Routing in Space and Time in Networks with Predictable Mobility

Shashidhar Merugu Mostafa Ammar Ellen Zegura
College of Computing, Georgia Institute of Technology,
801 Atlantic Drive, Atlanta, GA 30332
{merugu, ammar, ewz}@cc.gatech.edu
March 15, 2004

***Abstract*— We consider the problem of routing in emerging wireless networks where nodes move around explicitly carrying messages to facilitate communication in an otherwise partitioned network. The absence of a path at any instant of time between a source and destination makes the traditional mobile ad hoc routing protocols unsuitable for these networks. However, the explicit node movements create paths "over time" that include the possibility of a node *carrying* a message before forwarding to another suitable node. Identifying such paths over space and time is a key challenge in these *store, carry and forward* networks. In most of these networks, the mobility of nodes is predictable either over a finite time horizon or indefinitely due to periodicity in node motion. We propose a new *space-time routing* framework for these networks leveraging the predictability in node motion. Specifically, we construct space-time routing tables where the next hop node is selected from the current as well as the future neighbors. Unlike traditional routing tables, our space-time routing tables use both the destination and the arrival time of message to determine the next hop node. We devise an algorithm to compute these space-time routing tables to minimize the end-to-end message delivery delay. Our routing algorithm is based on a space-time graph model derived from the mobility of nodes. We empirically evaluate our approach using simulations and observe improved performance as compared to other approaches based on heuristics.**

I. Introduction

In traditional networks, whether wired or wireless, an intermediate node in a message path stores and forwards messages received from a previous hop to a next hop. Several emerging environments fall into a new paradigm of store, *carry* and forward messaging [1], [2], [3], [4], [5], [6], [7], [8]. In addition to the usual storing and forwarding of data, nodes in these new environments move around explicitly carrying messages to facilitate communication in an otherwise partitioned (or poorly connected) network. For example, a recent project in developing nations uses rural buses to provide Internet connectivity to otherwise isolated and remote villages which do not have any communication infrastructure [2]. Several other examples include robots walking through a sensor farm collecting data from individual sensors [3], hikers monitoring and disseminating weather information in a national park by walking along the park trails [4], and wild-life researchers driving through a forest collecting information about dispersed zebra population [5]. Several other networks such as ad hoc networks of satellites [6] and Inter-planetary Internet [7] share characteristics of store, carry and forward networks.

Two properties of the network topology present a challenge in routing messages in these environments. First, the topology varies with time as links form and break due to node movements. Second, network partitions are a norm rather than an exception in these environments. As a result, any time snapshot of the topology graph is almost always disconnected. Although mobile ad hoc routing protocols are designed to handle network topology dynamics, most of them inherently assume the existence of an end-to-end path between a source and a destination to initiate communication. This premise of path existence may not be valid at any instant of time in these new environments, thus making the traditional mobile ad hoc routing protocols unsuitable for these networks. However, the explicit node movements create paths "over time" that include the possibility of a node *carrying* a message before forwarding to another suitable node.

Consider an example mobile network of four nodes $\{A, B, C, D\}$ as shown in Figure 1. These nodes move in trajectories as indicated. Each sub-figure is a snapshot of the network during a time interval. A link is established between two nodes when they enter each other's radio coverage area. This link lasts as long as the nodes are visible to each other and is broken when they lose radio connectivity as they move away from each other. Note that in this example network, the graph is not connected at any point in time. However, the graph is connected over time by paths, where messages are carried for a certain duration on their way towards the destination. Consider routing a message from a source A starting at time $t = 0$ towards destination B. There are many possible paths that include waiting at intermediate nodes. The message can take a path P_1: A to D at $t = 0$, wait at D until $t = 2$, and D to C at $t = 2$ and C to B at $t = 3$. The message can also take an alternative path P_2 that includes waiting at the source A itself until $t = 2$ and delivering directly to B at $t = 2$.

We make the following observations from this example: *(i)* P_1 and P_2 are paths over space and time from A to B. The message travels from one node to another in space and also waits at a node in anticipation of a future link. *(ii)* A path of least end-to-end latency (P_2) need not always be the first available (P_1). The forwarding decision is best taken by looking ahead into the future, as a future link might deliver the message earlier than current available links. *(iii)* Although P_2 has the least latency and uses only a single hop, spatial hop count minimization, as

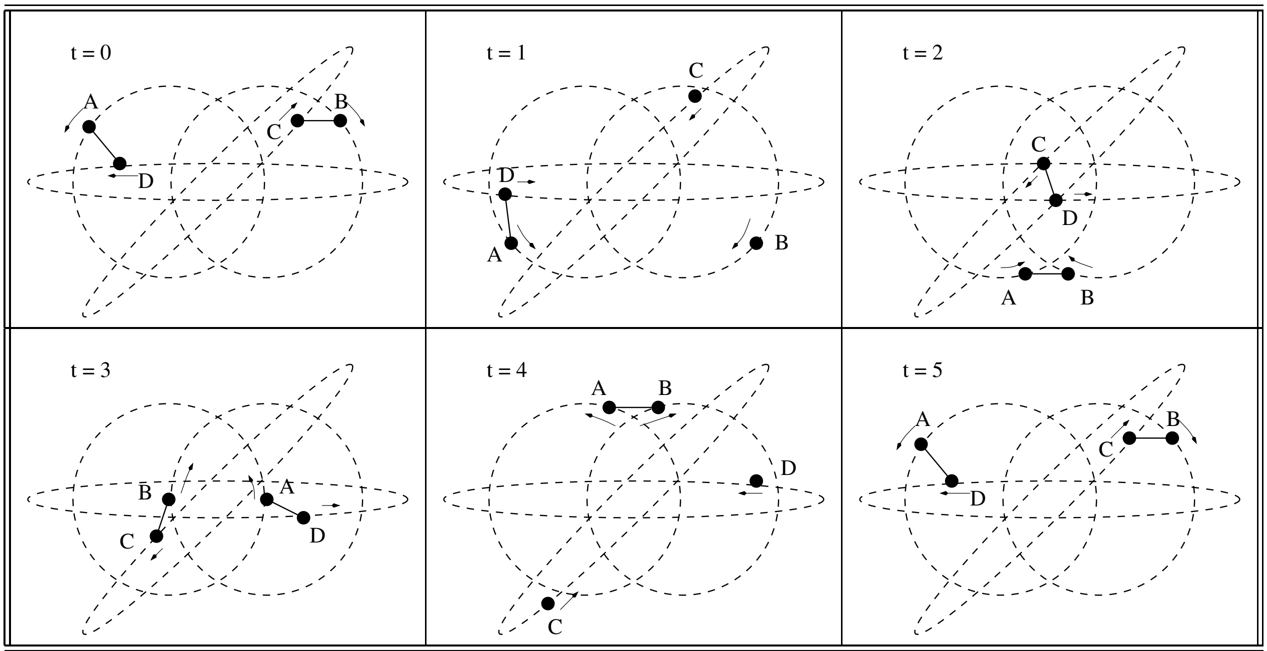

Fig. 1. An illustration of a varying topology of an example mobile network of four nodes A, B, C, D. The dashed circles and ellipses represent the trajectories followed by these mobile nodes. Each sub-figure is a snapshot of the network during the indicated interval of time. Solid lines between nodes represent communication links. Arrows represent their directions of movement.

is done in many flavors of ad hoc routing protocols, does not necessarily minimize the end-to-end latency of messages.

In this paper, we propose a *space-time routing* framework for instances of these networks that have predictable motion, either over finite time horizons or infinite time horizons due to periodicity. Specifically, we solve this routing problem: Given a set of nodes and how they move up to a certain time in the future, our goal is to construct space-time routing tables that specify when and to whom a node must forward a message in order to meet some routing objective. Our solution is based on a *space-time graph* model of the network derived from the mobility of nodes. The space-time graph model captures the dynamic evolution and connectivity over time of the network topology. We devise a routing algorithm using this space-time graph model to select a suitable next hop node from the current as well as the future neighbors. Consequently, the next hop for forwarding a message from a node is a function of both the destination and time, unlike traditional forwarding approaches based only on the destination of a message.

The remainder of the paper is organized as follows. We start with a model of network in Section II and introduce our space-time routing framework in Section III. We derive our space-time graph model in Section IV that forms a basis for our algorithm to compute routes in Section V. We present our evaluation methodology and results in Section VI. Section VII has related work followed by a summary in Section VIII.

II. Model

In this section, we describe our model of a network and how messages are forwarded from source to destination. Table I summarizes the notation used in our model.

