Mingle: A Percolative Routing Framework for Mobile Delay Tolerant Networks

Deepak Bastakoty Tong Zhou Romit Roy Choudhury
Dept. of Electrical and Computer Engineering,
Duke University

Abstract

Delay tolerant networks (DTNs) comprise of disconnected groups of mobile nodes that mingle with each other in unpredictable ways. This paper explores the possibility of enabling communication in delay tolerant networks by exploiting the density of wireless nodes and the diversity in their mobility. As a first step toward this goal, we develop a routing framework for connecting two nodes, that may not be reachable through wireless transmissions alone. We show how messages can be transported by opportunistically exploiting physical node mobility or wireless transmissions, whichever proves to be available. Our scheme is based on a distributed location service wherein nodes periodically gossip about the whereabouts of others nodes. The knowledge of whereabouts percolate quickly and serve as useful hints for routing data packets from a source to its destination. While the hints may not be accurate, we show how the property of *spatial locality* in DTNs can adequately compensate for the inaccuracies. Simulation results demonstrate that our framework can achieve a reasonably good tradeoff between delivery latency and control overhead. Encouraged by these results, we are currently implementing our scheme, Mingle, on a small-scale prototype testbed.

I. Introduction

The wide-scale adoption of wireless devices may enable a new platform for peer-to-peer opportunistic networking. One potential framework, termed *delay tolerant networks* (DTNs), is receiving increasing research attention [1–4]. Delay tolerant networks comprise of mobile wireless devices that are not necessarily connected to each other. Messages progress from a source to its destination by opportunistically exploiting wireless connectivity as well as physical node mobility. Intermediate nodes receive batches of packets (called *bundles* [1]), store and carry them in their local memory, and forward them either to better relays or to their final destinations. Of course, relying on physical node mobility for message transport can considerably increase the latency of communication[1]. However, the increasing density of wireless devices, and the inherent diversity in user mobility may serve to bound this latency to reasonable limits. Several applications, such as mobile sensor networks, may be tolerant to such latencies. In other classes of civilian applications, the utility of ubiquitous connectivity may override the inconvenience for higher latency. Delay tolerant architectures may empower all these classes of applications.

Need for DTNs: One may question the need for DTNs given that ad hoc networks are envisioned to address similar high level goals. In fact, ad hoc networks do not even sacrifice latency to achieve the kind of services that DTNs may support. However, we argue that by sacrificing latency, DTNs may resolve several issues that otherwise impair ad hoc networks from becoming a practical, scalable solution. In particular, node mobility in ad hoc networks cause frequent link failures, *critically affecting the connection-oriented nature of the system*. Coping with node mobility has proved to be difficult, especially when the network is likely to partition and reconnect

[1] In particular, when mobility is unpredictable, a message may get carried further away from the destination, further increasing the delay in delivery.

at unpredictable time instants. In contrast, the possibility of exploiting mobility as a communication opportunity (and not a peril) can make the problem manageable in DTNs. While latency may increase, the tradeoff may prove to be worthwhile in certain conditions.

Even when network partition is not a concern, ad hoc networks face a difficult task of maintaining end to end multihop connections over an inherently unreliable wireless channel [5]. The difficulty is pronounced when nodes move, often destabilizing the overall network. DTNs, however, can handle this issue by forwarding bundles in a non-pipelined manner (i.e., over one link at a time). Link by link communication is well understood over the wireless media, and hence, DTNs may offer the requisite reliability although at the cost of higher delivery latency.

Challenges: Translating the intuition of delay tolerant networks into an usable system is a non-trivial research problem. One important challenge pertains to locating and routing to a desired target that may not be reachable over wireless transmissions alone. Clearly, traditional route-discovery schemes (such as those based on flooding) will not work – *in a DTN, a flooded packet will only reach nodes that are within the originator's partition*. Moreover, even if a route is discovered, the route may quickly become invalid, triggering a new route discovery. Since repeated route discoveries can be prohibitively expensive, alternate solutions appear imperative for locating and routing in DTNs.

Design Choices: We consider some design choices toward locating a node in delay tolerant networks. (1) A trivial extension of the traditional approach could be to periodically initiate a network flood until a route is discovered between the source and its destination. Observe that a route will be discovered only when the source luckily initiates a flood while there exists an end to end wireless route between the source and its destination. In all other cases, the flood will "perish" within the source's cluster. Since routes can change in the time scale of end-to-end communication, the periodic flooding operation may have to be repeated for each bundle. (2) A flood could be initiated once, and each node could continue to persistently rebroadcast the message. Observe that nodes that move away from the flood-originator, and mingle into different clusters, will deliver the message to the members of the cluster. Clearly, the destination is guaranteed to receive the bundle. While this design can achieve the optimal latency for communication, it suffers from an extremely high overhead. Moreover, nodes will not know when the destination has been found, hence, will continue to rebroadcast the packets for a (pre-specified) maximum number of times. For battery operated wireless devices, this approach will clearly be unscalable. (3) To reduce the overhead of many redundant broadcasts, epidemic routing [6] requires nodes to transmit packets probabilistically. Under carefully chosen probabilities, it might be feasible to transport bundles to a desired destination, at relatively lower overheads. However, duplicating large-sized bundles can incur high energy and storage costs.

Our Broad Approach: Based on the above discussions, we argue that locating a node, every time a message needs to be sent, may not be scalable. Instead, we propose a proactive location service that can serve as the basis for geographic routing between any source-destination pair. To achieve this, we assume that nodes are equipped with GPS-type location information. Each node records its encounters with other nodes, while they move around in the network. The details of these encounters are cached at each node, and the caches are exchanged periodically with other nodes that are encountered later. Due to these proactive exchanges, any node in the network is likely to be partially aware of any other node's whereabouts. This is a result of an

interesting phase transition property of DTNs, wherein, a specific information can quickly reach a large fraction of others nodes, in a reasonably short duration. This property is shown in Figure 1. Capitalizing on this property, we design our communication framework – Mingle – for delay tolerant networks.

The partial awareness of a destination's location – called *hint* – can be used for geographically forwarding messages/bundles toward the destination. For instance, if node X receives a message destined for node D, and if node X had encountered D at location (x, y), and time t_D^X time units in the past, then node X can geographically forward the message toward coordinates (x, y). The value of t_D^X can be included in the message as a measure of confidence. Downstream nodes that receive this message, continue to forward it toward location (x, y), unless they had encountered D later than t_D^X, in which case, they redirect the message toward the location stored in their cache. As we show later, the trajectory of a message can quickly converge to its destination because network nodes are capable of providing progressively good hints. Percolating hints, and using them for communication, forms the core of our protocol.

