(\mathbf{B}, u, 1)) + (1 - b_{i}(u)) V_{T-1}(f(\mathbf{B}, u, 0)) \right],
\]

where \( r_{i} \) denotes the reward of contacting node \( i \), \( e.g., \) setting \( r_{i} \) of the most recently contacted node to zero can avoid getting stuck within the same domain.

The joint control comes at the cost of increased complexity. During online operation, it will require both more memory (\( O(\sum_{i=1}^{N} n_{i}) \)) to store the additional belief vectors and more computation per slot (\( O(\sum_{i=1}^{N} n_{i}^{2}) \)) to update them. During offline policy computation, the complexity scaling under peak speed diminishes as \( \sum_{i,j} P_{j} \) increases because the belief will any-way converge back to \( \mathbf{b}_{0} \) while the ferry is away. Therefore, the joint control is only suitable for small \( N \).

7. SIMULATIONS

We now test the proposed policies via simulations. We use a generalized random walk to model node mobility, where under peak speed \( m \geq 2 \) as defined in Section 5.2, a node moves from its current cell \( i \) to one of its \((m-1)\)-hop neighbors, cell \( j \), with transition probability \( P_{j,i} \) proportional to \( e^{-r/|j-i|} \) for a tightness parameter \( r \) and a home cell \( h \) (\(|j-i|\) is the taxicab distance between cells \( j \) and \( h \)). The home cell \( h \) is used to simulate asymmetric (localized) mobility. It models a spot that the node drifts toward (for \( r > 0 \)) or away from (for \( r < 0 \)) with strength \( |r| \). We set \( h \) to the center of the domain, and tune \( r \) to control node locality. Moreover, we control node activeness by setting \( \sum_{j \neq h} P_{j,i} = \beta \in (0, 1) \). We simulate the ferry’s mobility by storing only one value vector that dominates the optimal set of value vectors in every element.

The joint control policy, and assume that a contact occurs whenever the ferry is in the same cell as a node.

**Policies:** We first compare various dynamic control policies as well as the predetermined switching policy; see Fig. 3-5. Besides the myopic policy (“MY”), the switching policy (“SW”), and the optimal policy (“VI”; see Section 4.2.1), we also implement several state-of-the-art heuristic policies in POMDP: Point-Based Value Iteration [9] (“PBVI”), Interpolation -Extrapolation [3] (“IE”), and Fast-Informed Bound [3] (“FIB”). We set the parameters of PBVI and IE (the number of sampled belief vectors) to 10n, although we find their performance to be insensitive to this parameter. For each policy, we simulate \( 10^{4} \) Monte Carlo runs, where each run lasts for 9 slots starting from the steady state. Policy VI represents the best possible performance (with partial observations), and SW represents a benchmark as it is the best predetermined policy (see Claim 5.1); note that other existing ferry control solutions are not applicable in our settings. From the results, we see that although the myopic policy is suboptimal, it closely approximates the optimal and the other sophisticated policies at a much lower complexity. To understand this, we plot both the discounted and the actual performance under different \( \gamma \) (omitted due to space limitation), and observe that the actual performance is insensitive to \( \gamma \), which explains the good performance of the myopic policy as \( \gamma \) controls the level of greediness. Moreover, we see from Fig. 5 that the policies with better average performance also have smaller variance. Note that the dent around \( T = 5 \) for the dynamic policies is not an artifact, but an indicator that the ferry has met the first node and switched to another domain (thus everything restarts). The switching policy does not have this since it switches domains constantly. Similar observations hold in 2-D cases (omitted).

**Mobility parameters:** We then verify the impact of various node mobility parameters on the inter-contact time. To guarantee convergence, we fix the total number of contacts (\( 10^{4} \) in the simulations) instead of the total simulation time. First, we simulate the switching and the myopic policies under both 1-D and 2-D mobilities with different \( n \) to test the impact of domain size and layout, and compare the results with the bounds in Corollary 5.3; see Fig. 6. The switching policy and the lower bound are invariant to the domain layout as long as \( \mathbf{b}_{0} \) is fixed; however, the myopic policy is sensitive. As predicted by analysis, it performs better in constrained domains (1-D), improving the switching policy by an amount that grows linearly with \( n \). As the domain becomes unconstrained (2-D), the performance degrades, but is clearly better than the upper bound, implying that the upper bound is loose in 2-D, and dynamic control can still provide significant performance gain. As the (peak) node speed \( m \) increases (not shown), the myopic policy becomes less sensitive to domain layout, with inter-contact times for 1-D and 2-D both converging to the upper bound with \( d(n) \approx n/m \), i.e., \( E[K^{MY}] \approx (2m - 1)n/(2m) \).