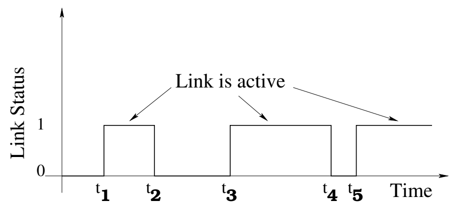

Fig. 2. Time-varying boolean function $L_{ij}(t)$ representing a communication link between two nodes v_i and v_j.

A. Network Connectivity

Consider a mobile network comprising n nodes represented by the set $V = \{v_1, v_2, \ldots, v_n\}$. Since the graph representing the connectivity between these nodes varies with time, we have $G(t) = (V, E(t))$ where V is the set of nodes and $E(t)$ is a time varying set of links between these nodes. The time-varying set of links $E(t)$ can also be represented by link functions of time. $L_{ij}(t)$ is a boolean function that represents whether a communication link is present or not between nodes v_i and v_j at time t. We assume that two nodes have a communication link when they are geographically close to each other within a distance threshold of γ such that the radio quality of a link between the two nodes satisfies the minimum requirement for a successful data communication. Let $P_i(t)$ denote the geographic position of node v_i at time t. We can define the time-varying link function $L_{ij}(t)$ for every pair of nodes (v_i, v_j) as:

$$L_{ij}(t) = \begin{cases} 1 & \text{if} \quad |P_i(t) - P_j(t)| \leq \gamma \\ 0 & \text{otherwise} \end{cases}$$

Figure 2 illustrates the time-varying nature of the link function. We note that there are $n(n-1)$ such link functions, corresponding to every pair of nodes.

Variable	Meaning
t	time index
v_i	i^{th} node
$G(t) = (V, E(t))$	time-varying graph representing network topology
$P_i(t)$	geographic position of i^{th} node
γ	distance threshold of radio coverage
$L_{ij}(t)$	time-varying link function between v_i and v_j
τ	granularity of time dimension in space-time graph
$\mathbf{T}$	duration of predictability

TABLE I

NOTATION USED IN OUR SPACE-TIME GRAPH MODEL

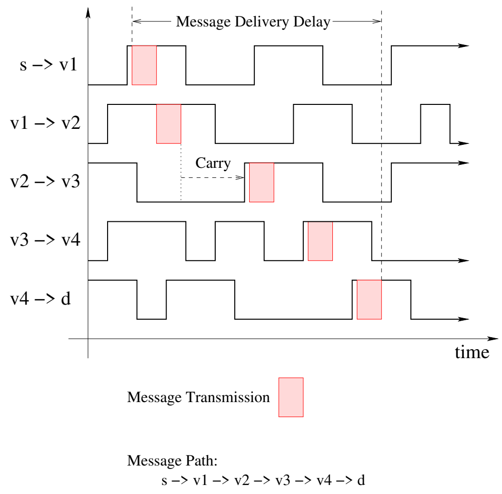

Fig. 3. An illustration of forwarding a message from source s to destination d over a sequence of nodes $(s, v_1, v_2, v_3, v_4, d)$. The links between each pair of nodes in this sequence vary with time and the message is forwarded at appropriate opportunities.

We define duration of predictability $\mathbf{T}$ as the time horizon to foresee the network topology evolution. Specifically, the geographic position function $P_i(t)$ is known from current time (t_c) up to $(t_c + \mathbf{T})$ in the future for all nodes $v_i \in V$. Mobile networks with periodic node movement, as shown in the example network in Section I, present a special case for prediction. Periodicity in the individual node motion provides a succinct representation and implicitly gives us an infinite duration of predictability (i.e., $\mathbf{T} = \infty$).

B. Forwarding Messages

Consider forwarding a message $M = (s, d, t_s, m)$ from a source s to a destination d. M arrives at the source at t_s and takes m time units for transmission from one node to another [1]. This message M can be transferred from v_i to v_j starting at time t_x only if $L_{ij}(t) = 1, \ \forall t \in \{t_x, t_x + m\}$. In other words, the message transmission interval $[t_x, t_x + m]$ is a subset of an *active* time interval of the link function L_{ij}. Consequently, a message forwarding path is a schedule $P(s, d, t_s, m) = (\pi, \omega)$. Here $\pi = (s, v_1, v_2, \ldots, v_l, d)$ is a sequence of nodes that comprise a path from source to destination and $\omega = (t_0, t_1, \ldots, t_l)$ are the times of start of transmission at respective hops where *(i)* $t_i \geq (t_{i-1} + m)$ – message has to be received completely before forwarding to next hop node and *(ii)* $L_{v_i v_{i+1}}(t) = 1, \ \forall t \in [t_i, \ t_i + m]$ – link is active during the message transmission interval.

Figure 3 illustrates message transmission on a path comprised of dynamically changing links. Intuitively, one can imagine a "band" (of message transmission duration) going from left to right as we travel from source to destination. Occasionally this band has to "slide" towards right when there is no link available for transmission immediately to the chosen next hop node. For example, in the figure, after receiving the message from v_1, v_2 could not forward it immediately to v_3 but had to carry for some time until the link $v_2 \rightarrow v_3$ became active.

III. SPACE-TIME ROUTING FRAMEWORK

A. Routing Tables

In our space-time routing framework, the message forwarding path $P(s, d, t_s, m) = (\pi, \omega)$ is realized collectively by local forwarding decisions of each intermediate node. Unlike many traditional routing protocols that use only spatial connectivity information, our space-time routing framework considers the time dimension as well as space. Since the network topology changes with time, the "best" outgoing link for a message depends not only on its destination but also on the topology evolution. Therefore, the next hop in our space-time routing framework is a function of both the destination and time. Figure 4 contrasts our space-time routing table with a traditional routing table. A space-time routing table is a matrix of two dimensions, one for destination addresses and the other for instances of time. Entries in this table are actions that might include carrying a message for a certain duration before forwarding to another node. For example, in the figure, a message that is ready for transmission at $(t - 2)$ to a destination d_i has to be carried for 2 time units and forwarded on a link to node H. We also note that the next hop node to the same destination d_i could be different at different times due to variation in network topology. Again, in the example figure, next hop node (H) at t is different from that (L) at t'.

[1] For simplicity, we assume that all communication links have equal transmission speed and that propagation delay is negligible. This assumption ensures that the message takes the same amount of m time units on any communication link. It is straightforward to adapt the formalism for varying link speeds and significant propagation delays.

Destination	Next Hop Node
$\vdots$	$\vdots$
d_i	h_i
$\vdots$	$\vdots$

(a) Traditional Routing Table

Destination	Time of Message Forwarding Lookup							
Address	0	$\dots$	$(t-2)$	$(t-1)$	t	$\dots$	t'	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
d_i			Carry 2 Forward H	Carry 1 Forward H	Carry 0 Forward H		Carry 0 Forward L	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

(b) Space-Time Routing Table

Fig. 4. Comparison of a space-time routing table with a traditional routing table

B. *Message Transmission Schedule*

When a message arrives at an empty node, it looks up the space-time routing table to determine the next hop node and transmission time. Subsequent messages that arrive at the same node look up the routing table to schedule their transmission. However, conflicts can occur as other messages might be queued at the node to use the same transmission opportunity. In that case, we look up the space-time routing table again with a new virtual arrival time that is past the missed transmission opportunity. We may need to repeat this look up operation several times until we find a feasible transmission time for the message.

C. *Routing Table Construction*

We define routing in space and time as the construction of space-time routing tables that aid in forwarding decisions to meet a particular routing performance objective (e.g., to minimize end-to-end delay). A naive solution to the routing problem is to list all feasible paths in space and time from the source to the destination. Among the set of feasible paths, one could search for an optimal path that, say, minimizes overall end-to-end delivery delay of a message. Although this solution is simple, it is computationally expensive; the number of paths can be exponential in the graph size.

Our goal, by contrast, is to

1) assimilate time-varying link information into a space-time graph model (Section IV),
2) apply a routing algorithm on these space-time graphs to identify these optimal paths efficiently (Section V), and
3) extract forwarding information from the computed paths into space-time routing tables.

