Controlling the Mobility of Multiple Data Transport Ferries in a Delay-Tolerant Network

Wenrui Zhao, Mostafa Ammar and Ellen Zegura
College of Computing, Georgia Institute of Technology, Atlanta, Georgia 30332
{wrzhao, ammar, ewz}@cc.gatech.edu

***Abstract*— As technology rapidly progresses, more devices will combine both communication and mobility capabilities. With mobility in devices, we envision a new class of *proactive* networks that are able to adapt themselves, via physical movement, to meet the needs of applications. To fully realize these opportunities, effective control of device mobility and the interaction between devices is needed. In this paper, we consider the Message Ferrying (MF) scheme which exploits controlled mobility to transport data in delay-tolerant networks, where end-to-end paths may not exist between nodes. In the MF scheme, a set of special mobile nodes called *message ferries* are responsible for carrying data for nodes in the network. We study the use of *multiple ferries* in such networks, which may be necessary to address performance and robustness concerns. We focus on the design of ferry routes. With the possibilities of interaction between ferries, the route design problem is challenging. We present algorithms to calculate routes such that the traffic demand is met and the data delivery delay is minimized. We evaluate these algorithms under a variety of network conditions via simulations. Our goal is to guide the design of MF systems and understand the tradeoff between the incurred cost of multiple ferries and the improved performance. We show that the performance scales well with the number of ferries in terms of throughput, delay and resource requirements in both ferries and nodes.**

***Index Terms*— System design, Simulations**

I. Introduction

As technology rapidly progresses, more devices will combine both communication and mobility capabilities. For example, mobile robots have been developed for a wide range of applications, such as military, disaster recovery, home and factories [3], [4], [5], [19]. Researchers are also developing robotic prototypes, such as robotic insects that can fly or walk on water [1], [2]. It is anticipated that devices with mobility and communication capabilities will become more popular in the future. In addition, device mobility can be achieved via the movement of other entities. For example, movement of people, ground or aerial vehicles may provide the required mobility for the carried devices. With mobility capability in communication devices, we envision a new class of *proactive* networks that are able to adapt themselves, via physical movement, to meet the needs of applications. This is in contrast to traditional networks where applications are required to adapt to network mobility conditions [23]. The physical adaptability of these networks has the potential to provide new or better services to users. To fully realize these benefits, however, effective control of the network, or the mobility of devices and the interaction between devices, is needed.

While controlled mobility is an active research area in the robotics community [10], there has been little effort in exploiting this capability for communication purposes. Li and Rus [20] consider proactive movement of nodes to deliver messages in a disconnected environment and present an algorithm to compute optimal node trajectories. In [12], Goldenberg et al. study mobility as a control primitive in mobile networks and present a distributed mobility control scheme to adjust node positions such that energy consumption in communication is minimized. In the NIMS project [18], Kaiser et al. propose the use of infrastructure-supported mobility in sensor networks for autonomous operations and physical reconfiguration. They describe the use of mobility to transport data and optimize wireless links.

In prior work [29], [30], we have proposed the Message Ferrying (MF) scheme for delay-tolerant networks where end-to-end paths do not exist between some or all nodes [11], [15]. To overcome network partitions, the MF scheme exploits controlled mobility to transport data. Specifically, a set of special mobile nodes called *message ferries* move around the deployment area and are responsible for carrying data between nodes. The MF scheme can be used in different delay-tolerant environments. For example, in a disaster scene where existing infrastructure is unusable, airplanes or vehicles can be used as ferries to transport data between users in separated areas. In sensor networks where power supplies are severely limited, mobile entities such as robots or manned vehicles can be deployed to approach and collect data from sensors in order to conserve sensor energy.

In MF networks, data is transported via ferry mobility. Therefore, the design of ferry movement, or *ferry routes*, will have significant impact on network performance. In our earlier work [29] we consider ferry route design in networks with a single ferry and develop algorithms to compute a ferry's route in order to achieve certain performance objectives[1].

In this paper, we study the use of *multiple ferries* in networks with stationary nodes, focusing on the design of ferry routes. The use of multiple ferries may be necessary in many situations due to performance and robustness concerns. First, the capacity of a single ferry to carry and forward traffic

This work was supported by the NSF Grant ITR-0313062 and AFOSR MURI Grant F49620-00-1-0327.

[1]In [30], we study the MF scheme in mobile networks but do not consider the ferry route design problem.

is limited by its movement capability. To allow for scalability in traffic load or geographic coverage, multiple ferries are required. Second, a single ferry system is vulnerable to ferry failures, ferry compromise or malicious attacks. Multiple ferries would provide the required level of fault tolerance via redundancy. On the other hand, with the possibilities of interaction between ferries, the use of multiple ferries brings significant challenges to the route design problem as compared to the single ferry case. Since data from nodes can be carried by different ferries, there is a question of how ferries are allocated to serve nodes. In addition, ferries may interact with each other, either directly or indirectly via nodes, to provide better communication services. In this case, the design of ferry routes should account for routing and load balancing among ferries, which inevitably complicates the problem. There is also the question of tradeoff between the increased cost of the use of more ferries and the extent of performance improvement realized.

We consider different strategies in the route design that reflect different assumptions about the network, e.g., whether the multiple ferries go over the same or different routes, and how ferries interact with each other. We develop algorithms to generate ferry routes that meet the traffic demand and minimize the average data delivery delay. Using simulations, we evaluate these algorithms under a variety of network conditions and study the data delivery performance of the MF schemes. Our goal is to guide the design of MF systems and understand the tradeoff between the incurred cost of multiple ferries and the improved performance. We show that the MF scheme scales well with the number of ferries in terms of throughput, delay and resource requirements in both ferries and nodes.

The rest of this paper is structured as follows. Section II gives an overview of the MF scheme and describes the network model and the ferry route design problem. In Section III, we provide an overview of four algorithms to compute ferry routes, which will be described in details from Section IV to Section VII. We present simulation results in Section VIII to evaluate these algorithms and the MF performance. Related work is reviewed in Section IX and the paper is concluded in Section X.

II. Overview of Message Ferrying Networks

In this section we first give an overview of the Message Ferrying scheme. Then we describe the network model and the ferry route design problem.

A. Message Ferrying Scheme

The Message Ferrying (MF) scheme exploits controlled mobility to provide physical connectivity between otherwise disconnected nodes. In an MF scheme, network devices are classified as *message ferries* (or *ferries* for short) or *regular nodes* based on their roles in communication. Ferries are devices which take responsibility of carrying data among other nodes, while regular nodes are devices without such responsibility[2]. Fig. 1 illustrates a simplified example of message ferrying, in which two ferries move around the area and relay data between otherwise disconnected nodes.

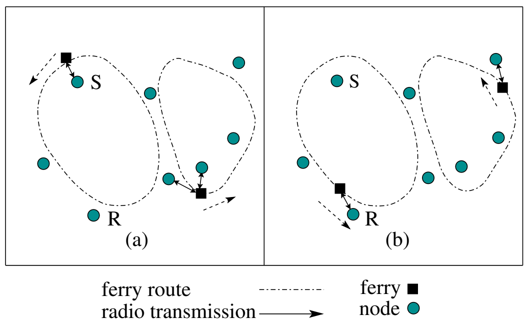


Fig. 1. An example of data delivery in an MF scheme.

In this paper, we consider networks that use multiple ferries to transport data between nodes. With multiple ferries, there are many possibilities of interaction between ferries. For example, data might be forwarded through multiple ferries before reaching the destination. In general, there are three types of ferry interaction.