Clearly, several issues arise. (1) The available hint may be stale, and a message may be di-

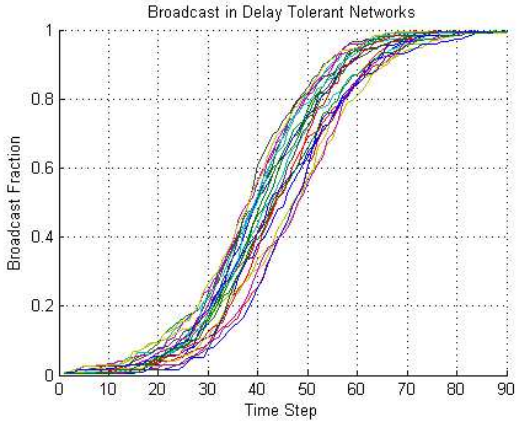

Fig. 1. The Y axis shows the fraction of nodes that have received a broadcast packet, while the X axis shows the time steps since the origination of the packet from the sender. A large fraction of the nodes receive the packet within 60 time-steps, when the probability of gossiping is 0.1. The network exhibits a phase transition property in percolating a message.

rected toward an inaccurate location. (2) Exchanging encounters with other nodes in the network has overheads associated to it. (3) The paths traced by the bundles may get longer, and may not quickly converge if the destination moves too fast. These and other issues will need to be resolved in order to construct a routing framework for DTNs. This paper will focus on these challenges.

Organization: The rest of the paper is organized as follows. Section II discusses the related work on routing in delay tolerant networks. Section III describes the assumptions and intuition behind our proposed scheme, while Section IV presents the scheme in detail. In Section V we evaluate Mingle through simulations. We discuss several issues in Section VI, and conclude with a brief summary in Section VII.

II. Related Work

Routing in disconnected ad hoc networks was initially considered in [7, 8]. Authors presented mechanisms to actively alter the trajectory of messages when forwarding nodes faced with a link disconnection. These approaches mostly assumed that the network is dominantly connected, and disconnections happened occasionally. In [9], authors considered networks that are dominantly disconnected, and thereby lack any trajectory while routing packets. In such networks, [9] proposed an epidemic routing scheme in which mobile nodes exchange messages between themselves, if they have not already received the same message. The authors show that at the expense of high latency and storage overhead, packet delivery can be achieved with high reliability. Several papers followed that exploited the opportunity to forward packets to nodes that can act as message ferries, and eventually deliver it to the destination [10].

Routing in delay tolerant networks was first investigated in [1], where the authors assumed every node in the network can precisely predict node mobility, and hence, can apply Dijkstra's algorithm to obtain minimal latency routes. Several approaches were later proposed that relaxed the above assumption. In [2], the authors propose to optimize routes based on the knowledge of each node's probability of meeting the destination. In [3], the authors further simplify the prediction of node mobility by assuming the destination to be a static node. Thus, any node is only required to predict the probability that the destination will be at a specific location. Based on the method in [3], authors in [11] propose a routing protocol in a realistic bus-based environment. As the schedule of each bus is pre-determined, predicting node mobility proves to be feasible. In an independent work, [4] investigates routing in car-based environments. While cars meet each other at unpredictable times, they are assumed to always have other cars as their neighbors. The authors propose a routing scheme over such a network environment.

When large-sized data packets needs to be transmitted over a DTN, it may be cost-effective to perform a route discovery before sending out the packets. This is in view of the large overheads associated to transmitting redundant data packets (where redundancy is necessary for reliability and lowering delivery latency). A reactive route discovery typically entails a broadcast of query packets. While simple in connected networks (such as ad hoc networks), broadcast is a non-trivial problem in DTNs. Flooding and its appropriate variants [12, 13], are not applicable in dominantly disconnected networks. In view of this, authors in [14–18] have proposed efficient broadcast schemes. In [14], authors have studied the impact of a "pull" mechanism wherein users request data from their mobile neighbors. In applications such as streaming media, they show that uninterrupted dissemination can be achieved with high reliability, especially in dense networks. In [15], we considered the problem of resource discovery in DTNs, and proposed a scalable scheme that identifies the location of a target node.

The paper proposes a gossiping mechanism [19, 20] in which certain nodes gossip about the location of other nodes. Gossiping has been regarded as an efficient method for information dissemination in wireless ad hoc networks. Later on, if a particular user intends to discover a target node, it uses the hints from the gossip construct multiple routes which are expected to converge toward the target. Similar ideas have been proposed in [17] where nodes exploit the memory of encounters to percolate information in intermittently connected nodes.
We show that physical mobility can partially emulate the impact of gossip, leading to the possibility of lesser communication overhead. Based on this, we propose a protocol where information is transported via wireless transmission as well as physical mobility. We present our ideas in the

form of a location estimation service in wireless ad hoc networks.

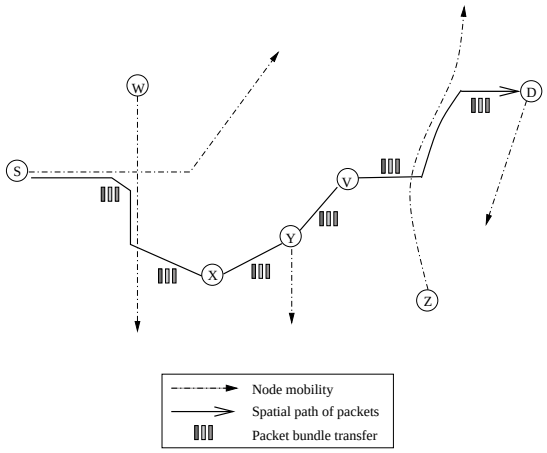

Fig. 2. The functional view of a DTN. Nodes pause and move in unpredictable ways. Bundles progress toward the destination through wireless transfers and through physical node mobility.

III. Assumptions and Intuition

We present the assumptions underlying Mingle, followed by the intuition behind the operation of the scheme.