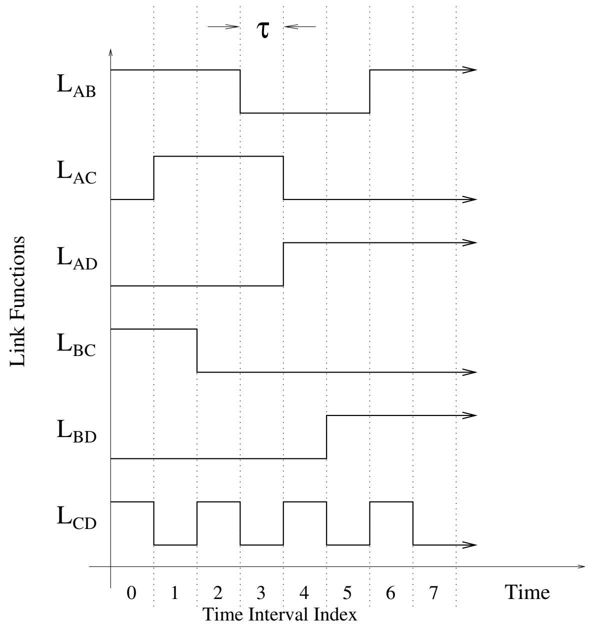

Fig. 5. Time-varying link functions of an example mobile network with four nodes $\{A, B, C, D\}$ (different from the one shown in Figure 1).

IV. Space-Time Graph

We start with an overview of our space-time graph model followed by details of its construction. For clarity, we illustrate this construction using a hypothetical mobile ad hoc network with four nodes $\{A, B, C, D\}$ whose link functions over time are given in Figure 5. We follow the notation summarized in Table I.

A. *Overview*

Our *space-time graph* model captures both the space and time dimensions of the network topology. The main idea is to construct a layered graph, where each layer corresponds to a discrete time interval (of length τ) in the life of the network.

We will return shortly to the issue of discretizing time, but first we describe the layered graph construction.

Starting from the set of nodes V in the network, multiple copies of each node are made and stacked up as layers on each other as illustrated in Figure 6. Each layer of this graph has one copy of every node in the network and a *column* of vertices in this layered graph corresponds to a single node in the network.

We introduce two kinds of links on this layered graph – *temporal* and *spatial* links. A formal description of the spatial and temporal link construction is given in the Appendix. Directed temporal links, shown in Figure 6, connect "time-copies" of the same node between consecutive intervals of time and hence remain within a single column, i.e., the vertical time dimension. Traversing a temporal link denotes "carrying" a message by a node.

Forwarding a message from one to node to another in the network is represented by a traversal of a spatial link. Construction of spatial links on this layered graph follows the link functions L_{ij}. As an example, the spatial links going from node A to node C are illustrated in Figure 7 following the link function L_{AC} drawn in Figure 5. Directed spatial links go from a vertex in one column to a vertex in another column, i.e., span the horizontal space dimension. In order to incorporate message transmission delays, these spatial links are *delayed* so that the destination vertex of each spatial link is as many time layers later than the source vertex as the transmission delay of a message. For example, a transmission of a message of size τ units from node A during interval 1 to node C is represented by the $A_1 \rightarrow C_2$ spatial link. Similarly, a transmission of a message of size 2τ units is denoted by $A_1 \rightarrow C_3$ spatial link, and so on.

We introduce a link coloring scheme to distinguish paths available on the space-time graph for messages of different sizes. Although messages can be arbitrarily long, we consider discrete bins of width τ for message sizes and each bin is assigned a unique color drawn from a set of colors $\{0, 1, 2, \ldots, C_{max}\}$. Temporal links are colored using a special "wild-card" color 0 that matches all colors. The color of a spatial link is determined by the message transmission delay that the link represents. For example, the spatial links $A_1 \rightarrow C_2$, $A_2 \rightarrow C_3$ and $A_3 \rightarrow C_4$ are all assigned color number 1 as they represent a delay of τ units. Similarly, $A_1 \rightarrow C_3$ and $A_2 \rightarrow C_4$ are colored number 2 as they correspond to message transmission delays of 2τ units.

Routing a message (s, d, t_s, m) of size m from source node s to destination node d starting at time t_s at the source is now equivalent to finding a colored path on this space-time graph. The source s and start time t_s accurately describe the time layer and vertex corresponding to the source. The destination d is represented by the column of vertices spanning all time layers corresponding to node d. The message size m corresponds to a bin of color c where $(c-1)\tau < m \leq c\tau$. A path on the space-time graph using spatial links with color c and wild-card colored temporal links represents a feasible route from source to destination. The shortest path among all these feasible paths corresponds to a route of least end-to-end delay.

A shortest path algorithm can be applied to this space-time graph to find routes. This space-time graph can also be enhanced by assigning appropriate weights to the spatial and tem-

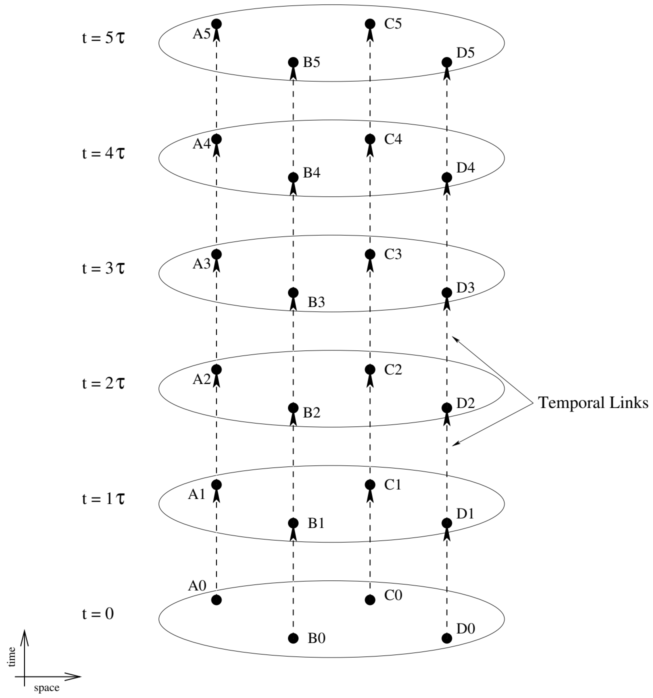

Fig. 6. Temporal links of the layered space-time graph. Each layer corresponds to a discrete time interval in the life of the network. Temporal links connect the same node across consecutive layers.

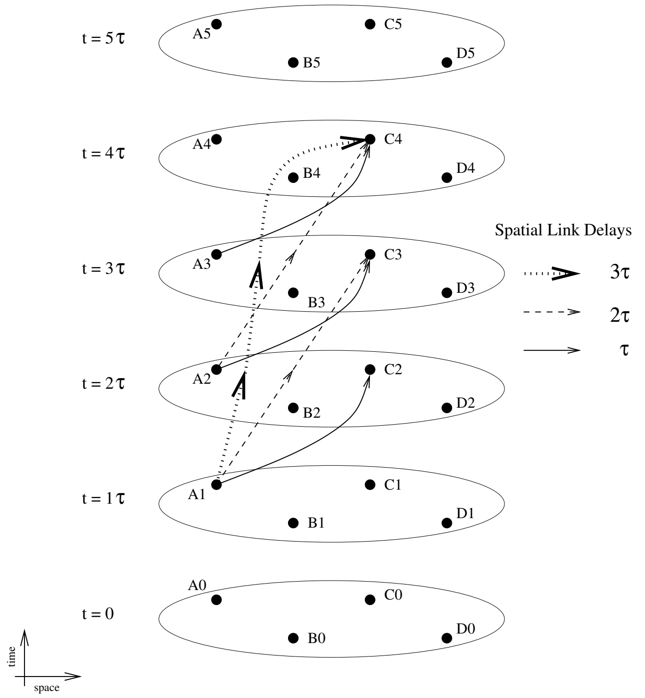

Fig. 7. Spatial links of the space-time graph. For simplicity, spatial links that go only from node A to node C following the link function L_{AC} of Figure 5 are shown here. The spatial links are colored appropriately to incorporate message transmission delays as indicated in the legend.

poral links to reflect different routing criteria. For example, one could also assign weights to the spatial links proportional to the cost of a message transmission. A shortest path algorithm on this weighted graph finds routes that use least overall cost of message transmission. Similarly, we could also assign weights to the temporal links to study the effects of queuing and costs of storing messages at intermediate nodes.

B. Level of Time Granularity

We now consider the issue of discretizing time by choosing an appropriate value of τ. Our choice for the length of this time interval τ is motivated by a need for an accurate, yet succinct representation of the dynamic network topology. In general, a small value of τ implies a large number of layers in the space-time graph whereas a large value of τ would not be able to capture most of the link activity. We describe below a choice of such a representative time unit that is small enough to capture the network topology evolution, but long enough to describe it in less number of layers.