- *No interaction*. Each ferry operates on its own without relaying data with other ferries.
- *Ferry relaying*. Ferries exchange data between each other directly. This requires ferries to be physically close to each other in order to communicate.
- *Node relaying*. Ferries interact with each other via stationary nodes, i.e., nodes buffer and relay data between ferries. So nodes need to have enough storage and energy for buffering and relaying data.

Note that with node relaying, there is no need for ferries to meet each other because of the use of stationary nodes as relays. With ferry relaying, however, ferries need to synchronize their movement to meet each other for data exchange. As discussed later, synchronization has significant impact on the design of ferry routes.

B. Network Model

We now describe the network model considered in this paper. We consider networks with n stationary regular nodes, each equipped with a wireless radio capable of transmitting to other nodes within a distance r. The data rate of the radio is W bits per second[3]. We assume that the network is sparse such that nodes are disconnected from each other (i.e., the distance between each pair of nodes is larger than the radio range r)[4].

We assume that bandwidth requirement between each pair of nodes is known and constant over time. In practice, while

[2]We will use the term “node(s)” to refer to regular nodes or both regular nodes and ferries. The usage should be clear from the context.

[3]We can set W to account for the unreliability of the wireless medium, e.g., the loss rate of the links.

[4]Our model can be applied to connected or partially connected networks. In these networks, ferries can carry data among a set of gateway nodes, which in turn communicate with other non-gateway nodes nearby using traditional ad hoc routing algorithms.

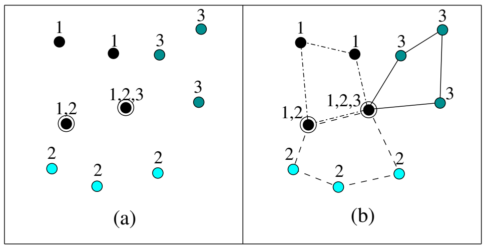


Fig. 2. Overview of ferry route design algorithms.

traffic might not be known in advance, traffic demand can often be estimated. This is true especially for the MF scheme, which is expected to operate at large time scales, e.g., in minutes or hours. In addition, since ferries visit nodes occasionally, the impact of traffic burstiness is of less concern. To provide communication services to regular nodes, m ferries are deployed, $m \geq 1$ and $n \geq m$. We assume that each ferry moves at a constant speed and ferries are equipped with the same radios as nodes, i.e., the radio range is r and the data rate is Wbps. Ferries and nodes communicate via a shared wireless channel. Therefore transmission and reception cannot occur at the same time. For a network with m ferries, the total data rate that these ferries can support is no more than $0.5mW$bps. This is because the wireless channel is shared among transmissions from data sources to the ferries and from the ferries to data destinations. Given m ferries that could transmit or receive at Wbps simultaneously, we have the maximum total data rate of $0.5mW$bps.

C. Ferry Route Design Problem

In this paper, we consider the design of ferry routes. We choose to minimize the average data delivery delay. While the delay of MF is significantly larger as compared to connected networks, achieving lower delay is still desirable for many applications. Specifically, we try to minimize the *weighted delay* D for all traffic which is defined as

$$D = \frac{\sum_{1 \leq i,j \leq n} w_{ij} d_{ij}}{\sum_{1 \leq i,j \leq n} w_{ij}} \tag{1}$$

where w_{ij} and d_{ij} are the weight and average delay for data from node i to node j. The weight w_{ij} specifies the relative importance of reducing delay for certain traffic and may be independent of the data rate.

Now we define the ferry route design problem, which consists of finding optimal ferry routes such that the bandwidth requirements are met and the weighted delay is minimized. In this paper, we consider periodic ferry routes, i.e., ferries visit the same set of nodes periodically. In previous work [29], we show that designing an optimal route for a single ferry is NP-hard. Thus as a generalized problem, the ferry route design problem with multiple ferries described above is also NP-hard. In the following sections, we will present several heuristic algorithms to compute ferry routes.

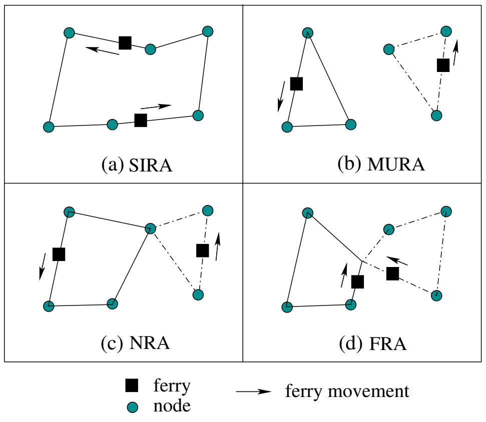


Fig. 3. Example of ferry routes generated by different algorithms.

III. Overview of Ferry Route Design Algorithms

In this section we give an overview of the four ferry route design algorithms that we develop and present in the following sections. In general, these algorithms calculate ferry routes in three phases. In Phase 1, nodes are assigned to ferries. Ferries are responsible for carrying data for assigned nodes and cooperatively provide connectivity between each pair of sender and receiver. This assignment can be flexible, e.g., a node may be assigned to multiple ferries. Fig. 2(a) shows an example of node assignment where the numbers near each node denote the ferries the node is assigned to. We can see that some nodes are assigned to more than one ferry. Phase 2 calculates each ferry route based on the locations of assigned nodes and the traffic load. Fig. 2(b) depicts an example of three ferry routes. In both Phase 1 and 2, the algorithms focus on minimizing the weighted delay without considering the bandwidth requirements. In Phase 3, the ferry routes are extended, if necessary, such that the bandwidth requirements are met. So, rather than directly addressing the ferry route design problem, these algorithms break it into three sub-problems and solve each sub-problem.

As discussed in Section II, there are three types of interaction between ferries. In the following sections, we will present algorithms to calculate ferry routes based on different assumptions of ferry interaction. In Section IV, we describe the Single-Route Algorithm (SIRA) in which all ferries follow the same route. Section V presents the Multi-Route Algorithm (MURA) in which ferries can follow different routes. In both SIRA and MURA, there is no interaction between ferries. In Section VI and VII, we present the Node Relaying Algorithm (NRA) and the Ferry Relaying Algorithm (FRA), which utilize node relaying and ferry relaying, respectively, to forward data between ferries. Fig. 3 shows simplified examples of ferry routes generated by these algorithms. These algorithms differ mainly in how nodes are assigned to ferries because of the different assumptions about ferry interaction. Besides, FRA requires special treatment in calculating routes due to the synchronization issue.

These four algorithms represent different strategies in designing ferry routes. In an MF network with multiple ferries, data may be relayed via multiple ferries before being delivered to the destination. Thus for data between a sender/receiver pair, the average delay consists of delay in each ferry hop, which includes both the waiting time in node buffers before data is forwarded to a ferry and the carrying time in the ferry before data is relayed to another ferry or the destination. So, without relaying data between ferries, SIRA and MURA minimize the number of ferry hops. FRA tries to minimize the waiting delay in nodes through direct interaction between ferries. NRA tries to minimize the carrying delay in each ferry hop by using stationary nodes as relays.

IV. Single-Route Algorithm (SIRA)

In this section, we describe the Single-Route Algorithm (SIRA) in which all ferries follow the same route but with different timing (see Fig. 3(a)). That is, each node is assigned to all ferries which share the responsibility of transporting data between nodes. In the following we focus on Phase 2 and 3, i.e., how to design a single ferry route for m ferries. In prior work [29], we have developed algorithms to compute a ferry route for a single ferry. We now extend these algorithms for the case with multiple ferries. We assume that ferries have the same speed and move on the ferry route with equal distance in between.