A. Assumptions

A functional view of the DTN is shown in Figure 2. As generalized in the figure, nodes relay messages on behalf of other nodes. When a downstream relay is unsuitable or unavailable, the bundle is carried physically (via user mobility) and delivered to a better relay, or to the final destination. In this paper, we assume that nodes are equipped with GPS capabilities, and have $O(n)$ storage capacity (where n is the number of nodes in the network). When nodes arrive in mutual communication range, they exchange "hello" packets for purposes of neighbor discovery. We also assume that the nodes remain within range for a duration that is sufficient for pending message transfers. While this may not necessarily hold, we do not address the problem in our proposed scheme. In this paper, we focus on finding any one end-to-end connection between a given node pair (similar to a ping), in a dominantly disconnected delay tolerant network.

B. The Intuition

The whereabouts of a large number of nodes can be difficult to keep track of, especially when nodes in the network are not necessarily connected. Moreover, the diverse and whimsical movement of nodes make mobility prediction difficult. To cope with this, we capitalize on the very node density and mobility diversity of the network – we attempt to translate the perils into an opportunity. Specifically, when many mobile nodes mingle and move in diverse directions, we exploit them as an opportunistic platform for spreading information. We require each node to cache the whereabouts of other nodes in the network, based on their earlier encounters. When these nodes are made to exchange their cached information, the phase transition property of the network gets triggered – the cache contents spread extremely fast, as evident from Figure 3. Any source may now be able to (approximately) locate its destination, say D, based on the information available in its own cache. Of course, given that destination D may also be moving, D's location information becomes increasingly noisy with elapsing time. We exploit *spatial locality* in the

network to cope with this issue. By spatial locality, we mean that the location of a node, D, is likely to be known more accurately by other nodes that are spatially near D. If a message is directed toward an approximate location, the trajectory of the message gets progressively corrected as it moves closer to the destination. As we show later, the effect of progressive correction can be beneficial in DTNs.

An important question arises at this point. Since the mobility of nodes cannot be controlled,

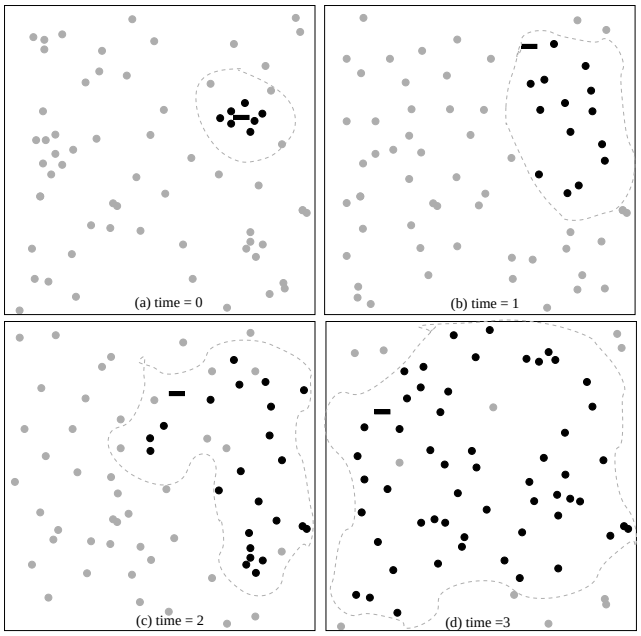

Fig. 3. We simulated a network in which the location of the rectangular node was initially known only to the black nodes (time 0). As the black nodes moved using random way-point mobility, they exchanged their caches with new nodes they met. Every node that came to learn of the rectangular node was colored black. The diffusion of information is shown to be dramatically fast.

the behavior of information percolation will essentially be uncontrolled too. To illustrate this, consider a worst case scenario in which all nodes in a network decide to stop moving. In such a scenario, physical transportation of an information would cease, and the information diffusion would have to rely completely on wireless transmissions. This indicates that the blend of wireless communication and physical transportation depends on the mobility patterns of the nodes. Where nodes move faster, physical mobility can carry information, reducing the consumption of scarce network resources (such as bandwidth, energy, etc.). However, note that when nodes slow down, the overhead of wireless transmission does not increase heavily. This is because, as nodes move slower, their location becomes less uncertain, and thus, a desired level of location estimation can be achieved with fewer message exchanges. In other words, we indicate the possibility of a self-sufficient network, that copes with high and low mobility, at reasonably low bandwidth overheads. The ideas in this paper aim to realize this possibility.

IV. MINGLE: A PERCOLATIVE ROUTING PROTOCOL

This section presents our ideas on Percolative Routing. For the purposes of easier explanation, we describe how a sender node, S, transmits a single message to a destination node, D. The transmission of multiple messages can be achieved through repeatedly invoking the proposed scheme.

We evaluate our scheme in Section V using metrics of end to end latency and transmission overhead.

Caching Encounters:
We assume that as nodes move into the communication ranges of other nodes, beacons are exchanged between them[2]. The GPS location of the nodes are included in the beacons. As a result, nodes learn about their mutual locations, and update their local caches with this information – we call this operation an *encounter*. As an example, if nodes A and B encounter each other at time t_i, while they are at locations (x_A, y_A) and (x_B, y_B) respectively, then node A updates its cache with the following tuple.

$$\{node\ B,\ t_i,\ (x_B, y_B),\ IsNeighbor:\ TRUE\}$$

Node B updates its own cache similarly. While nodes A and B remain in each other's communication range, they periodically update the same entry with newer (more recent) time-stamps and locations. The updation terminates automatically once the nodes have moved out of their mutual communication ranges.

Once A and B are not immediate neighbors, they set $IsNeighbor$ to $FALSE$, indicating that the encounter occurred in the past. The tuple in node A's cache is now of the form

$$\{node\ B,\ t_j,\ (x_B^{'}, y_B^{'}),\ IsNeighbor:\ FALSE\}$$

where $t_j \geq t_i$. Now, as these nodes encounter new nodes in the network, they mutually gossip their memory of encounters. For example, if node A meets node P at a later point of time, say t_k, then node A informs P about all its encounters in the recent past, including that with node B. If P has not met B any later than t_j, then P updates its own cache with the memory of A's encounter with B. However, if P had happened to meet B later than t_j, then node A updates its own cache with the memory of P's encounter with B. This exchange of more recent encounters happens not just for B, but for several other nodes that are in A's and P's cache memory. Thereafter, assuming that P learned about B's recent location from A, node P pretends that P met B at time t_j in the past, and continues to gossip this information with subsequently encountered nodes. Nodes that meet P similarly get informed about B, and further percolate the gossip. The choice of gossip probability dictates the tradeoff between dissemination latency and control overhead. We discuss this tradeoff later in this paper.