Assuming the network topology is predictable up to a finite horizon we obtain for each pair of vertices, v_i, v_j, all the intervals during which the link between v_i and v_j is active. We then consider the set $\mathcal{T}$ consisting of the lengths of all the active intervals corresponding to all pairs of vertices. Since every link transition in the network is separated by some time length in $\mathcal{T}$, a time unit that is an integral divisor of all the lengths in $\mathcal{T}$ would allow us to capture all the link transition information. Hence, we set the representative time unit τ for the construction of our space-time graph to be the greatest common divisor of all the time lengths in the set $\mathcal{T}$. Figure 5 illustrates the selection of τ based on the greatest common divisor criteria.

Link function approximation: Since the link functions of different pairs of nodes are possibly independent of each other and the value of τ depends on the lengths of the active intervals of all pairs of nodes, it is highly likely that the τ obtained using the greatest common divisor criteria might have a very small value. This would in turn result in a large number of layers in the space time graph. To address this problem, we propose the following approximation to the link functions.

We start with a lower bound τ_l that we wish to maintain for a value of τ. A good estimate of τ_l can be derived from the least possible size of messages in the network. Our first approximation ignores any link activity that lasts less than τ_l time units as the link does not last long enough for any message transmission. Next, we align the link function transition points along multiples of τ_l by shortening the link activity interval. For example, with a choice of $\tau_l = 2$, an interval of $[9, 15]$ is approximated to $[10, 14]$ such that the transition points are multiples of τ_l. When the transition points of link functions are thus aligned along the multiples of τ_l, the lengths of inter-transition gaps in the transition point collection would also be multiples of τ_l. Therefore, our representative time unit τ would an integral multiple of τ_l.

Although this approximation loses potential transmission time at the beginning and end of an active time interval, this loss is always less than τ_l time units at either end point. With a suitable choice of τ_l, the benefits of such an alignment would outweigh the approximation losses incurred.

V. Routing using the Space-Time Graph

In this section, we formulate the problem of routing messages in a dynamic network in terms of finding the shortest paths in the space-time graph described in the previous section. We also propose an algorithm that exploits the structure in the space-time graph in order to efficiently compute the shortest paths between every pair of nodes beginning at any instant of time.

A. Problem Formulation

Of all the paths from source to destination, we are interested in a path that minimizes the end-to-end message delivery delay. Minimizing delay is perhaps the most common routing performance metric. However, we do note that other routing performance metrics, such as number of spatial hops, are potentially of interest, especially when every message transmission is expensive. Although our focus in this section is to minimize delay, the space-time graphs are extensible to incorporate alternative routing performance optimizations.

The space-time graph of a dynamic network topology has the property that the end to end delay for any message is exactly equal to the length of the path traversed in the space-time graph. Now, routing a message of size m at time t from a source node v_i to a destination v_j corresponds to finding a path of color c (where $(c-1)\tau < m \leq c\tau$) in this space-time graph from vertex v_{it} to any vertex in the "column" set $\{v_{js} | \ 0 \leq s < \mathbf{T}\}$ of the space-time graph. Specifically, minimum delay routing is equivalent to finding the shortest path, i.e, the path with the least overall delay $|s - t|$. This can be formally stated as the optimization problem,

$$\min_{p \in G',\ \mathrm{color}(p)=c,\ v_{it} \xrightarrow{p} v_{js}} |s-t| \quad 0 \leq s < \mathbf{T} \tag{1}$$

where p is a path of color c in the space-time graph G' from the source vertex v_{it} corresponding to the source node v_i at time t and some time copy v_{js} of the destination node v_j.

B. Routing Algorithm

Our routing algorithm is closely related to the Floyd-Warshall algorithm that computes the shortest paths between all pairs of vertices in a graph [9], [10]. If we only need single-source multiple destination paths, we could techniques similar to the ones described here to design a routing algorithm based on Dijkstra's shortest path algorithm [11], [10].

First, we observe that routing in space time can be formulated as a shortest path problem. Hence, an obvious approach to solve the problem is to apply shortest path algorithms such as the Floyd-Warshall algorithm or the Dijkstra's algorithm. However, since the topology of a mobile network varies with time, a shortest path computed at one time instant may no longer be shortest or even exist at a later time. Hence, neither the Floyd-Warshall nor the Dijkstra's algorithm, in their original form, are directly applicable in computing shortest paths that hold over time. Therefore, we propose techniques to adapt these shortest path algorithms to reflect the notion of "paths across time" on these dynamic network topologies. More specifically, we use our space-time graph and its properties as a basis to compute the shortest paths that are preserved in both space and time.

1) Delay Invariant: The Floyd-Warshall algorithm uses a dynamic programming formulation of the all-pairs shortest path problem to achieve a time complexity of $\Theta(|V|^3)$, where $|V|$ is the total number of vertices in the graph. In a space-time graph G with n nodes and a duration of predictability of $\mathbf{T}$ time units, the total number of vertices is $n\mathbf{T}$ – one vertex for each node at each time unit. A naive adaptation of the Floyd-Warshall algorithm would require $\Theta(n^3\mathbf{T}^3)$ operations. Since $\mathbf{T}$ is usually much larger than n, this would be a very inefficient solution. Fortunately, it is possible to exploit the structure of the space-time graph to reduce the search space and achieve a much better computational complexity of $\Theta(n^3\mathbf{T})$.

In particular, we identify the following *delay invariant* for the shortest paths in our space-time graph.

$$\mathcal{D}(i, j, (t-x), c) \leq \mathcal{D}(i, j, t, c) + x$$

where $\mathcal{D}(i, j, t, c)$ denotes the delay of the shortest path of a specified color c from a source node v_i to a destination node v_j beginning at time t at the source node. Alternatively, $\mathcal{D}(i, j, t, c)$ is the delay corresponding to the shortest path of color c from the vertex v_{it} in the space time graph to any vertex in the column corresponding to v_j (i.e. $\{v_{js}|\ 0 \leq s < \mathbf{T}\}$). The delay invariant states that any message that starts from a source node v_i towards a destination node v_j, but begins earlier than t, say $(t-x)$, will not suffer a delay greater than $(\mathcal{D}(i, j, t, c) + x,)$. To see why the above delay invariant holds, we observe that the delay $(\mathcal{D}(i, j, t, c) + x)$ corresponds to a valid path consisting of temporal links from $v_{i(t-x)}$ to v_{it}, and the shortest path from v_{it} to the space-time nodes corresponding to v_j. Hence, the delay $(\mathcal{D}(i, j, t, c) + x)$ forms an upper-bound on the shortest possible delay $\mathcal{D}(i, j, t-x, c)$ from $v_{i(t-x)}$ to the space-time nodes corresponding to v_j. Note that the delay invariant holds irrespective of the color of the message. We use this delay invariant property to reduce the search space when computing the shortest delay paths.

2) Algorithm Details: Routines 1 and 2 show the pseudocode of our algorithm for computing the shortest paths for all possible messages in the space-time graph. There are two main components in our algorithm - the initialization phase and the shortest path computation phase. Each of these phases needs to repeated for all possible path colors $c \in \{1, \cdots, C_{max}\}$ in order to obtain the complete routing tables.

The initialization phase computes the shortest path delay between a source vertex v_{it} and any vertex in the set $\{v_{js}|\ 0 \leq s < \mathbf{T}\}$ for all $i, j, t : 1 \leq i, j \leq n,\ 0 \leq t < \mathbf{T}$ based only on paths consisting of at most one spatial link, i.e., not involving any intermediate nodes. This step, however, incorporates all the temporal link information into the delay matrix. In the initialization phase, for each edge $(v_{it}, v_{j(t+c)})$ of a specified color c, we set the $\mathcal{D}^0(i, j, t, c)$ to c. Also, when there is no direct link between v_i and v_j at t, but $\mathcal{D}^0(i, j, (t+1), c)$ has some finite value d, then we set $\mathcal{D}^0(i, j, t, c)$ to $d+1$. This second operation is based on the the delay invariant property described earlier.

The shortest path computation phase is based on dynamic programming formulation that involves decomposing the problem in the same way as in Floyd-Warshall's original algorithm. More specifically, our algorithm computes for each iteration $k,\ 1 \leq k \leq n$ (outermost loop of the algorithm), the shortest path between a source vertex v_{it} and any vertex in the set $\{v_{js}|\ 0 \leq s < \mathbf{T}\}$ using only intermediate nodes drawn from the subset $\{v_1, \cdots, v_k\}$. Let $\mathcal{D}^k(i, j, t, c)$ denote the shortest path delay computed in k^{th} iteration. The following equation establishes a relationship between the shortest path delays computed in k^{th} iteration with those in the previous $(k-1)^{th}$ iteration. This relationship is also illustrated in Figure 8.