A. Single Ferry Route Design

We first describe how to calculate a single ferry route. We try to minimize the weighted delay without considering the bandwidth requirements, which will be addressed in the next section. To compute the weighted delay, we need to determine the average delay for data between each sender/receiver pair. In an MF network, the average delay for data from node i to node j consists of the waiting delay in node i before a ferry picks up the data and the carrying delay in the ferry before reaching node j. Given constant data rates and m ferries, the average waiting delay is $L/(2mf)$ where L is the length of the ferry route and f is the ferry speed. This is because the round trip time for a ferry is L/f. With m ferries, the time between ferry visits is L/mf and on average, the waiting delay is half of that, i.e., $L/(2mf)$. Suppose that the distance from node i to node j in the route is l_{ij}. The carrying delay is l_{ij}/f. So the average delay for data from node i to node j is $L/(2mf) + l_{ij}/f$. To compute the ferry route, we adapt solutions for the well-studied traveling salesman problem (TSP) [7], [16] which compute a route to visit nodes. But instead of optimizing the length of the route as in TSP, we optimize the weighted delay. Fig. 4 illustrates a sketch of the algorithm, which generates an initial route using some TSP heuristic algorithm, e.g., the nearest neighbor heuristic, and then refines the initial route using local optimization techniques. We consider the following 2-opt and 2H-opt swap operations to improve the route.

- *2-opt swap.* Consider the route as a cycle with edges that connect consecutive nodes in the route. A 2-opt swap removes two edges AB and CD from the route and replaces them with edges AC and BD while maintaining the route as a single cycle.
- *2H-opt swap.* A 2H-opt swap moves a node in the route from one position to another.

```
Compute an initial route using TSP heuristic algorithm;
do
    Apply 2-opt swaps;
    Apply 2H-opt swaps;
while (weighted delay is reduced);
Extend ferry route to meet bandwidth requirements;
```

Fig. 4. Single-route algorithm.

The algorithm tries to reduce the weighted delay of the route by applying 2-opt swaps and 2H-opt swaps until no further improvement can be found.

B. Bandwidth Requirements

In the previous section, we consider only the weighted delay. Now we consider how to extend the ferry route, if necessary, to meet the bandwidth requirements of the nodes. For any given route, the achieved data rate of a node is λW where λ is the fraction of time the node communicates with ferries. So we need to extend the amount of time ferries spend in the vicinity of those nodes that do not otherwise have enough time to transmit or receive their data. In practice, the extension could consist of changing ferry speed, so that the ferry spends more time on certain parts of the route. However, since we have assumed constant ferry speed, the extensions consist of detours in the vicinity of the under-served node(s). Obviously the detours should be as short as necessary, in order to reduce the delay.

We formulate this problem using linear programming as follows. Let x_i be the length of detour in the vicinity of node i. We assume ferries move to the location of each node, thus the total length of the ferry route that is within the radio range of node i is $x_i + 2r$. Let s_i be the total data rate for node i which is the sum of data rates in both transmission and reception. By distributing traffic load equally to ferries, each ferry is responsible for supporting a data rate of s_i/m. Thus we have

$$\frac{(x_i + 2r)W}{L + \sum_{j=1}^{n} x_j} \geq \frac{s_i}{m}$$

where L is the length of the ferry route before extension. After transformation, we get the following optimization problem.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} x_i, \\ \text{subject to} \quad & mWx_i - s_i \sum_{j=1}^{n} x_j \geq s_i L - 2mrW, \\ & x_i \geq 0 \text{ and } 1 \leq i \leq n. \end{aligned} \tag{2}$$

The above problem can be solved efficiently using methods like Simplex [21].

```
EWD(op): EWD of node assignment after operation op

Set the number of ferries to $n$;
Assign each node to a ferry;
**while** number of ferries $> m$ **or** EWD is reduced **do**
    Identify the best $overlap$ or $merge$ operation $op_s$;
    Identify the best $merge^-$ or $reduce$ operation $op_l$;
    **if** EWD($op_s$) $<$ EWD($op_l$) **and**
       EWD($op_s$) $<$ current EWD **then**
          Perform $op_s$;
    **else**
          Perform $op_l$;

Refine node assignment to maintain feasibility;
Compute each ferry route;
```

Fig. 5. Multi-route algorithm. EWD refers to estimated weighted delay.

V. Multi-Route Algorithm (MURA)

In the MURA algorithm, ferries may follow multiple routes to carry data between nodes (see Fig. 3(b) for an example). But ferries do not relay data between themselves. So data is carried by at most one ferry.

Fig. 5 shows a sketch of the MURA algorithm which uses a greedy heuristic for assigning nodes to ferries. MURA starts with n ferries and each node is assigned to a ferry. That is, each ferry route consists of one node. MURA refines the node assignment and reduces the number of ferries to m by using four types of operations, which will be described in Section V-B. In each step, MURA estimates the weighted delay of the resulting node assignment for each operation and chooses to perform the best one until the number of ferries is m and no further improvement can be found. We will describe the estimation of weighted delay in Section V-A. Then MURA modifies the node assignment, if necessary, to insure feasibility, i.e., there is a path between each sender/receiver pair and the total traffic load on each route is lower than its capacity. Given the node assignment, we can apply the algorithms in SIRA to compute each ferry route.

In the following, we will describe how to estimate the weighted delay for a node assignment. Then we explain how to assign nodes to ferries and maintain feasibility.

A. Estimated Weighted Delay

We first describe how to estimate the weighted delay, or calculate the *estimated weighted delay* (or *EWD* for short). For a node assignment, the sender and receiver may be assigned to the same route or different routes. The EWD needs to account for the weighted delay contributed by both types of traffic.

To compute the EWD for traffic within a route, we consider the following factors. First, the EWD should reflect the weights of traffic within the route, as implied by the definition of the weighted delay. Second, the data delay consists of the waiting time at nodes and the carrying time at ferries, both related to the length of the route. Thus the EWD should account for the length of the route. Third, in an MF network, achieving higher data rate implies that ferries need to spend more time communicating with nodes, resulting in longer routes and larger delays. So the EWD should consider the traffic load in the route. Finally, the EWD should consider the number of ferries used in a route because it affects both the waiting delay at nodes and the amount of traffic carried by each ferry.

Combining these factors together, we now define the EWD. Let L be the length of a TSP route for nodes in the route. Let α and ω be the total data rate and weight of traffic within the route. Suppose that the number of ferries following the route is k. The maximum data rate μ of the route, or the capacity of the route, is $0.5kW$bps. We define the EWD of the route as a two-component tuple (E^*, E') where

$$E^* = \begin{cases} \omega L(1+\alpha-\mu) & \text{if } \alpha \geq \mu; \\ 0 & \text{if } \alpha < \mu; \end{cases} \tag{3}$$

$$E' = \begin{cases} 0 & \text{if } \alpha \geq \mu; \\ \omega L(1+\frac{1}{k})(1+\frac{\alpha}{\mu-\alpha}) & \text{if } \alpha < \mu. \end{cases} \tag{4}$$

When the traffic load is over the capacity, E^* is positive and measures how much the ferry route is overloaded. To differentiate E^* between the case where α equals to μ and the case with $\alpha < \mu$, we add a constant term (1 in this paper) in computing E^* when $\alpha \geq \mu$. When the traffic load is below the capacity, E^* is zero and E' approximates the weighted delay for traffic in the route. Obviously E^* is the more significant component when comparing two EWDs, i.e., $(E_1^*, E_1') > (E_2^*, E_2')$ if either $E_1^* > E_2^*$ or $E_1^* = E_2^*$ and $E_1' > E_2'$ is true.