Routing based on cached locations:
Consider a source node, S, that intends to communicate packets/bundles to a destination, D. Assuming that node caches have warmed up, node S is likely to find the location of D in its own encounter cache[3]. Let us denote this location as L_D^S, and the timestamp associated to this location as t_i. Of course, L_D^S is likely to be stale. Nevertheless, this stale information may guide the bundle toward the approximate location of D, obviating a network-wide flood. Therefore, node S geographically forwards the bundle toward location L_D^S, i.e., node S finds a neighbor that is geographically closer to D than S itself. The bundle also includes the timestamp, t_i, indicating that node D was at location L_D^S at time t_i in the past. As the bundle (geographically) progresses toward the approximate location, relay nodes may correct the bundle's trajectory. Specifically, a

[2]Several protocols require such beacon exchanges, including neighbor discovery protocols, time synchronization protocols, etc.

[3]If the location is unavailable, node S randomly picks a location, and assigns it to D. The timestamp for this location is assigned to be $-\infty$.

relay node, R, that has a memory of its encounter with D, (say from time t_j, where $t_j \geq t_i$), guides the bundle toward its own cached location, L_D^R. The values of L_D^R and t_j are again included in the message, so that downstream relays do not redirect the packet unless they have encountered D later than t_j. Thus, at this point, the bundle header has the following form.

$$\{Src : S,\ SrcLocation : L_S,\ Dest : D,\ L_D^R,\ t_j,\ Hops\ h\}$$

As the message closes in toward the destination, it gains more and more accurate information about the destination's location. This is because of *spatial locality* – nodes that are located near D, are likely to have encountered D more recently, and are likely to posses better estimates of D's location. These nodes iteratively correct the bundle's trajectory, allowing the bundle to converge toward D. Once the bundle reaches the destination, node D can reply back with an acknowledgment, using the same geographic forwarding mechanism. The acknowledgment will include the destination node as S, with the destination's location and timestamp as L_S, and t_i, respectively.

Reducing route length:
Routing one message toward the destination may face some issues. First, a bundle may not progress toward the destination if an intermediate node, X, does not find a downstream relay that is geographically closer to the destination. We call this a *routing hole*. When stuck at a routing hole, X will continue to carry the message in its own cache, until it finds a suitable downstream candidate. While carrying the message, X may move in a direction that is away from the destination of the message. This can increase the latency of communication. Second, it is possible that the information about the destination's location is significantly *stale*. This happens especially when the destination has moved quickly to a location that is not well populated, and hence, may have encountered only a few nodes. In such a scenario, many nodes in the network still have the stale information, while the new information is beginning to percolate in the network (i.e., prior to the phase transition time window). In this scenario too, the latency of communication can increase since the bundle will move toward an incorrect direction, and its trajectory will not get adequately corrected.

We improve the latency of communication by drawing on redundancy. Node S sends K copies of the same message, one toward the location that it thinks is correct. The remaining $(K-1)$ copies are are directed as follows. With reference to the destination's location, S chooses $(K-1)$ virtual locations, such that the lines joining node S with each of these locations are angularly separated by $\frac{2\pi}{K}$ radians. Let us denote each of these virtual locations as $(x_i,\ y_i),\ i \in [1, K-1]$. Now, S directs $(K-1)$ copies of the message, one toward each of these directions. These copies are also forwarded geographically, and when they reach the vicinity of their respective destinations, $(x_i,\ y_i)$, they start moving toward the location of the actual destination, D. The diagram in Figure 4(a) illustrates the idea with $K = 2$. We expect that by initiating the search for D from K different regions in the network, it might be feasible to address the problem of *holes* and *stale* information.

A. Protocol Details

Choice of K: Observe that all the K copies may eventually reach the destination, at different points in time. While this is redundant, the minimum latency among all the K copies might be lower than sending one copy along the line joining S and D. Clearly, this is a tradeoff. One way to handle this *overhead-latency tradeoff* could be through more informed choice of K. For example, a source node S, that already has a hint of D's location, can choose a smaller value of

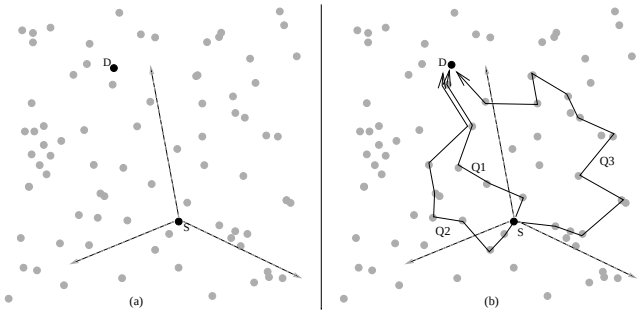

Fig. 4. (a)Choosing 3 directions for geographically forwarding the same query. (b) Tracing the path of 3 queries while they converge toward destination D.

K, depending on how stale the hint is. One way of choosing this value of K can be

$$K = min(\left\lceil \frac{(t_{current} - t_{last-encounter})}{C} \right\rceil, K_{max})$$

where C and K_{max} are appropriately chosen constants. Thus, a node that knows of a recent encounter with D, spawns very few copies of the message to propagate into the network. Moreover, these messages may not be transmitted in random directions, but can be guided to envelope the expected location of D. The expected location of D is the region over which D is likely to be, given that D was encountered at a location (x_D, y_D), at $t_{last-encounter}$. The expected region can be calculated as a circular region centered at (x_D, y_D), and with a radius of $v_{max}(t_{current} - t_{last-encounter})$, where v_{max} is the maximum speed with which a node might move. Guiding few messages toward the expected region can reduce the latency of communication, while limiting message redundancy.

Choice of Gossip Interval: The frequency with which nodes exchange their cached information leads to a tradeoff. Shorter exchange intervals lead to quicker percolation, but at the expense of higher overhead. A suitable value of exchange interval is one that minimally separates two consecutive gossip transmissions in time, such that these two transmissions are received by distinct sets of nodes. This implies that a node that has its neighbor-set continuously changing (i.e., more mobile), must gossip more frequently, as opposed to another node, whose neighbor-set hardly changes with time (almost stationary). An approximate choice of this gossip-interval can be $\frac{2R}{2v}$, where R is the communication range and $2v$ is the relative velocity of nodes that intends to gossip. Our justification is that, on an average, a node is likely to change its neighbor set every $\frac{2R}{2v}$ time units. With transmission ranges of $250m$, and average speeds of $25m/s$, a node should gossip around once per 10 seconds. In other words, a gossip probability of 0.1 is likely to facilitate good percolation. We will quantify this choice through simulations in Section V.