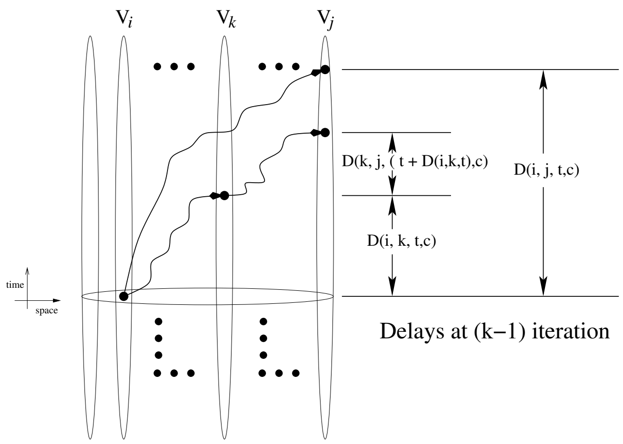

Fig. 8. Computing the shortest path delay from source node v_i starting at time t to destination node v_j. At the k^{th} iteration, we compare the new path using node v_k to the shortest delay calculated in previous $(k-1)$ iterations.

$$\mathcal{D}^{(k)}(i, j, t, c) \leftarrow \min \begin{cases} \mathcal{D}^{(k-1)}(i, j, t, c) \\ \\ \mathcal{D}^{(k-1)}(i, k, t, c) + \\ \mathcal{D}^{(k-1)}(k, j, (t + \mathcal{D}^{(k-1)}(i, k, t)), c) \end{cases}$$

Intuitively, at any time t, for any pair of nodes (v_i, v_j), consider all paths starting from v_i at time t and reaching v_j at some later time, such that the intermediate vertices in these paths is taken from the subset $\{v_1, \cdots, v_k\}$. This operation corresponds to the k^{th} iteration of the outermost loop of our algorithm. Let p be the shortest path among all of them. If p uses only nodes from $\{v_1, \cdots, v_{k-1}\}$, then it would have already been discovered in the $(k-1)^{th}$ iteration. On the other hand, if p uses v_k as an intermediate node, then it can be broken into two sub-paths p_1 and p_2, where p_1 goes from v_i to v_k, starting at time t and taking d_1 time units. The second sub-path p_2 goes from v_k to v_j, starting at time $t+d_1$ and taking d_2 time units. Thus p can be constructed by looking at the shortest path from v_i to v_k starting at t and the shortest path from v_k to v_j starting at $t + d_1$, where both paths use nodes only from the subset $\{v_1, \cdots, v_{k-1}\}$ and hence were found by the $(k-1)^{th}$ iteration. Successively performing this computations till $k = n$ provides the desired shortest delay over all the paths in the space-time graph.

It is easy to see that the initialization phase has a complexity of $\Theta(n^2\mathbf{T})$ and the shortest path computation has a complexity of $\Theta(n^3\mathbf{T})$ based on the `for` loops as the rest of the computations and comparisons are constant time operations.

VI. Evaluation

In this section, we evaluate, by simulation, the performance of the routing algorithm described in Section V by comparing

Routine 1 Shortest_Delay_Path

Input: Space-time graph $G' = (V', E')$ with colored edges, number of vertices n, time period $\mathbf{T}$, message color c
Output: Shortest path length from node v_i at time t to node v_j of color c $\mathcal{D}(i, j, t, c),\ 1 \leq i, j, < n;\ 0 \leq t < \mathbf{T}$
Method:

```
D = Initialize_Delay (G', n, T, c)
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            for t = 1 to T do
                d_curr = D(i, j, t, c)
                d_1 = D(i, k, t, c)
                d_2 = D(i, k, (t + d_1), c)
                d_alt = d_1 + d_2
                if (d_alt < d_curr) then
                    D(i, j, t, c) = d_alt
                    intHop[v_i][v_j][t] = v_k
```

Routine 2 Initialize_Delay

Input: Space-time graph $G' = (V', E')$ with colored edges, number of vertices n, time period $\mathbf{T}$
Output: Shortest path length from node v_i at time t to node v_j of color c, $\mathcal{D}(i, j, t, c),\ 1 \leq i, j, < n;\ 0 \leq t < \mathbf{T}$ based on temporal links and single spatial hops.
Method:

```
for t = T - 1 downto 1 do
    for i = 1 to n do
        for j = 1 to n do
            if((v_{i,t}, v_{j,t+c}) ∈ E_c) then
                D(i, j, t, c) = c
            elseif (D(i, j, (t + 1)) < ∞) then
                D(i, j, t) = D(i, j, (t + 1)) + 1
            else
                D(i, j, t) = ∞
```

with three other algorithms based on heuristics. We also explain our evaluation methodology and analyze results from our experiments.

A. Heuristics

Several routing algorithms can be designed using heuristics drawn from routing in traditional networks. We describe three such heuristics below.

1) **Hot Potato Routing** In hot potato routing (HPR), a node forwards a message to the earliest neighbor. When multiple neighbors are available simultaneously, it picks one at random to forward the message. Due to the variation in a node's neighbors over time, hot potato routing, in some sense, "throws the message around" the network until it reaches its final destination. The life-time of a message in the network is limited by a maximum count on the number of hops. Although this local greedy heuristic minimizes the wait time of messages at every node, forwarding loops can occur.
2) **Most Frequent Neighbor Routing** Nodes maintain a set of frequent neighbors in this most frequent neighbor routing (MFN) approach. Neighbor frequencies are computed from observations over a long duration. In most frequent neighbor approach, a node forwards a message to its most frequent neighbor. Like hot potato routing, most frequent neighbor routing is also a greedy heuristic that tries to minimize wait time at a node by seeking the most frequent neighbor as a next hop. A message is no longer forwarded when it reaches a maximum limit on the number of forwarding hops.
3) **Epidemic Routing** In epidemic routing (ER) [12], a node forwards a message to all its neighbors. This forwarding operation is repeated both in space and time. The scope of this flooding operation is limited by an expiration time associated with each message. Unlike HPR and MFN where delivered messages are flushed out of the network, messages continue propagating in ER in different parts of the network even after a copy of the message might have been delivered to its destination.

B. Methodology

We have written an event-driven message based simulator for this study. A network of 128 nodes is simulated on a two dimensional plane. We consider a square region of $\{(0, 0), (1000m, 1000m)\}$ as our simulation area. We use two kinds of node mobility models – one with predictability up to a finite time horizon and the other with periodic node motion that gives infinite predictability. We describe both models here:

- **Finite Time Horizon** When simulation begins, the nodes are populated randomly on the square simulation area. Nodes move in straight lines with a randomly chosen motion vector (*speed, direction and duration*), a variation of the random walk mobility model used in evaluating many ad hoc network routing protocols [13]. Each node changes its motion vector between 1 and 4 times chosen uniformly at random during the entire simulation. Speeds are also sampled uniformly from $[1, 5]$ m/s. A node's direction of motion is chosen from the set $\{\pi/4, \pi/2, 3\pi/4, \ldots, 2\pi\}$ radians uniformly at random. We assume that nodes within 50 distance units of each other can communicate. The finite time horizon extends to 512 simulation units for predictability in node motion.
- **Infinite Time Horizon** Nodes move in circular trajectories[2]. The centers of the circles are chosen uniformly at random from the grid points of the square simulation area. The radii of circles are chosen uniformly from the set $\{100, 200, 300, 400, 500\}$. The start location (at $t = 0$) of each node is also selected uniformly at random on the circumference of the circle at any multiple of $\pi/8$ radians. The nodes are assumed to move at a constant speed. We chose the angular speeds of motion again uniformly at random from the set $\{\pi, \pi/2, \pi/4, \cdots, \pi/512\}$ radians per time unit. As earlier, we consider a distance threshold of 50 units to allow communication between two nodes. Since the nodes in the network move in a periodic fashion, the network topology has infinite predictability.

[2] We have chosen circles for simplicity, but our results extend to other geometries as well. What matters more in the message communication is the likelihood of intersections of node trajectories than the actual geometric shape of these trajectories.