We now explain the definition of EWD in (4) in more details. The use of total traffic weight ω and the route length L is obvious. The factor $(1+\frac{1}{k})$ is to approximate the average delay for traffic within the route. As discussed in Section IV, the average delay for data from node i to node j is $L/(2kf) + l_{ij}/f$, where l_{ij} is the distance from node i to node j in the route. If we set l_{ij} to $L/2$, the average delay becomes $\frac{L}{2f}(1+\frac{1}{k})$. So we use the factor $(1+\frac{1}{k})$ in (4). To account for the impact of traffic load, EWD incorporates the factor $(1+\frac{\alpha}{\mu-\alpha})$. This is because to meet the bandwidth requirements, ferries need to spend enough time within range of nodes, which implies detours in ferry routes, as explained in Section IV-B. Suppose that x is the total length of detours. To support a total data rate of α, we have

$$\frac{x\mu}{x+L} = \alpha \quad \Rightarrow \quad x = \frac{\alpha L}{\mu - \alpha}.$$

Here we make an approximation that the length of the route that is within range of nodes is x. So the total length of the route after extension is $L(1+\frac{\alpha}{\mu-\alpha})$. By combining these factors, we obtain the definition of EWD in (4).

We further define the EWD for traffic between two routes, say route i and j. To support this traffic, MURA may need to extend route i or j such that both the sender and receiver are on the same route because data is not relayed between ferries. Therefore, we define the EWD for traffic between route i and j as the increase in the EWD of the extended route. Specifically, suppose that route i is extended to overlap with route j by visiting the closest node in route j. The EWD of route i will increase due to the increase in the length of the route and the traffic load. Similarly, the EWD of route j would increase if route j is extended. So we define the EWD for traffic between route i and route j as the minimum increase in the EWD by extending either route i or j. Given the EWDs for traffic within each route and between routes, we can compute the EWD for the node assignment by summing up these EWDs, i.e., adding the two components of each EWD respectively.

To compute the EWD of a node assignment, we need to determine the traffic each route is responsible for. In this paper, we adopt the following routing strategy to balance load among ferry routes. Initially, no traffic is assigned to ferry routes. We assign traffic to a route if both the sender and receiver are on the route. When traffic is within multiple routes, traffic will be assigned to a route such that the increase of the EWD is minimal. Then we consider traffic between routes. Similarly, traffic will be assigned to routes with minimum increase of EWD. Given the assignment of traffic, we can calculate the EWD of the node assignment as described above.

B. Assigning Nodes to Ferries

In this section, we describe how MURA assigns nodes to ferries using the EWD. We consider four types of operations on ferry routes. Let i and j be two routes.

- $overlap(i, j)$. This operation overlaps two routes, i.e., one route is extended to include a node in the other route. The number of ferries in each route does not change. When there are multiple nodes in i or j, we choose the overlapping node such that the resulting node assignment has the minimum EWD.
- $merge(i, j)$. This operation combines i and j into a new route. The number of ferries in the new route is the sum of the number of ferries in i and j.
- $merge^-(i, j)$. This operation is the same as $merge(i, j)$ except that the number of ferries for the new route is one less than that of $merge(i, j)$.
- $reduce(i)$. This operation reduces the number of ferries in route i by one. Thus it only applies to routes with more than one ferry.

Now we describe how MURA chooses the best operation to perform in each step. Specifically, MURA identifies the best *overlap* operation among all pairs of routes by computing the EWD of the resulting node assignment. Similarly, MURA identifies the best $merge$, $merge^-$ and $reduce$ operations. Because the $overlap$ or $merge$ operation does not reduce the total number of ferries used, it will be chosen only when it outperforms the $merge^-$ and $reduce$ operations and improves upon the current node assignment. If so, MURA will perform either the $overlap$ or $merge$ operation, depending on which one achieves the lower EWD. Otherwise, MURA will perform either the $merge^-$ or $reduce$ operation, again depending on the EWD. MURA will continue this process until the number of ferries is m and no further improvement can be found.

So far, nodes are assigned to ferries such that the EWD is minimized. Due to the fact that MURA is a greedy heuristic, there is a possibility that the node assignment is not feasible, i.e., some sender and receiver are not in the same route or the total amount of traffic in a route is over its capacity. To insure feasibility, MURA further refines the node assignment if necessary. Specifically, if the total amount of traffic in a route is over its capacity, MURA performs a $merge$ or $overlap$ operation between this route and another route such that the resulting EWD is minimal. Similarly, if there is traffic between routes, MURA performs a $merge$ or $overlap$ operation between these routes. This procedure continues until the node assignment is feasible.

```
procedure NFRA(r, c)
    Divide area into a grid of $r \times c$ cells;
    Assign each node to the cell it belongs to;
    Compute traffic load on cells using geographic routing;
    Add inter-route connectivity;
    Allocate ferries to cells;
end

/* main algorithm */
minEWD ← $\infty$;
for every $(c_1, c_2)$ that satisfies $c_1 c_2 \le m$ do
    NFRA($c_1$, $c_2$);
    if current EWD < minEWD then
        minEWD ← current EWD;
        (min$C_1$,min$C_2$) ← ($c_1$, $c_2$);
NFRA(min$C_1$, min$C_2$);
Compute each ferry route;
```

Fig. 6. Sketch of NRA and FRA algorithms.

VI. Node Relaying Algorithm (NRA)

In the Node Relaying Algorithm (NRA), data is relayed between ferries via nodes. That is, ferries forward data to a node and other ferries receive data from this node. By exploiting the fact that nodes are stationary, node relaying eliminates the requirement of synchronization between ferries.

Fig. 6 shows a sketch of NRA. NRA adopts a geographic approach for assigning nodes to ferries. Specifically, the deployment area is divided into a grid of $c_1 \times c_2$ cells. Ferries are assigned to cells and would carry data for nodes within the cell and relay data between cells. This actually implicitly assigns nodes to ferries via cells. Since the number of ferries is m, we have $c_1 c_2 \le m$. For every possible combination of c_1 and c_2, NRA computes the estimated weighted delay as described in Section V-A and chooses to perform according to the combination that achieves the minimum EWD. Given the node assignment, we can compute each ferry route using algorithms in Section IV.

In the following, we will describe how NRA adds connectivity between ferry routes and allocates ferries to routes.

A. Connectivity between Ferry Routes

As described above, each ferry route is initially within a cell. When the sender and receiver of traffic are located at different cells, data has to be forwarded through multiple cells before reaching the destination. In the following, we describe the routing algorithm used and how NRA extends ferry routes to maintain the required connectivity of the network. NRA uses geographic routing, i.e., data is forwarded along cells that intersect with the line segment connecting the source and destination. Fig. 7 shows an example of geographic routing where data from node n_1 to n_2 is routed through cells C_1, C_3 and C_5. To support data forwarding between cells, NRA extends ferry routes to add connectivity between ferry routes. Specifically, when there is traffic between two neighboring cells, NRA performs the *overlap* operation defined in Section V on the ferry routes in these cells. The *overlap* operation extends one route to overlap with the other and the overlapping node will relay data between these two routes. NRA continues this process until there is a path between each sender/receiver pair according to the routing algorithm.

The above discussion assumes that each cell contains nodes. In case of irregular node distribution, there might be empty cells that contain no node. We classify two types of empty cells, depending on whether there is traffic forwarded through the cell. If there is no through traffic, we can ignore this cell. When an empty cell has through traffic, however, a ferry route needs to set up to relay data. Instead of using a separate ferry route for each such empty cell, we use a single route for all neighboring empty cells as follows. Suppose that C_0 is an empty cell with through traffic. Let S be a set of empty cells, which initially contains only cell C_0. NRA adds an empty cell to S if the cell has through traffic and it is adjacent to another cell in S. NRA continues adding cells to S until no more cells can be added. The cells in S then form a region which contains no node but needs to relay traffic. To handle through traffic in S, NRA sets up a new route by identifying the set of non-empty cells that forward or receive traffic with cells in S and constructing a ferry route that passes through all these cells, e.g., choosing one node from each of these cells and forming a new route. By overlapping routes in neighboring cells and adding routes for empty cells, NRA insures that there is a path, possibly through multiple ferry routes, between each sender/receiver pair.