V. Performance Evaluation

In this section, we evaluate the performance of Mingle, and compare its performance to two other protocols, namely *Optimal Connection Protocol* and *Periodic Broadcast Protocol*.

Optimal Connection (OC): This scheme is used to determine the least latency that might be achievable in a given network environment. In OC, every node that receives a broadcast message persistently rebroadcasts it with very high frequency (i.e., 100 times in every time unit). The

persistent broadcast continues until the destination has received the first message, at which point the end to end latency is recorded. Observe that by persistent broadcasting, OC explores every possible path in the network, and thereby can determine the lower bound on latency[4]. Clearly, the overhead associated with Optimal Connection will be extremely high, making it an impractical protocol. Hence, we will use OC only to characterize the least communication latency in a given mobile network environment, and use it as a benchmark to evaluate our protocol.

Periodic Broadcast (PB): This scheme is a simple variant of flooding, adapted to suit the disconnected nature of DTNs. In PB, a sender initiates a broadcast once every T_{PB} time units. For every broadcast, nodes that receive the broadcast message forward it *only once* after waiting for a random jitter. Since a broadcast is likely to perish within the sender's cluster, the sender repeats it every T_{PB} time units until the broadcast has reached the destination (we record the end to end latency when the broadcast first reaches the destination). Clearly, PB reduces the overhead of communication since transmissions are less frequent than OC. However, it will incur higher latency than OC, especially when the network is dominantly partitioned into several clusters. We compare Mingle against these protocols, using metrics of connection latency and overhead.
We have designed a network simulator in Java for evaluating and comparing the performance of DTN routing protocols. Our simulator models the basic PHY and MAC layer characteristics, but omits details such as channel fading, precise interference calculations, etc. We believe that such inaccuracies in simulation may not severely impact the evaluation of routing protocols in DTNs. Hence, we focus on the connection dynamics of the network, and the interactions between nodes while they move and encounter each other.

In a simulated square space of width $2,500m$, we implemented two node mobility models as described below:

1. *Random Waypoint Model*: In this model, each node starts from a random location. The node then chooses a destination point, moves to the destination at a random speed between realistic minimum and maximum values, and pauses for a bounded random duration. Subsequently, the node chooses a new destination and repeats the above process. This continues till the end of the simulation[5].

2. *Cluster-based model*: In this model, several regions[6] are designated as meeting points of nodes. During initialization, each node picks a random starting location. While the simulation is running, each node selects a destination based on a parameter called *meeting bias*, denoted as $\beta \in [0, 1]$. Parameter β represents the probability that the next chosen destination is within any one of the meeting regions. With probability $[1 - \beta]$, a random location in the simulated space is chosen as the next destination. The node moves towards its destination using a speed chosen randomly between realistic minimum and maximum values. Upon reaching the destination, the node pauses for a random duration, and repeats the process. The model is inspired from the observation that human activities are often spatially collocated (conferences, meetings, cafes, classrooms, etc.) This model may partially reflect such tendencies toward node clustering.

[4]Note that the least latency path need not necessarily comprise only of wireless links. Paths that are partially comprised of physical mobility will also be singled out by OC.

[5]We have incorporated the necessary modifications proposed in [?] to ensure that the network reaches a steady state.

[6]A region is defined by a circle around a specific location.

Parameter	Value
Number of Nodes	10 – 400
Communication Range	250m
Radius of Meeting Region	250m
Number of Clusters	5
Speed of Nodes	20m/s – 50m/s
Pause Durations	10 dt – 120 dt
Time Steps	2s
Meeting Bias	0 – 1
Video	www.duke.edu/˜ db29/vdo.html

TABLE I
PARAMETERS USED IN SIMULATION

Table I shows all the experimental parameters used in the simulation. All the results presented in this paper are derived from an average of 50 simulation runs, each with a different seed. Some videos of the simulation may be accessed at the url provided in Table I.

*A. **Connection Latency***

In this subsection, we compare the performance of Mingle, Optimal Connection (OC), and Periodic Broadcast (PB) in terms of end to end connection latency. We examine how latency is affected by varying node density, meeting bias, and pause time. Figure 5 shows the connection latency as a function of node density for random waypoint (RW) mobility. Figure 6 shows the same for cluster based mobility. For these graphs, pause time is uniformly distributed in the interval $[10, 40]$ time units (each time unit corresponding to 2 seconds). For the cluster-based model, *meeting bias* is assigned as $\beta = 0.7$.

As expected from the figures, Optimal Connection (OC) has the minimum latency for all cases. Notice that latency reduces with higher node density since wireless connectivity becomes increasingly more available, and the message relies less on physical mobility. Periodic Broadcast (PB) has the highest latency because the broadcast has to repeat multiple times before it finds a complete end to end connection. Mingle attains latencies close to the optimal at moderate and high node densities. For random waypoint, when the number of nodes reaches above 100, we see that Periodic Broadcast and Mingle both approach the ideal latency. This is expected since our simulation space is 10 transmission ranges wide and therefore 100 nodes provide almost complete wireless connectivity.
At lower densities, the performance of Mingle degrades, but remains substantially better than PB. For cluster-based mobility, Mingle remains close to OC even at lower densities, and outperforms PB even at larger densities. This is because Mingle effectively utilizes the fewer nodes that move between clusters. DTNs that demonstrate high clustering tendencies with some randomly moving nodes may be particularly amenable to Mingle. Human networks may be one instance of such DTNs.

Figure 7 shows the individual connection latencies for 200 different simulation runs. The latency of Mingle is sorted in increasing order and plotted together with the corresponding PB

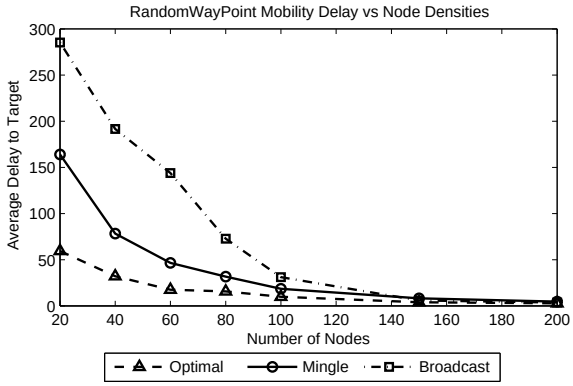

Fig. 5. Connection latency for random waypoint with increasing node density.