We have considered two kinds of message simulation. In the first one, we simulate different forwarding algorithms with only one message in the network. This exercise is to bring out the best case performance of each message forwarding algorithm for a suitable comparison. In the second case, we consider a realistic message traffic model with bursty arrivals of messages. We choose 128 source-destination pairs for exchanging messages. The sources and destinations are selected uniformly at random from the entire node population. The inter-arrival time between bursts is drawn from an exponential distribution with a mean of 10 time units. The length of each burst of messages is sampled uniformly between $[5, 15]$ messages. Our evaluation results are collected from multiple simulation runs, each with different seeds for random number distributions.

The simulation proceeds in increasing order of time-stamps of events. Message arrivals and departures at a node are the main events in our simulation. For each message, we keep track of the nodes visited on its way from source to destination. We also record the arrival and departure times of this message at each of the nodes on the path taken from source to destination. When a message arrives at a node, the forwarding process estimates its departure time and the next hop neighbor from the current node according to the simulated routing algorithm. We have implemented our routing algorithm to identify shortest paths in space and time (SPST) described in Section V and each of the three heuristics based algorithms described in Section VI-A.

C. Simulation Results

1) Network Topology: Our choice of mobility parameters generates a network that is always disconnected and rapidly varying. For example, in our random-walk motion simulations, a network of 128 nodes has about 61 links on average (standard deviation = 7.8) at any instance of time. About 6.5 links either form or break up every simulation time unit. Clearly, the number of links are far less to ensure paths between any source destination pair. This poor network connectivity limits the applicability of traditional ad hoc routing protocols. Also, even if a path does exist between a pair of nodes, it is highly likely that it may be disrupted frequently leading to a large overhead in repair and maintenance of routes using traditional ad hoc routing protocols.

Routing Algorithm	Delivery success rate (%)
SPST	100.00
HPR	15.7
ER	39.94
MFN	60.08

TABLE II
SUCCESS OF MESSAGE DELIVERY

2) Success of message delivery: In this experiment, we studied the success rate of delivery of each of the four routing algorithms. A message is successfully delivered if it reaches its destination before it exceeds its resource limit (e.g., maximum number of hops in hot potato and most frequent neighbor routing). We define success rate of a routing algorithm as the ratio of the number of messages that are successfully delivered to the total number of messages that are forwarded. Table II shows the percentage of success rates of the four routing algorithms. Although the network topology is always disconnected at any instance in time, there are paths in space and time that keep the network connected "over time". Our SPST routing algorithm identifies these paths and is clearly able to route almost all messages to their destinations resulting in a very good success rate of message delivery. It is possible, however, that there may not be a space-time path from a source to destination within the time horizon and our algorithm fails to deliver in such a case. Our simulations indicate that such an occurrence is rare and depends on particular node mobility patterns. The other three heuristics based routing approaches perform poorly. They propagate the message around the network until it is either delivered or runs out of resources. Their success of delivery depends on the likelihood of occurrence of the right sequence of links to create a path to the destination and actually using those links at the right time. The success rates of HPR, MFN, and ER can be improved by increasing the resource limit but at the cost of increasing resource usage. Other factors also determine the success rates of these heuristics. For example, the structural properties of the network topology, whether it is clustered or not, sparse or dense, determine the number of nodes reached by a flooding algorithm such as ER.

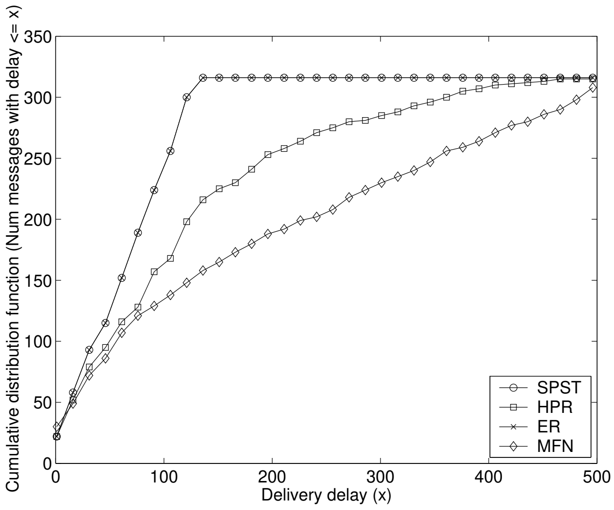

Fig. 9. Distribution of message delivery delay from source to destination for various routing algorithms

3) End-to-end message delivery delay: In this experiment, we studied the performance of the four routing algorithms with respect to the delay of message delivery. We define the end-to-end message delivery delay as $\Delta = |t_d - t_s|$ where t_s is the message arrival time at the source and t_d is the time of delivery at the destination. The message delivery delay along a route includes the transmission delays of comprising links as well as *opportunity* delays where a message is carried by a node for a certain duration until it meets a suitable node for forwarding.

Figure 9 plots a cumulative distribution function (CDF) of end-to-end message delivery delays. For a fair comparison, we have included delays of only those cases when a message is

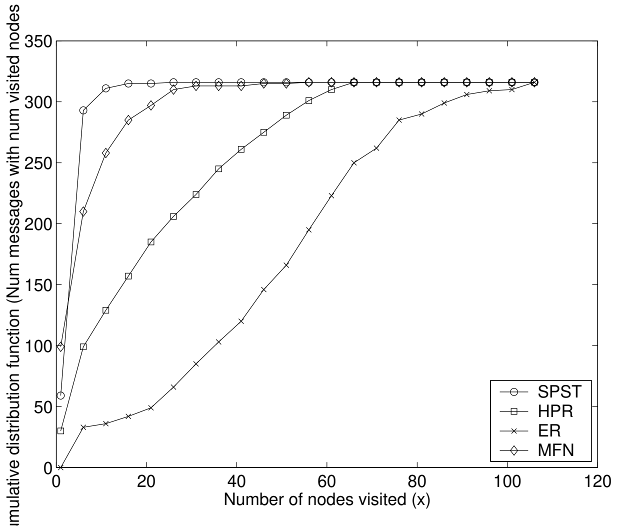

Fig. 10. Distribution of number of nodes visited while forwarding a message by various routing algorithms

successfully delivered by all the four routing algorithms. First, we note that the message delivery delays of SPST and ER are equal and the least. Since ER floods the message along all links in space and time, there is certainly a copy of the message traveling along the shortest delay path identified by our SPST algorithm. Hence, a message reaches its destination after the same time in both these algorithms. Although both HPR and MFN use a greedy heuristic to minimize delay locally at a node, they incur longer delays globally than SPST and ER.

4) Resource usage: We study the cost of message forwarding in this experiment by observing the number of nodes visited by a message on its route from source to destination. This cost is representative of message transmissions as every intermediate node forwards the message. In case of ER, an intermediate node forwards the message more than once to reach its neighbors at different times.

Figure 10 is a cumulative distribution function of the number of nodes visited by a message as it is forwarded. Similar to the previous plot, this figure also plots values for only those cases when a message is successfully delivered by all the four routing algorithms. Although we have observed that ER has the same message delivery delay as SPST, a message visits many more nodes due to its flood-based forwarding making it an expensive choice for least delay. A message forwarded by SPST visits the least number of nodes as the routes are computed by examining the topology evolution up to the time horizon. HPR and MFN, on the other hand, use many more nodes to reach destination because of their message hand-off nature.

5) Delays with bursty traffic: Our earlier experiments compared the relative merit of each of four forwarding algorithms by examining their best case performance. In this experiment, we evaluate the delays incurred by messages under a realistic message traffic model that includes bursty arrivals. In Figure 11, we plot a distribution of the message delivery delays for our SPST routing algorithm and HPR routing algorithm. The x-axis corresponds to different values of delay and y-axis corresponds to the count of messages that have incurred this delay. We observe that, even under bursty traffic, our SPST routing algorithm does better than hot potato routing in terms of mean end-to-end delay. Figure 12 is a distribution of the delays incurred at a single node. These single hop delays measure the amount of time spent by a message at an intermediate node. The message may wait at a node because either there are other messages ahead of it in the node queue or a link to the appropriate next hop neighbor is not available yet. Since HPR algorithm uses a greedy heuristic, messages incur a lower single hop delay than those that use SPST algorithm.