B. Assigning Ferries to Routes

The geographic routing algorithm used might distribute traffic load unevenly among ferry routes. For example, the ferry routes in the center of the deployment area often need to carry more traffic because more traffic passes through that area. Ferry routes might also have uneven load due to the irregularity in node distribution. To provide load balancing, NRA allocates ferries to routes according to the traffic load. Specifically, NRA initially assigns one ferry to each route. If there are remaining ferries, NRA computes the expected weighted delay for each ferry route and allocates one of the remaining ferries to the route with the highest EWD. The addition of a ferry decreases the EWD of this route. NRA will repeat this process until all ferries are allocated. Using this approach, NRA accommodates uneven traffic load in ferry routes by allocating more ferries to routes with higher load.

VII. Ferry Relaying Algorithm (FRA)

In the Ferry Relaying Algorithm (FRA), data may be forwarded through multiple ferry routes while being routed

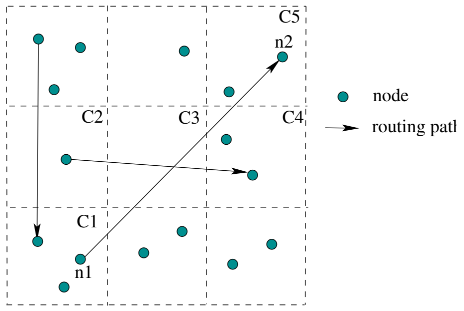


Fig. 7. Example of geographic routing in NRA.

to the destination. But instead of relaying data via nodes as in NRA, ferries exchange data directly. Therefore, ferry routes need to be synchronized, to some extent, for two ferries to meet each other. When the length of two ferry routes are not the same, the contacts between ferries often occur irregularly. So data has to be buffered in a ferry for a significant period of time before it can be forwarded to another ferry, leading to long delays. In this paper, therefore, we consider ferry relaying by synchronizing ferry routes in length.

The operation of FRA is similar to NRA (see Fig. 6 for a sketch of the algorithm). Briefly, FRA divides the area into a grid of $c_1 \times c_2$ cells and uses the geographic routing algorithm to forward data. Based on traffic load, FRA allocates ferries to routes to balance load among routes. As in NRA, FRA chooses the best combination of c_1 and c_2 to minimize the EWD. However, due to the synchronization requirement between ferry routes, FRA differs in how to add connectivity between routes and how to calculate ferry routes. In addition, FRA sets up a ferry route for each cell except empty cells that have no through traffic.

In the following, we will describe the issue of synchronization between ferry routes and explain how to compute ferry routes in FRA.

A. Synchronization between Ferry Routes

In this section, we describe how to achieve synchronization between ferry routes. We assume that all routes have the same length. In the next section, we will describe how routes are extended to have the same length. Suppose that the area is divided into a grid of $c_1 \times c_2$ cells. We first consider the case with a one-dimensional grid, i.e., $c_1 = 1$ or $c_2 = 1$. In this case, a route interacts with at most two other routes in the neighboring cells. We assume that ferries meet each other at the middle points of the boundary between cells, which we call *contact points* (see Fig. 8(a)). To insure ferries in neighboring cells are able to meet each other in every period of their movement, FRA requires that ferries in neighboring cells move in reverse direction, and the contact points partition each ferry route into two segments of the same length (see Fig. 8(a)).

Now we consider the case with a two-dimensional grid, i.e., $c_1 > 1$ and $c_2 > 1$. In this case, a ferry route may interact with up to eight other routes in neighboring cells. Similar to the case with a one-dimensional grid, each route has eight contact points. To insure contacts between ferries in all neighboring cells, ferries move in different directions and timing, as shown in Fig. 8(b). FRA also requires that contact points partition each route into eight segments of the same length.

The above discussion assumes that each route has a single ferry. When a route has multiple ferries, the contact points will be visited more frequently. In this case, the above requirements hold except that the route length should be proportional to the number of ferries in the route, i.e., L/k is the same for all routes where L is the length of a route and k is the number of ferries in the route.

B. Calculating Ferry Routes

In this section, we describe ferry route design in FRA, which consists of designing individual ferry routes and synchronizing routes to have the same length. As discussed above, for synchronization purpose, the ferries need to visit regular nodes within a cell as well as the contact points on the boundary of the cell. In addition, the route segment between contact points should be of the same length. We adopt the following approach in computing the route for each cell. We first construct an initial route that contains only the contact points. Then we insert regular nodes to the route until all nodes are in the route. This can be done by adding nodes to the route segment that is between the two closest contact points to the node.

Given the ferry route computed as above, we need to extend the ferry route, if necessary, to meet the bandwidth requirements. We extend the linear programming problem in (2) to account for synchronization. Specifically, in addition to the constraints for bandwidth requirements for each node, we add new constraints that the route segments between contact points are of the same length.

Now we consider how to synchronize all ferry routes. FRA identifies the ferry route with the maximum length and extends other routes to have the same length. Let L_m be the maximum length of ferry routes. Consider a route i that has length L_i, $L_i < L_m$. To extend route i, FRA increases the detour of ferry route within the radio range of each node. Suppose that the length of the detour for node j in route i is x_j. FRA extends x_j to x'_j such that $(x'_j + 2r)/L_m = (x_j + 2r)/L_i$ where r is the radio range. Note that the achieved data rate for node j is proportional to the portion of time node j is within range of ferries, i.e., $(x_j + 2r)/L_i$ before extension and $(x'_j + 2r)/L_m$ after extension. Therefore, the bandwidth requirements are met after synchronization.

VIII. Simulation Results

In this section, we evaluate the performance of the Message Ferrying schemes using simulations. We first compare the performance of the four algorithms under different network conditions. We then focus on how MF scales with traffic load and the number of ferries used. Our results show that MF scales well with the number of ferries in terms of throughput, delay, and resource usage in nodes and ferries.

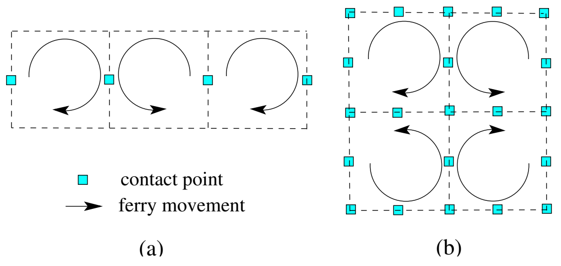


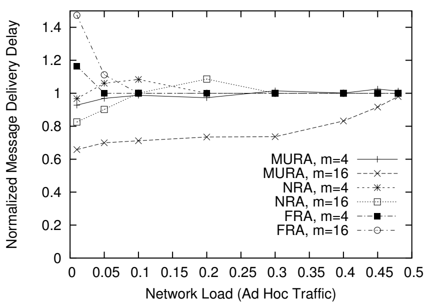
Fig. 8. Synchronization between ferry routes.

We now describe the network settings studied in this paper. In our simulations, n nodes are randomly distributed in a $5km \times 5km$ area. m ferries are deployed which move at a speed of $10m/s$. The transmission range of both the nodes and the ferries is $100m$ and the data rate is 10mbps. We consider both *ad hoc traffic*, where each node sends data to a randomly chosen node, and *sensor traffic*, where all nodes transmit data to a special node called the *sink*. In addition, traffic rates can be uniform, i.e., all flow have the same data rate, or non-uniform where $0.1n$ flows transmit 90% of the total traffic and the rest sends 10% of the total traffic. We assume that data are generated at constant bit rates and set the weights of all traffic to be the same as the data rates. Given node locations and the traffic models, we use the algorithms developed in previous sections to compute the ferry routes. For each setting, the result is averaged over 20 runs with different random seeds.