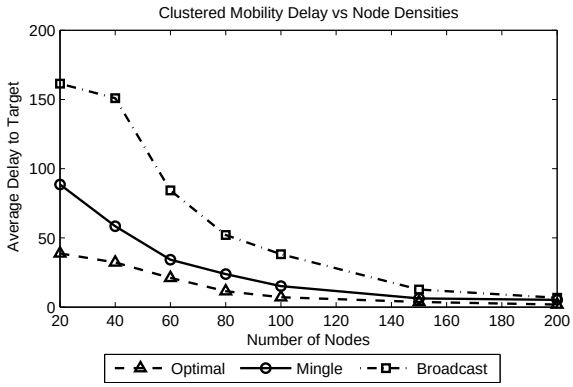

Fig. 6. Latency for cluster-based mobility with increasing node density.

latency. Observe that in some cases, PB performs better than Mingle. This occurs when the destination is located within the source's cluster, and thus, a broadcast can wirelessly reach the destination. However, Mingle may not necessarily utilize the wireless connection because *this connection may not be along the geographic straight line joining the source and destination.* As a result, Mingle may partially rely on physical mobility, which increases the delay of message delivery. In general however, PB incurs substantially higher latency (and variance) than Mingle.

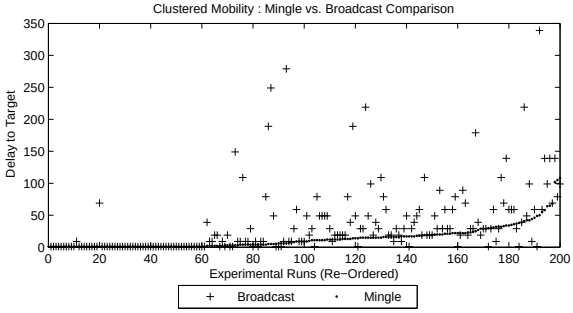

Fig. 7. Latency comparison for individual simulation runs with 100 nodes.

Figures 8 and 9 show the variation of connection latency with increasing pause time. The pause times are randomly selected from an interval of $[0.5, 1] \times T_{max}$ time units. T_{max} is plot-

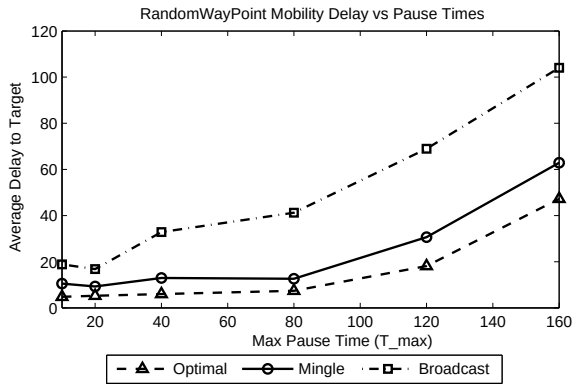

Fig. 8. Latency for random waypoint mobility model.

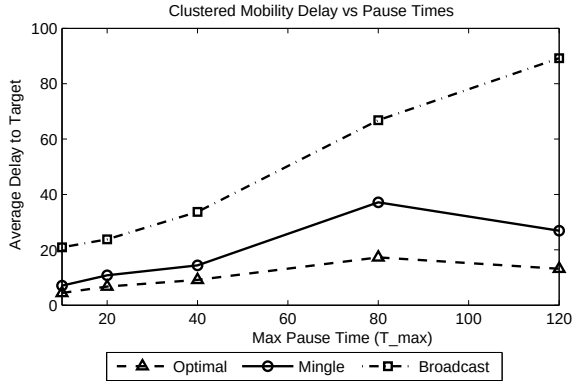

Fig. 9. Latency for cluster-based mobility model.

ted in the x-axis. The graphs show an increase in latency with increasing pause time, across all the evaluated protocols. This happens because increasing pause time dampens the rate at which nodes can mingle in the network. As a result, fewer opportunities arise for forwarding a message towards its destination. This causes a message to remain in the cache of a particular node for longer durations of time, leading to higher latencies in message delivery.

We also notice that with larger pause times, the performance of our scheme degrades more gracefully than that of Periodic Broadcast. This is because PB does not take advantage of the store-and-carry mechanism; when a group of nodes do not encounter new relay nodes, the message perishes within that group. With Mingle and OC, since every intermediate node stores, carries, and forwards the data towards the destination, a reasonable latency can be achieved even with fairly high pause times. This highlights Mingle's ability to exploit mobility when necessary.

Figure 10 evaluates Mingle for varying meeting bias, β. Observe that as β increases, the latencies increase slightly due to increased clustering and decreased mingling among nodes. However, some nodes continue to carry information between the clusters thereby enabling our protocol to perform reasonably well. Notably, the performance of PB degrades significantly since no end-to-end connection can be found when the network is predominantly clustered. This result perhaps indicates that reactive routing (based on on-demand route discovery) may not be suitable for disruption/disconnection prone networks. The route discovery process may incur significant delay, while the discovered route may not persist for the duration of the data session. A proactive

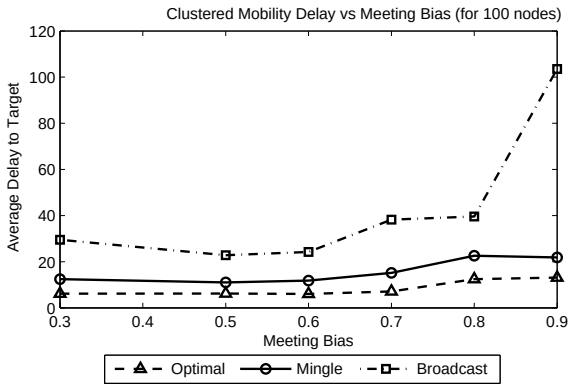

Fig. 10. Connection latency increases as the network becomes more clustered.

scheme like Mingle will probably be necessary to cope with frequent node disconnections.