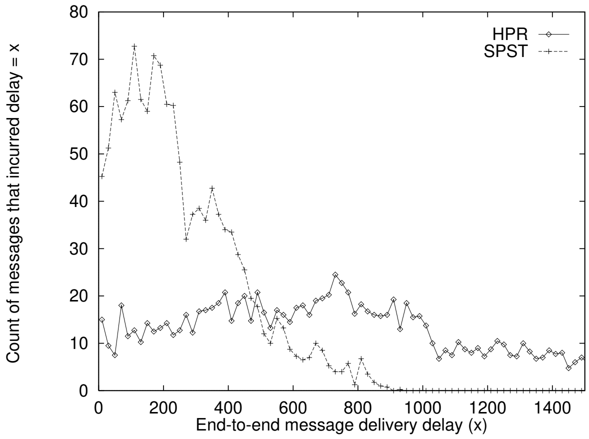

Fig. 11. Message delivery delay in bursty traffic model

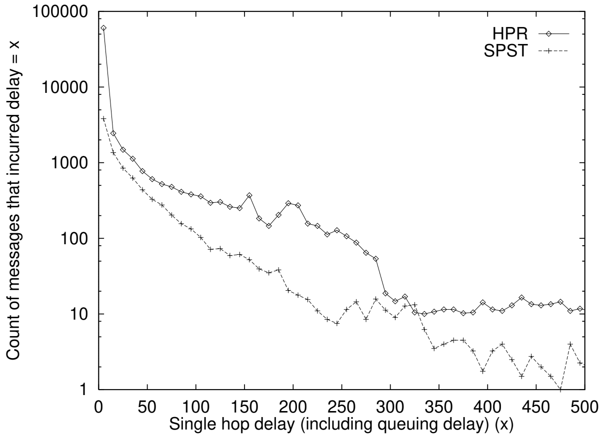

Fig. 12. Delay incurred at a single node

VII. Related Work

Space-time routing (STR) has been proposed recently for mobile ad hoc networks with a similar goal as ours to address the dynamic nature of the network topology [14]. However, STR considers a different parameter of the time dimension – the age of a routing entry to the destination. Our interest in time, by contrast, is how long does a message wait at a node to minimize overall delay to the destination. STR works with past history of routes, whereas, we look ahead to find new best routes for messages. FRESH, a protocol proposed by the same authors to highlight STR, suggests a new route discovery mechanism where route request propagation is guided by "encounter ages" [15]. A FRESH node forwards the route request to another node that has seen the destination more recently than the former node. The FRESH approach is based on a heuristic that a node with a recent encounter age is more likely to be in the vicinity of (or perhaps closer to) the destination than another node with an old encounter age. We believe that the STR approach (including FRESH) of using history of routes can blend with our proposal to identify routes that are not only stable, based on past history, but also provide best opportunity to deliver message (e.g., in the shortest delay) to the destination.

Our work on space-time routing draws on a precedence of several proposals that leverage predictability to improve routing. Su et al. apply mobility prediction to enhance unicast and multicast routing protocols [16]. They associate an "expiration time" to routes in the ad hoc network and the route selection algorithm prefers routes with longer expiration time. McDonald and Znati propose a framework for computing probabilities of a path and its availability over time in a mobile ad hoc network [17], [18]. They use this probabilistic model to develop an algorithm for clustering nodes such that the chances of existence of a route are high. They also build on this probabilistic model and develop another node proximity model that includes an adaptive learning strategy to estimate the chances that two nodes remain within a given distance threshold [19]. Likewise, Jiang et al. compute the probability that a particular link that is available now would last up to a given time in the future considering movements of nodes [20]. Measurement-based prediction schemes most often use global positioning system (GPS) information to track node location and movement and thereby compute coordinates and longevity of links [21].

Unlike flow-oriented forwarding approach used in many on-demand ad hoc routing protocols, message-oriented forwarding alleviates the overhead of route repair under rapid network dynamics. Forwarding message decisions, even for those messages belonging to the same flow, are independent in message-oriented scheme and hence are fairly robust to link changes in the network topology. Few recent proposals such as Epidemic routing [12] and Probabilistic routing [22] share this message oriented forwarding scheme with our proposal on space-time routing. Epidemic routing is a form of flooding where messages spread through the network by a simple pair-wise exchange protocol where a node forwards only those messages that its neighbor hasn't seen earlier. Probabilistic routing uses a history of node encounters to forward packets along a highly likely path. While these heuristic schemes are applicable in general, our space-time routing goes one step further, especially for those contexts where the network topology evolution, though rapid, is predictable. Mobile ad hoc networks with deterministic node motion fall into this category and hence benefit from such efficiency gains of our space-time routing framework.

Our inspiration for the time layers in the space-time graph comes from time-expanded graphs. Time-expanded graphs were originally proposed in the theory of network flows by Ford and Fulkerson [23]. The time-expanded graphs translate a problem of network flow over time to a classical "static" network flow problem and allow us to apply standard tools of graph theory. Derivations of time-expanded graphs are used to solve the quickest transshipment problem, dynamic multi-commodity flow problems [24], [25], transportation-related problems. We adapt the time-expanded graphs for our work on space-time routing as follows. First, we deviate from the continuous fluid flow models of network flow theory to discrete variable sized messages that are prevalent in real-world ad hoc networks. Second, although the flow of data spans time, the network graph remains constant in the basic version of time-expanded graphs. However, in our case of mobile ad hoc networks, the topology graph varies with time in addition to transmissions of messages that span time. Hence, a straight-forward and simple repetition of the graph for each time layer, as is done in the time-expanded graphs, is not directly applicable in our case. We construct each layer of our space-time graph following the time-varying link functions as explained in Section IV.

VIII. Conclusions

In this paper, we have considered routing messages in special networks where the topology, due to its disconnectivity, is not amenable to traditional mobile ad hoc routing protocols. In these networks, nodes move around explicitly carrying messages and creating paths over time. Our work explored this idea of routing in both space and time dimensions. Specifically, we took advantage of predictability in node motion available in some of these networks to construct space-time routing tables over a time horizon. These space-time routing tables consider time in addition to destination to determine the next hop node for forwarding. We derived a space-time graph model that gives a symbolic representation of paths taken by messages as they are routed in space and time. Using these space-time graphs, we presented an adaptation of the Floyd-Warshall algorithm to solve the minimum delay routing problem on these dynamic network topologies that are almost always disconnected at any instant in time. Our evaluation results support our observations that least delay paths in space and time using minimal resources can be identified with very high probability.

Routing in space and time has far reaching implications in efficient topology design of ad hoc networks. For example, one could model a network of nodes with directional antennas using space-time graphs. The sweeping rate of antenna, the exact instances in time when a particular direction is locked and the duration of lock can be varied to design a particular network topology that achieves efficient routing. Similarly, in networks with power adjustable transmission, one could design an efficient scheme to turn the transmitter on and off at appropriate time instances to save energy. We plan to explore this direction of efficient topology design with our background on routing in space and time.

Appendix

Space-Time Graph Construction

We describe below the construction of our layered space-time graph $G' = (V', E')$ starting from the set of nodes V and the link functions L_{ij} $(\forall i, j)$. We show how the temporal and spatial links are added to the time layers of vertices.

1) Time layers of vertices: For each node in V, we create multiple copies of the node corresponding to each τ-time interval up to $\mathbf{T}$. Hence, the vertex set V' of the space-time graph can be defined as

$$V' = V \times \{0, 1, \ldots, (\mathbf{T}-1)\},$$
$$\text{i.e., } V' = \{v_{it} | 1 \leq i \leq n; 0 \leq t < \mathbf{T}\} \quad (2)$$

where v_{it} denotes the pair (v_i, t), i.e., node v_i at time interval t.

2) Temporal links: For each node in V, we connect the vertices in V corresponding to successive intervals of time by directed links in order to allow for the possibility of messages being carried by the node. Hence, the set of temporal links E_0 of the space-time graph can be defined as

$$E_0 = \{(v_{it}, v_{i(t+1)}) | \ 1 \leq i \leq n; \ 0 \leq t < \mathbf{T}\}; \quad (3)$$

When the network topology is periodic, we can also add links that wrap around from time interval $(\mathbf{T}-1)$ to time interval 0. (i.e., $(v_{i(\mathbf{T}-1)}, v_{i0}) \in E_0$).

Since a node can potentially carry messages of arbitrary sizes, we color these temporal links with a wildcard color zero that denotes any color.

3) Spatial links: The spatial links of our space-time graph are derived from the link functions L_{ij} of pairs of nodes in the network. We define different sets E_c of spatial links based on message transmission delays. A spatial link that belongs to E_c has a message transmission delay of $c\tau$ time units and is colored c. These spatial links are suitable for traversal for messages of size m that fall into the message size bin c.