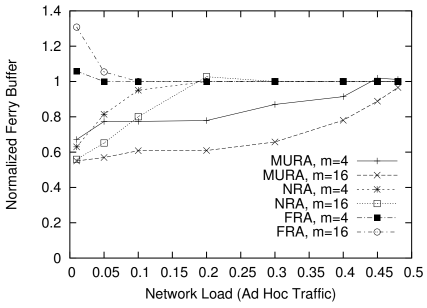
We use the weighted delivery delay defined in (1) as the main performance metric to evaluate the MF schemes. In addition, we consider the resource requirements, namely buffers and energy, in nodes and ferries. We define the *buffer requirement* for a node as the minimum amount of buffer such that no data would be dropped in the node due to buffer overflows. For a ferry route with length L and k ferries, the buffer requirement for a node is bL/fk bits where b is the transmission rate of the node and f is the ferry speed. This is because ferries visit the node every L/fk period of time. Similarly, the *buffer requirement* for a ferry is the minimum amount of buffer such that no data would be dropped in the ferry due to buffer overflow. We also measure the energy consumption in nodes and ferries, which can be approximated using the average transmission rate of nodes and ferries, including both originating and through traffic.

A. Comparison of Algorithms

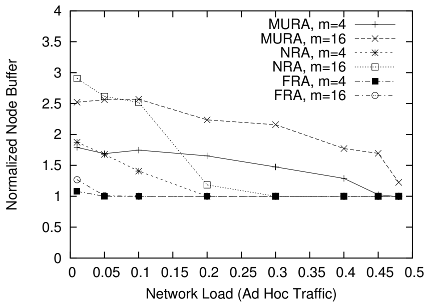
In this section, we compare the performance of the four algorithms presented in previous sections. We first present results for networks with 40 nodes and uniform ad hoc traffic. In Fig. 9, the results are normalized to the performance of SIRA. Fig. 9(a) shows the weighted delivery delay for routes computed using the four algorithms. The results are normalized according to the SIRA algorithm. We make the following observations. First, these algorithms achieve similar weighted delay when the number of ferries is small or the traffic load is high. In both cases the space for route design is



(a) Message delivery delay



(b) Ferry buffer



(c) Node buffer

Fig. 9. Performance of different algorithms with ad hoc traffic.

significantly limited. Second, MURA achieves the best delay when the number of ferries is large. In contrast, FRA performs worst due to the route synchronization which significantly increases the length of each route. Third, SIRA achieves reasonably good performance for ad hoc traffic regardless of its simplicity. Fourth, when the traffic load is relatively high, both NRA and FRA often become the same as SIRA, i.e., a single route is used and all ferries are assigned to this route. This is because relaying data increases the total traffic load in the network, which would be costly when the original traffic load is relatively high.

Fig. 9(b) depicts the buffer requirement in the ferries. We can see that both NRA and MURA require less buffering as compared to SIRA. This is because of the shorter routes generated by these algorithms. In SIRA, the route must visit all nodes, thus the length of the route is normally much larger than routes in NRA or MURA. With a longer route, data will be kept in ferry buffers for a longer period of time, leading to larger buffer requirements in ferries. Similarly, the buffer requirement for FRA is large because of route synchronization.

Fig. 9(c) shows node buffer requirements which differ significantly from ferry buffer requirements in Fig. 9(b). We can see that SIRA uses the smallest number of buffers in nodes. While ferry buffers are determined by the length of the routes, node buffers are determined by the average time between contacts with the ferries. This is because node buffers store data generated at the node before being transmitted to a ferry. In SIRA, while the route is long, the average time between contacts with ferries is short because all ferries are following the same route. So nodes require fewer buffers in SIRA. We note that MURA requires more buffers than NRA which uses nodes to relay data. This is because in NRA, each route is often limited within a cell. In contrast, the routes generated by MURA may span the area due to the random traffic pattern we used. Thus MURA uses more buffers.

We evaluate the performance of the four algorithms in networks with uniform sensor traffic. The results are similar to the case with ad hoc traffic and skipped here due to space limits. One notable difference is that as compared with SIRA, both MURA ad NRA perform better with sensor traffic, which is due to the load balancing effects of using multiple routes.

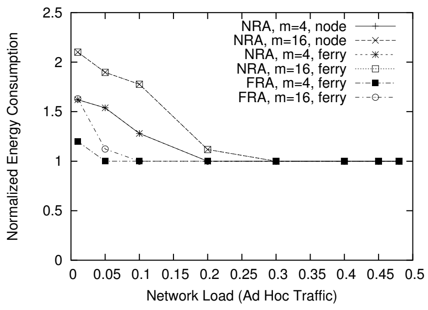


Fig. 10. Energy consumption of different algorithms with ad hoc traffic.

We also run simulations for networks with non-uniform ad hoc traffic and non-uniform sensor traffic. The results are similar but SIRA performs better with non-uniform traffic. This is because with non-uniform traffic, estimation of the weighted delay becomes less accurate, which degrades the performance of MURA/NRA/FRA.

We now consider the energy consumption for these algorithms. Since MURA and SIRA do not relay data, the total energy consumption in the nodes and ferries are the same for both algorithms. In addition, FRA does not relay traffic using nodes, so the energy consumption in nodes is the same as SIRA/MURA. Fig. 10 depicts the energy consumption in NRA and FRA, normalized to that of SIRA/MURA. We can see that in NRA, nodes may need to transmit as much as twice of its own data, which suggests that NRA is not suitable for environments where nodes are constrained in power supplies.

As we can see, these algorithms perform differently in the achieved delay and resource requirements in ferries and nodes. In general, MURA achieves the best delay among the four algorithms. Thus in the rest of this paper, we present only results of MURA.

B. Impact of Traffic Load

We now study the performance of the MF scheme under different traffic loads. Fig. 11(a) shows the weighted delay for ad hoc traffic. We can see that the delay increases with the throughput per node. The increase of delay is slow when the throughput is low but becomes dramatic when the throughput is above some threshold. This behavior suggests that the network should operate under this threshold. Fig. 11(a) also illustrates that this threshold can be increased by using more ferries. The behavior of the tradeoff between throughput and delay shown in Fig. 11(a) is similar to that in connected networks. However, in an MF network, this tradeoff is the consequence of the physical separation between nodes, instead of the burstiness of traffic as in connected networks.

Fig. 11(b) and Fig. 11(c) show the buffer requirements in ferries and nodes, which have similar behavior as the weighted delay. In fact, the increase in both buffer requirement and delay results from the longer routes when traffic load is high. This is because the ferries need to spend more time communicating with nodes. When ferry routes become longer, waiting delay for data in nodes is larger. So the node buffer requirements increase. In addition, ferries need to receive more data from nodes in each visit when traffic load is high, requiring more buffers to hold the data.

C. Impact of Number of Ferries

In this section, we study the data delivery performance with different number of ferries. Fig. 12 shows the results for networks with 40 nodes and uniform ad hoc traffic. Fig. 12(a) shows that for given traffic load, the delay decreases as the number of ferries increases. This is as expected because with more ferries, each ferry needs to carry less amount of traffic and/or visit smaller number of nodes, leading to shorter routes and smaller delays.

When the traffic load is low, e.g., at a rate of 10kbps per node, the improvement in delay due to the increased number of ferries is modest. This is because the delay is dominated by the distance between nodes. However, when the traffic load is high, an increase in the number of ferries can significantly reduce the delay. For example, for a per node throughput of 56kbps, the delay is 1.57 hours when 2 ferries are used while the delay is 11.12 hours when only one ferry is used. This can be explained by the behavior of the tradeoff between throughput and delay shown in Fig. 11. In this case, 56kbps is close to the capacity of a single ferry. Thus the addition of another ferry will reduce the load by half, resulting in modest delay.