In Section IV-A, we discussed Mingle-K as an optimization for reducing route-lengths in Mingle. Recall that Mingle-K initiates $(K-1)$ copies of the message in radially outward directions so that holes can be bypassed, and path-stretch from stale information can be reduced. Figure 11 shows results from increasing K. While latency reduces, the improvements were not as substantial as we had expected. The reason is as follows. In Figure 4, consider the copies that are directed radially away from the one toward the destination. Now, if these probes do not propagate too far away from the source S, then they begin to converge toward D through paths similar to the one between S and D, and are thus subject to similar (hole and staleness) problems. If the probes propagate sufficiently far, then the latency due to this detour increases proportionally. In this case, even if a redundant probe finds a better route, or bypasses a hole, the benefits may not compensate for the latency incurred during the detour. As a result, in many cases the straight line trajectory directed towards the destination proves to provide the least latency. In some cases nevertheless, the latency reduces significantly, as evident from Figure 4. Observe that since we are radially distributing the redundant probes, when using two probes, the second packet would be directed diametrically opposite from the destination. This translates to almost negligible performance benefits. When using three packets, the performance improvement is conspicuous, but the improvements saturate for higher K. However, it is worth noting that if wireless links are failure-prone, this type of redundancy may reasonably increase connection reliability. The trade-off between redundancy and reliability is a topic of our future work.

B. Transmission Costs

We report the transmission costs associated to Mingle in comparison to Periodic Broadcast (PB). The transmission cost represents the number of data packets transmitted from the source to the destination. As expected, Mingle significantly outperforms PB because the latter requires the data packets to be forwarded by all nodes in the network, irrespective of where the destination is. Since Mingle transmits along a geographic direction, and often carries the packet through mobility, the overheads are significantly lower. Figures 12 and 13 show the results for 200 individual simulation runs. The simulations are sorted for increasing transmission cost of PB.

Of course, Figures 12 and 13 may be partly unfair because Mingle incurs overheads in exchanging caches which was not accounted for. While we discuss overheads of cache exchanges

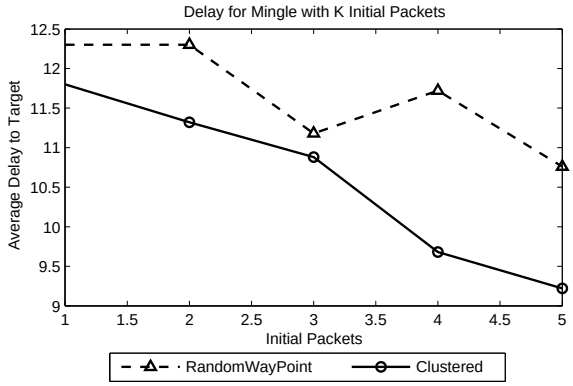

Fig. 11. Comparison of Mingle-K with K=1 and K=3.

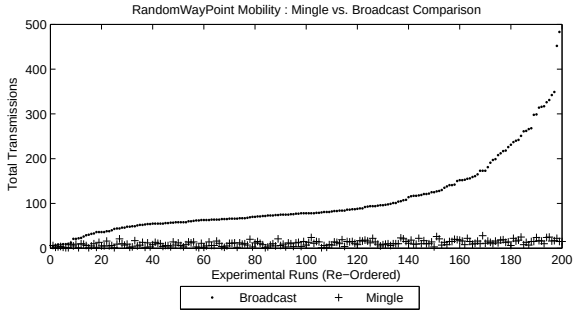

Fig. 12. Transmissions for Mingle and Broadcast in random waypoint.

in the following subsection, we note that broadcasting data packets as in PB is an unscalable solution. In particular, when data packets are of large sizes, flooding can be excessively expensive. Moreover, since flooding has to be done for every distinct source-destination pair, the overhead is not amortized. As mentioned earlier, these observations seems to suggest that proactive mechanisms may be suitable for delay tolerant networks architectures.

C. Control Overhead (from exchanging caches)

When using Mingle, nodes exchange their caches of earlier encounters with other nodes. These exchanges offer each node with a *hint* of the destination's location, which in turn eliminates the need for explicit location/route discovery. Of course, the exchange of caches incur overheads.

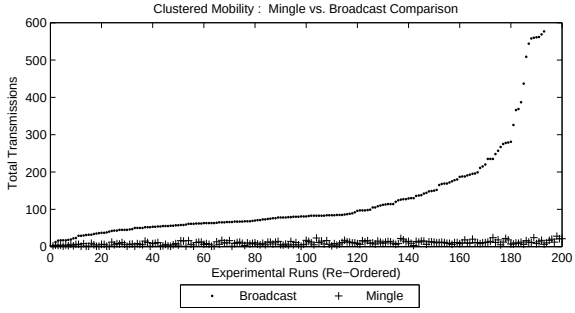

Fig. 13. Transmissions for Mingle and Broadcast in Cluster Based Model.

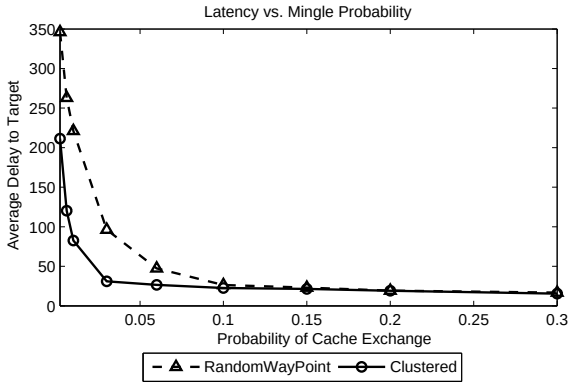

Fig. 14. Connection latency for increasing cache transaction probability.

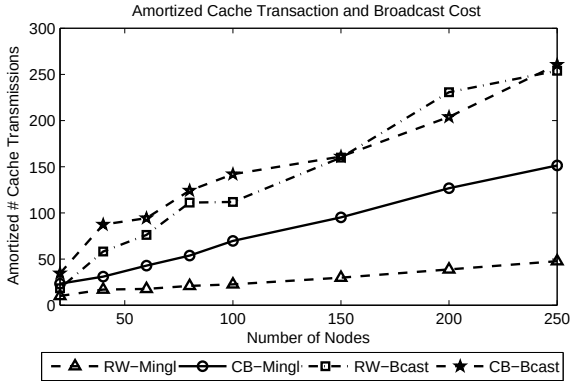

Fig. 15. Amortized control overhead with increasing network size

However, the overhead depends on the probability with which Mingle exchanges caches. With lower probability of exchanging, we expect the latency to increase as a tradeoff for lower control overheads. An important question is *how sensitive is the network performance to the exchange probability?* Figure 14 shows this result, indicating that the probability of cache exchanges can be reduced to 0.1 without significant loss in latency.