In order to obtain the set of c colored edges E_c, we consider all the vertex pairs $(v_{it}, v_{j(t+c)})$ such that the link function L_{ij} is active during the interval $(t\tau, (t+c)\tau]$ where τ is the constant time interval between the different layers of the space-time graph, i.e.,

$$E_c = \{(v_{it}, v_{j(t+c)}) | \ 1 \leq i, j \leq n; \ 0 \leq t < \mathbf{T};$$
$$L_{ij}(s) = 1 \, \forall s \in (t\tau, (t+c)\tau]\}; \quad (4)$$

Each edge in the set E_c is assumed to be colored c. The complete set of edges E' of the space time graph is obtained as the union of the temporal and all the different colored spatial linkes, i.e., $E' = \bigcup_{c=0}^{c_{max}} E_c$.

Integer Program Formulation

We formulate the shortest delay routing problem on the dynamic network topology of ad hoc network as a linear integer program in this section. This integer program formulation assumes the knowledge of message arrivals in addition to the dynamic topology evolution. A solution to this integer program provides a schedule of message transfers that minimizes the average delivery time of messages.

In this analysis, we define message types based only on the (source, destination, color) combinations. For a particular message type $p = (s, d, c)$, let $M(p, t)$ denote the number of messages with length corresponding to the color c that originate at the source node $s = src(p)$ during the time interval $((t-1), t]$ and are destined for the node $d = dst(p)$. For the purpose of integer program formulation of shortest delay routing, we assume that we are given the message traffic arrival functions $M(p, t)$ for all message types p up to a time $\mathbf{T}$ ($0 \leq t \leq \mathbf{T}$).

We start with our colored space-time graph defined in previous section. We enhance this model by defining a capacity function $C \ : \ E_c \bigcup E_0 \ \rightarrow \ Z^+$ on the spatial links as: $\forall i \neq j, \ C(v_{it}, v_{js})$ is equal to capacity of the link between v_i and v_j in number of messages that can be transmitted in a unit time. For temporal links of the type $(v_{it}, v_{i(t+1)})$, the capacity $C(v_{it}, v_{i(t+1)})$ is defined as the limit on the number of messages that can be stored at the node v_i.

Variables Let $\delta^+(v_{it})$ and $\delta^-(v_{it})$ respectively denote the set of incoming links and the set of the outgoinh links at vertex v_{it}. Let X_a^p be the number of messages of type p carried on a link a and let l_a be the delay associated with the link a. Let $D(p, t)$ be the number of messages of type p that arrive at the destination node $d = dst(p)$ in the interval $((t-1), t]$. For a sufficiently large value of $\mathbf{T}$, the number of messages produced in time period $\mathbf{T}$ is in practice almost equal to the number of messages that reach their destination in the same time period.

Objective Since our objective is to find the optimal schedule corresponding to minimum average delay over a time period $\mathbf{T}$ and the number of packets produced in this time period is constant, we minimize the total delays on each edge in the space time graph, i.e.,

Minimize

$$\frac{\sum_{p \in P} \sum_{a \in E_c \bigcup E_0} X_a^p l_a}{\sum_{t=0}^{T} \sum_{p \in P} M(p, t)} \quad (5)$$

subject to constraints [6], [7], [8], [9], [10], [11], and [12].

Node Conservation Constraints

$\forall i, t, p: \ 1 \leq i \leq n; \ 0 \leq t \leq T; \ p \in P$;

When $v_i \neq \text{src}(p); \ v_i \neq \text{dst}(p)$,

$$\sum_{a \in \delta^+(v_{it})} X_a^p = \sum_{a \in \delta^-(v_{it})} X_a^p. \quad (6)$$

When $v_i = \text{src}(p)$,

$$\sum_{a \in \delta^+(v_{it})} X_a^p + M(p, t) = \sum_{a \in \delta^-(v_{it})} X_a^p. \quad (7)$$

When $v_i = \text{dst}(p)$,

$$\sum_{a \in \delta^-(v_{it})} X_a^p = 0, \quad (8)$$

$$\sum_{a \in \delta^+(v_{it})} X_a^p = D(p, t). \quad (9)$$

Overall Packet Conservation Constraints

$\forall p : p \in P$;

$$\sum_{t=0}^{T} M(p, t) = \sum_{t=0}^{T} D(p, t). \quad (10)$$

Capacity Constraints

$\forall a: \ a \in E_c \bigcup E_0 (c = color(p))$

$$\sum_{p \in P} X_a^p \leq C(a). \quad (11)$$

Non-negativity Constraints

$\forall a, p: \ a \in E_c \bigcup E_0(c = color(p)); \ p \in P;$

$$X_a^p \geq 0. \tag{12}$$

REFERENCES

[1] W. Zhao, M. Ammar, and E. Zegura, "A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks," in *ACM MobicHoc 2004 (to appear)*, 2004.

[2] A. Pentland, R. Fletcher, and A. A. Hasson, "A Road to Universal Broadband Connectivity," in *Second International Conference on Open Collaborative Design for Sustainable Innovation; Development by Design (dyd02)*, December 2002.

[3] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks," in *IEEE Workshop on Sensor Network Protocols and Applications (SNPA)*, May 2003.

[4] A. Beaufour, M. Leopold, and P. Bonnet, "Smart-Tag Based Data Dissemination," in *First ACM workshop on Wireless Sensor Network and Applications*, October 2002.

[5] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet," in *The Tenth International Conference on Architectural Support for Programming Languages and Operating Sytems (ASPLOS-X 2002)*, October 2002.

[6] C. Shen, G. Borkar, S. Rajagopalan, and C. Jaikaeo, "Interrogation-based relay routing for ad hoc satellite networks," in *IEEE Globecom*, Taipei, Taiwan, November 17-21 2002.

[7] S. Burleigh, V. Cerf, R. Durst, K. Fall, A. Hooke, K. Scott, and H. Weiss, "The Interplanetary Internet: A Communications Infrastructure for Mars Exploration," in 53^{rd} *International Astronautical Congress, The World Space Congress*, 2002.

[8] T. Small and Z. 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*, Jun. 2003.

[9] R. W. Floyd, "Algorithm 97 (SHORTEST PATH)," *Communications of the ACM*, vol. 5, no. 6, 1962.

[10] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, U.S.A., 1990.

[11] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[12] A. Vahdat and D. Becker, "Epidemic Routing for Partially Connected Ad Hoc Networks," Tech. Rep. CS-200006, Duke University, 2000.

[13] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, no. 5, pp. 483–502, 2002.

[14] H. Dubois-Ferriere, M. Grossglauser, and M. Vetterli, "Space-Time Routing in Ad Hoc Networks," in *Second Intl. Conference on Ad Hoc Networks and Wireless*, 2003.

[15] H. Dubois-Ferriere, M. Grossglauser, and M. Vetterli, "Age Matters: Efficient Route Discovery for Mobile Ad Hoc Networks UsingEncounter Ages," in *Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2003.

[16] W. Su, S-J. Lee, and M. Gerla, "Mobility Prediction and Routing in Ad Hoc Wireless Networks," *International Journal of Network Management*, 2000.

[17] A. B. McDonald and T. F. Znati, "A Path Availability Model for Wireless Ad Hoc Networks," in *IEEE WCNC*, Sep. 1999, pp. 35–40.

[18] A. B. McDonald and T. F. Znati, "A Mobility-Based Framework for Adaptive Clustering in Wireless Ad Hoc Networks," *IEEE JSAC*, vol. 17, no. 8, pp. 1466–1487, 1999.

[19] A. B. McDonald and T. Znati, "Predicting Node Proximity in Ad Hoc Networks: A Least Overhead Adaptive Model for Selecting Stable Routes," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2000.

[20] S. M. Jiang, D. J. He, and J. Q. Rao, "A Prediction-Based Link Availability Estimation for Mobile Ad Hoc Networks," in *IEEE Infocom*, Apr. 2001.

[21] W. Su, S-J. Lee, and M. Gerla, "Mobility Prediction in Wireless Networks," in *IEEE Military Communications Conference*, Oct. 2000.

[22] A. Lindgren, A. Doria, and O. Schelen, "Poster: Probabilistic Routing in Intermittently Connected Networks," in *Fourth ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003)*, 2003.

[23] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, 1962.

[24] L. Fleischer and M. Skutella, "The Quickest Multicommodity Flow Problem," in *Integer Programming and Combinatorial Optimization*. 2002, vol. 2337, pp. 36–53, Springer-Verlag Lecture Notes in Computer Science.

[25] B. Klinz and G. J. Woeginger, "Minimum Cost Dynamic Flows: the Series-Parallel Case," in *Integer Programming and Combinatorial Optimization*. 1995, vol. 920, pp. 329–343, Springer-Verlag Lecture Notes in Computer Science.