Fig. 12(b) and Fig. 12(c) show the buffer requirements in nodes and ferries with different number of ferries. As expected, the buffer requirements decreases as the number of ferries increases. This is consistent with the results shown in Fig. 11.

D. Impact of Traffic Patterns

In this section, we investigate how traffic patterns impact the performance of an MF network. Fig. 13(a) compares the delay under different types of traffic. The traffic load in the figures is defined as nb/mW where b is the per node throughput. We can see that for the same traffic load, networks with non-uniform traffic can achieve lower delay than networks with uniform traffic. This is because for non-uniform traffic, the algorithms can optimize the delay for traffic with larger weights, which improves the total weighted delay. We also note that when the traffic load is low, the delay for both sensor traffic and ad hoc traffic is similar. However, when the traffic load is high, the delay with sensor traffic is larger than that with ad hoc traffic. This is because of the traffic aggregation at the sink in networks with sensor traffic.

Fig. 13(b) shows the ferry buffer requirements. We can see that the buffer requirement with sensor traffic is higher than that with ad hoc traffic. This is because with sensor traffic, a ferry needs to buffer data from all nodes in the route before the ferry meets with the sink and transmits data to the sink. So ferries require more buffers as compared to the case with ad hoc traffic.

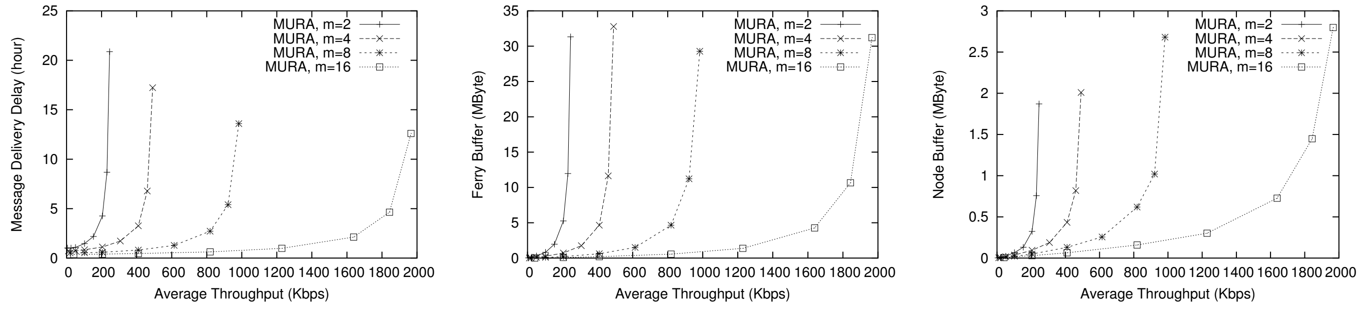
Fig. 13(c) depicts the node buffer requirements under different traffic models. Contrast to ferry buffer requirements, nodes require more buffers in the case with ad hoc traffic. With sensor traffic, data from all nodes are sent to the sink. As compared to ad hoc traffic, this concentration of traffic leads to fewer routes with more ferries in each route. So ferries visit nodes more often which reduces the node buffer requirements.

IX. Related Work

In this section, we review some related work on delay-tolerant networks (DTNs) and mobility-assisted schemes. DTNs are an emerging class of networks where end-to-end paths may not exist between some or all nodes [11], [15]. This is in contrast to the traditional network model which assumes networks are connected. In [15], Jain et al. study routing in DTNs and develop routing algorithms based on different knowledge about network topology.

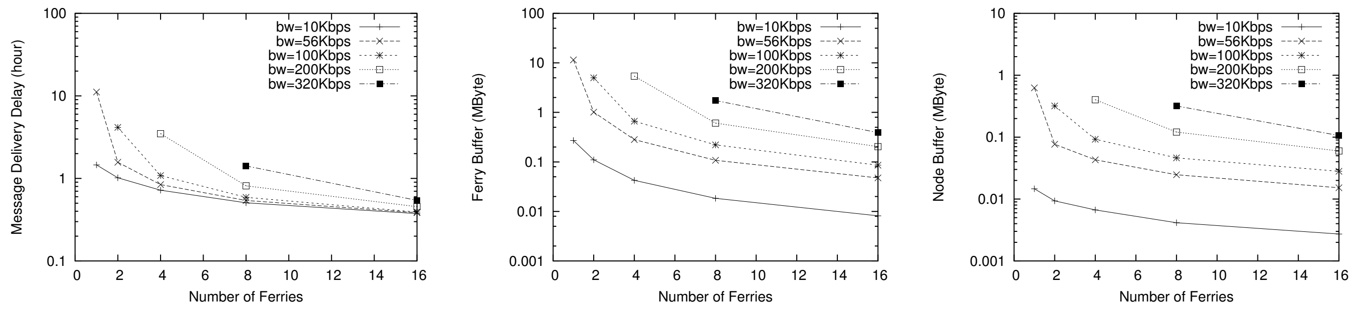
While DTNs may lack end-to-end paths at any instant in time, researcher are observing that node mobility can be leveraged to provide paths over time. The main idea is that a mobile node carries a packet for a period of time as part of realizing a path from source to destination. For example, Vahdat and Becker [26] propose Epidemic Routing in which mobile nodes carry data and exchange data when they meet, essentially flooding data throughout the network. In the Data Mules project, Shah et al. [24] propose to exploit mobile entities in the environment to collect and transport data from sensors to access points, thus conserving energy in resource-limited sensors. In a theoretical paper [13], Grossglauser and Tse prove that mobility can significantly improve the capacity of the network. There is also other work in exploiting node mobility for data transport [6], [9], [14], [17], [25]. The above approaches exploit *existing* node mobility for communication purposes. Our work differs in the use of controlled mobility which enables the network to adapt to applications.

Controlled mobility is an active research area in the robotics community; see [10] and its references. However, there has been little effort in exploiting this capability for communication purposes. Li and Rus [20] consider proactive movement of nodes to deliver messages in a disconnected environment and present an algorithm to compute optimal node trajectories. In [12], Goldenberg et al. present a distributed mobility control scheme to adjust node positions such that energy consumption in communication is minimized. They focus on connected



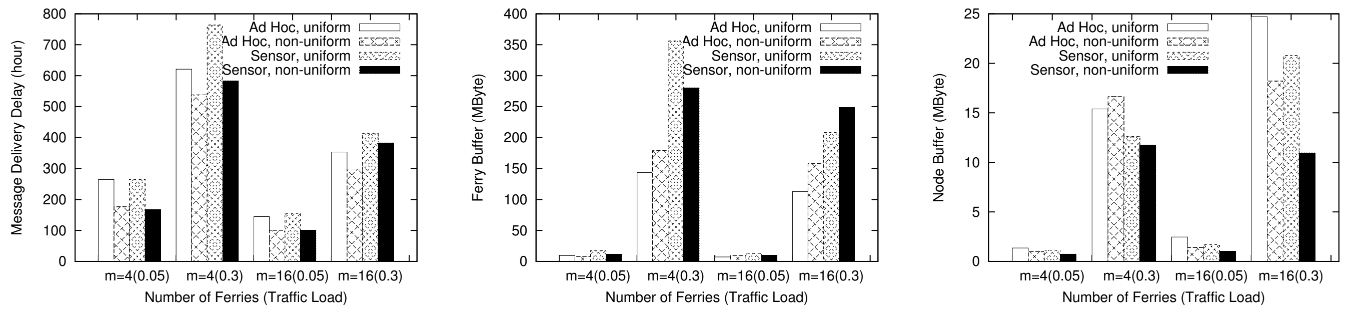
(a) Message delivery delay (b) Ferry buffer (c) Node buffer

Fig. 11. Performance under different traffic loads.