Next, we test the impact of locality and activeness by varying the tightness parameter \( r \) and the activeness parameter \( \beta \); see Fig. 7-8. Fig. 7 shows that the ferry’s performance is highly sensitive to variations in node locality. Dynamic control (“MY”) outperforms the predetermined control (“SW”) when the node steady-state distribution is close
to uniform (|τ| ≪ 1), but if |τ| exceeds a threshold, the distribution becomes so localized that the two policies become identical. Interestingly, instead of smooth transitions, we see sharp jumps at both ends of the optimality region of the switching policy. This region contains the optimality region derived from Corollary 5.4 (outside the vertical lines), verifying it as a sufficient condition. Fig. 8 confirms that the switching policy and the lower bound are independent of the level of activeness (note: as β → 0, the nodes become static, but their initial locations are still random), while the myopic policy quickly converges to the lower bound with the decrease of β as predicted in Theorem 5.6.

Extensions: Finally, we test the extensions in Section 6. In Fig. 9, we test the impact of having more gateway nodes per domain, assuming i.i.d. mobility of the nodes. As predicted by Corollary 6.1, both inter-contact times are inversely proportional to node density. Their gap is the largest at L = 1 (40%) and shrinks as L increases since the ferry will anyway meet a node within the first few slots. Again, the upper bound for dynamic control is loose in 2-D cases.

So far we have only tested the local control (by assuming N = ∞). To test the effect of global navigation for finite N, we simulate the myopic policy for joint control as in (19), where r1 = 0 if the latest contact is in domain i and r1 = 1 otherwise, together with a 2-tiered control using sequential order at the global tier and the myopic policy at the local tier, the switching policy, and the random control (random cell and domain selection). We simulate 1000 Monte Carlo runs (starting from the steady state) of 40 slots each; see Fig. 10–11. Fig. 10 confirms that joint control can significantly outperform the 2-tiered control in absolute performance by adaptively optimizing the destination domain. The switching policy yields poor performance for small N, even worse than random control, as it blindly jumps between cells that are more probable in the steady state (for finite N, the actual belief will deviate from the steady state over time). However, the distribution of contacts among different domains (Fig. 11) shows that the joint controller only serves a pair of domains, while the others are more balanced. This is because in order to maximize the value of past information, the joint controller always returns to the previously visited domain and never moves to new domains. This observation reveals a tradeoff between performance and fairness; further investigation is left to future work.

8. CONCLUDING DISCUSSIONS

We have studied mobility control of autonomous data ferries with partial observations. In contrast to existing solutions, the proposed solution is fully dynamic. Our analysis and evaluations show promising performance improvement measured by increased contact rate, especially in cases when the performance suffers most. Next, we discuss several possible extensions while leaving the details to future work. Specifically, although our solution is limited by our assumptions (single ferry, known Markovian mobility, etc.), our approach is applicable to a much broader range of problems as discussed below:

Alternative performance criteria: Although initially focused on contacts, the proposed framework can be extended to model other performance criteria such as throughput and delay. One way is to use the rewards r; in (19) to model these criteria and allow the ferry to update them over time. For example, we can model throughput by setting r; to be the number of messages the ferry has buffered for node i, which will also minimize the average delay per message.

Multi-ferry control: In cases that require multiple ferries, a straightforward solution is to divide the domains into groups and assign only one ferry per group such that ferry loads are balanced in the long run. The drawback of this solution is that some ferries can be temporarily under-utilized, whereas the number of messages the ferry has buffered for node i, which will also minimize the average delay per message.

Constraints/costs of ferry movements: While we have given the ferry the full freedom to move anywhere, the control framework allows one to impose practical constraints such as speed limit, obstacles, etc. One way is to restrict the feasible action space, where at time t, the action space U_t should only include cells reachable from the ferry’s current location u_{t-1} in one slot. A related issue is the costs of navigation, which can be modeled by incorporating negative rewards into the payoff function (e.g., r(z, u) = z - c(u)).

Channel effects: We have ignored channel effects by assuming that a ferry will always detect a node within the same cell. In practice, noise and imperfect channel conditions may cause the ferry to miss a node even if it is within range, and this effect can be modeled by incorporating an overlook probability into the observation model (i.e., P_z(s, u, z) = p_o > 0 for s = u, z = 0).

Non-Markovian mobility and unknown parameters: We have assumed the node mobility models to be Markovian and known. While one can always use a Markovian model as an approximation, the resulting control performance will be application-dependent and require case-by-case investigation. A related question is how to estimate model pa-
parameters in practice. One way is to rely on training, e.g., by recording node trajectories over a training period. Alternatively, one may let the ferry learn node mobility while ferrying data using reinforcement learning techniques.

From these discussions, we see that our approach can be extended to address a rich set of problems related to autonomous data ferry control, while our current solution can serve as a concrete building block in the solution space.

9. REFERENCES