We now compare Mingle with PB in terms of amortized control overhead. The overhead is calculated as the total number of transmissions from the beginning to the end of a communication (averaged over 50 runs), amortized over all the nodes in the network. Figure 15 shows the overhead for increasing node density (the y axis is in the logarithmic scale). Evident from the figure, the amortized overhead from proactively exchanging caches is still lower than PB for both random waypoint and cluster-based mobility. This is encouraging because the overhead in Mingle does not increase with increasing traffic. In contrast, reactive route discovery schemes can increase not only over different flows, but also over multiple packets in the same flow. This is because a route in a DTN gets stale quickly, and new routes may be completely different from the old ones, requiring fresh discovery. Therefore, as long as there is active traffic in the network, a proactive scheme like Mingle proves to be cost-effective. When network traffic reduces, the proactive overheads are clearly unnecessary. We are presently working on adapting the cache-exchange probability of Mingle based on active network traffic.

VI. Discussion and Future Work

Choosing Gossip Probability: We have chosen the p_{gossip} based on the sensitivity of latency to the value of the probability. However, if the mobility models are significantly different, then a different probability might be better. An ideal solution should be able to choose this probability on the fly, perhaps by observing the network behavior in a distributed manner. One idea in this direction could be to adapt the probability based on the rate of new encounters – a node that encounters many more new nodes can choose to gossip at a higher probability. We intend to design a scheme that can estimate a suitable probability based on observable metrics, and yet uphold the good quality of location hints.

Proactive Overhead: The proactive overhead in maintaining node locations does not get well amortized when few node-pairs communicate in the network. To address this, a potential approach could be to exchange cached information only about the active nodes. Moreover, encounters that have happened in the distant past, can be excluded when the caches are exchanged. Reducing the overheads through such optimizations can prove to be beneficial. We are addressing these issues currently.

Absence of Location Information: Our scheme assumes that all nodes are aware of their respective locations. In many realistic scenarios, nodes may not be equipped with GPS or other indoor localization facilities. Routing in such scenarios is a difficult challenge. One practical approach could be to have a subset of nodes equipped with location information (one may envision deployed devices enabled with GPS). Nodes that do not have location information can acquire them whenever they are in proximity to such deployed infrastructure. Of course, the acquired locations will get noisy over time, but may still be beneficial if exploited intelligently. This is an open research problem that has not been pursued in the past, and is of interest in our future work.

VII. Conclusion

The vagaries of the wireless channel, along with node mobility, makes wireless multihop networks prone to partitions and disruptions. An emerging network model to address these problems is delay tolerant networks. As opposed to connection-oriented networking over a mobile distributed platform, DTNs are attempting to opportunistically transport packets through wireless transmissions, or physical node mobility, whichever is more suitable. In this paper, we present a communication framework that exploits the density of proliferating wireless devices, and diversity in user mobility. Using a scheme that shares knowledge of prior encounters among each other, nodes maintain a view of other nodes in the network. Messages are forwarded geographically, based on the (approximate) location of a destination. We show that even when caches are exchanged with a low frequency (low overheads), the latency of communication can be reasonably small. We are currently planning on implementing this scheme on GPS-enabled cellular phones, for field trial with students from the Duke University.

References

[1] Sushant Jain, Kevin Fall, and Rabin Patra, "Routing in a delay tolerant network," in *ACM Sigcomm*, 2004.

[2] Mirco Musolesi, Stephen Hailes, and Cecilia Mascolo, "Adaptive routing for intermittently connected mobile ad hoc networks," in *WOWMOM*, 2005.

[3] Brendan Burns, Oliver Brock, and Brian Neil Levine, "Mv routing and capacity building in disruption tolerant networks," in *INFOCOM*, 2005.

[4] Jing Zhao and Guohong Cao, "Vadd: Vehicle-assisted data delivery in vehicular ad hoc networks," in *IEEE INFOCOM*, Apr 2006.

[5] S. Douglas, J. De Couto, D. Aguayo, B. Chambers, and R. Morris, "Effects of loss rate on ad hoc wireless routing," in *MIT Laboratory of Computer Science Technical Report, MIT-LCS-TR-836*, March 2002.

[6] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Tech. Rep., Dept. of Computer Science, Duke University, October 2000.

[7] Qun Li and Daniela Rus, "Sending messages to mobile users in disconnected ad-hoc wireless networks," in *Mobicom*, 2000.

[8] Wenrui Zhao and Mostafa H. Ammar, "Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks," in *FTDCS*, 2003, pp. 308 – 314.

[9] Brent Chun and Amin Vahdat, "Workload and failure characterization on a large-scale federated testbed," Tech. Rep. IRB-TR-03-040, Intel Research Berkeley, 2003.

[10] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *Proceedings of Mobihoc*, 2004.

[11] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *INFOCOM*, 2006.

[12] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu, "The broadcast storm problem in a mobile ad hoc network," *Wireless Networking*, 2002.

[13] Philippe Jacquet, Paul Mhlethaler, and Amir Qayyum, "Optimized link state routing protocol," Internet-draft, IETF MANET Working Group, November 1998.

[14] Martin May, Vincent Lenders, and Gunar Karlsson, "Delay tolerant broadcast," in *CHANTS workshop*, 2006.

[15] Romit Roy Choudhury, "Brownian gossip: Exploiting node mobility for diffusing information in wireless networks," in *Workshop on Stochasticity in Distributed Systems (StoDis)*, 2005.

[16] A. Khelil, "Mobility-aware buffering for delay-tolerant broadcasting in manets," in *SPECTS*, 2006.

[17] Wei jen Hsu and Ahmed Helmy, "Encounter-based message broadcasting in ad hoc networks with intermittent connectivity," in *Poster in MobiHoc*, 2005.

[18] Khaled Harras, Kevin Almeroth, and Elizabeth Belding-Royer, "Delay tolerant mobile networks (dtmns): Controlled flooding in sparse mobile networks," in *IFIP Networking*, 2005.

[19] David Kempe, Alin Dobra, and Johannes Gehrke, "Gossip-based computation of aggregate information," in *IEEE Symposium on Foundations of Computer Science (FOCS'03)*, 2003.

[20] Zygmunt J. Haas, Joseph Y. Halpern, and Li Li, "Gossip-based ad hoc routing," in *Infocom*, 2002.