(a) Message delivery delay (b) Ferry buffer (c) Node buffer

Fig. 12. Performance with different numbers of ferries.



(a) Message delivery delay (b) Ferry buffer (c) Node buffer

Fig. 13. Performance under different traffic models.

networks and do not consider the use of mobility to transport data, which are different from our work. In the NIMS project [18], Kaiser et al. propose the use of infrastructure-supported mobility in sensor networks. They propose an infrastructure to support physical reconfiguration via node mobility and describe the use of mobility to transport data and optimize wireless links. While the MF approach uses special nodes as ferries, it does not require an infrastructure to support mobility as in the NIMS project which aims to reduce sensing uncertainty. In [27], Wang et al. use controlled mobility to improve the coverage of sensor networks. However, they do not aim to improve communication performance.

In prior work [29], [30], we introduce the Message Ferrying concept and study networks with a single ferry. In [29], we consider networks with stationary nodes and develop algorithms to compute a ferry's route in order to achieve certain performance objectives. In [30], we develop two variations of the MF schemes for networks with mobile nodes. However, we do not consider the ferry route design problem. In this paper, we study stationary networks with multiple ferries. The use of multiple ferries brings significant challenges to the route design problem.

In recent work by Burns et al. [8], the authors study the problem of augmenting the capacity of a DTN through autonomous agents which move in the network with the purpose of increasing network performance. The idea of "autonomous agents" is similar to message ferries in our scheme. The authors present a control-based approach and develop multiobjective controllers to control the mobility of autonomous agents. The work in [8] focuses on the dynamic control of node movement which differs from our work that addresses the design of ferry routes in a more static environment. In addition, we consider various types of ferry interaction such as ferry relaying and node relaying.

There is some research on topology control [22], [28] which focuses on adjusting node transmit powers to create a desired topology. While this approach adapts network configuration as MF, the MF scheme differs in the use of mobility, instead of transmission power, to control connectivity between nodes.

X. Conclusion

In this paper, we considered the Message Ferrying scheme in delay-tolerant networks and studied the use of multiple ferries. We focused on the ferry route design problem and developed four algorithms to generate ferry routes that meet the traffic demand and minimize the weighted delay. These algorithms represent different strategies in designing ferry routes. We considered algorithms that assume no interaction between ferries, either using a single route (SIRA) or multiple routes (MURA). We also considered algorithms that allow data relaying between ferries directly (FRA) or indirectly (NRA).

We evaluated these algorithms under a variety of network conditions using simulations. We showed that MURA achieves the best delay among the four algorithms while FRA performs the worst. In addition, SIRA has been shown to perform reasonably well when the number of ferries is small or the traffic load is high, regardless of its simplicity. SIRA also requires the least amount of buffers in nodes and does not use nodes to relay data, which is desirable for networks with resource-limited nodes. Based on these algorithms, we study the performance of MF networks. We showed that the MF scheme scales well with the number of ferries in terms of throughput, delay and resource requirements in both ferries and nodes. So for MF networks, scalability can be achieved by adding more ferries.

In this paper, we considered networks with stationary nodes and known traffic demand. In the future, we would like to study message ferrying in mobile networks and networks with dynamic traffic. Moreover, we plan to investigate other types of ferry routes. For example, ferries may visit nodes with different frequencies to provide different services to nodes. Finally, we would like to study the integration of message ferrying with other methods of communication, e.g., direct communication between ferries.

References

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

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

[3] Center for robot-assisted search and rescue http://crasar.csee.usf.edu/.

[4] CNN News: Goodbye vacuum, hello robot! http://www.cnn.com/2003/tech/11/10/vacuum.robot/index.html, Nov 2003.

[5] CNN News: Industrial robot sales surge in west. http://www.cnn.com/2003/tech/biztech/10/22/industrial.robots.ap/, Oct 2003.

[6] A. Beaufour, M. Leopold, and P. Bonnet. Smart-tag based data dissemination. In *First ACM WSNA Workshop*, September 2002.

[7] J. Bentley. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing*, 4:387–411, 1992.

[8] Brendan Burns, Oliver Brock, and Brian Neil Levine. MV routing and capacity building in disruption tolerant networks. In *IEEE Infocom 2005 (to appear)*, March 2005.

[9] A. Chakrabarti, A. Sabharwal, and B. Aazhang. Using predictable observer mobility for power efficient design of sensor networks. In *Information Processing in Sensor Networks*, Palo Alto, CA, April 2003.

[10] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, Apr 2004.

[11] K. Fall. A delay-tolerant network architecture for challenged internets. In *ACM SIGCOMM'03*, 2003.

[12] David Goldenberg, Jie Lin, A. Stephen Morse, Brad Rosen, and Yang Richard Yang. Towards mobility as a network control primitive. In *ACM MobiHoc 2004 (to appear)*, Tokyo Japan, 2004.

[13] M. Grossglauser and D. Tse. Mobility increases the capacity of ad-hoc wireless networks. In *IEEE INFOCOM 2001*, April 2001.

[14] Amir Alexander Hasson, Richard Fletcher, and Alex Pentland. DakNet: A road to universal broadband connectivity. *Wireless Internet UN ICT Conference Case Study*, 2003.

[15] Sushant Jain, Kevin Fall, and Rabin Patra. Routing in a delay tolerant network. In *Sigcomm 2004 (to appear)*, Portland, OR, 2004.

[16] D. Johnson and L. McGeoch. Experimental analysis of heuristics for the STSP. *The Traveling Salesman Problem and its Variations*, 2002.

[17] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. In *ASPLOS'02*, October 2002.

[18] William Kaiser, Gregory Pottie, Mani Srivastava, Gaurav Sukhatme, John Villasenor, and Deborah Estrin. Networked infomechanical systems (nims) for ambient intelligence. *Technical report, UCLA*, 2003.

[19] E. Krotkov and J. Blitch. The defense advanced research projects agency (darpa) tactical mobile robotics program. *International Journal of Robotics Research*, 18(7):769–76, 1999.

[20] Q. Li and D. Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *ACM MOBICOM*, August 2000.

[21] G. Nemhauser, A. Rinnooy Kan, and M. Todd (editors). Optimization. 1989.

[22] R. Ramanathan and R. Hain. Topology control of multihop wireless networks using transmit power adjustment. In *IEEE INFOCOM*, 2000.

[23] Jerome Saltzer, David Reed, and David Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, nov 1984.

[24] R. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling a three-tier architecture for sparse sensor networks. In *IEEE SNPA Workshop*, 2003.

[25] Tara Small and Zygmunt Haas. The Shared Wireless Infostation Model - A New Ad Hoc Networking Paradigm (or Where there is a Whale, there is a Way). In *ACM MobiHoc*, June 2003.

[26] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. *Technical report, Duke University*, 2000.

[27] Guiling Wang, Guohong Cao, and Tom La Porta. Movement-assisted sensor deployment. In *Infocom 2004*, Hongkong, China, 2004.

[28] R. Wattenhofer, L. Li, P. Bahl, and Y. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *IEEE INFOCOM 2001*, April 2001.

[29] Wenrui Zhao and Mostafa Ammar. Proactive routing in highly-partitioned wireless ad hoc networks. In *the 9th IEEE International Workshop on Future Trends of Distributed Computing Systems*, May, 2003.

[30] Wenrui Zhao, Mostafa Ammar, and Ellen Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *ACM MobiHoc 2004*, Tokyo Japan, 2004.