The Breadcrumb Router: Bundle Trajectory Tracking and Geographic Source Routing in DTN

Tomasz Kalbarczyk# Brenton Walker& Christine Julien# Angela Hennessy& Pedro Santacruz# Jonas Michel# Amy Alford\$

#The University of Texas at Austin, Austin, TX

Email: {tkalbar, pesantacruz, jonasrmichel, c.julien}@utexas.edu

&Laboratory for Telecommunications Sciences, College Park, MD

Email: {brenton, ahennes1}@math.umd.edu

\$The University of Maryland, College Park, MD

Email: aloomis@math.umd.edu

ABSTRACT

In delay-tolerant networks, generally, geographic knowledge and influence often emerge as key components of moving data around the network, but existing DTN artifacts focus almost exclusively on the networking aspects of moving bundles and not on the more inherently physical concepts of space and time. In this paper, we look at the protocols and data structures necessary to add flexible and expressive support for geographic tracking and routing to delaytolerant networks. On the one hand, our GeoTracking extension block allows for the implementation of expressive tracking of bundles' movements through space and time. On the other hand, our GeoRouting extension block and an associated BREADCRUMB router show how expressive space-time information can also be used to direct bundles through the delay-tolerant network. We start by motivating the conceptual underpinnings of GeoTracking and GeoRouting, then we describe how we integrate this functionality into the IBRDTN implementation of the bundle routing protocol. We empirically demonstrate our approaches on a set of network scenarios emulated in a DTN network emulator.

Categories and Subject Descriptors

C.2.1 [Network Architecture and Design]: Store and forward networks

Keywords

Delay-tolerant networks, bundle protocol, geo-routing

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

ExtremeCom 2014
Copyright ACM ...\$15.00.

1. INTRODUCTION

Applications in delay-tolerant networks (DTNs) often desire to both track the path(s) of data through the network and to directly influence the movement of the data. DTNs are almost always integrated into some physical space that also influences that movement of nodes, data, and the phenomena about which the nodes communicate. In traditional IP networks, the **traceroute** tool and **source routing** protocols have assisted in *tracking* and *directing* packets of data, but the focus has traditionally been on movement through the logical network and not through physical space.

When the physical and logical intertwine, data movement must often reflect various aspects of the physical space the data inhabits. We address the dual challenges of tracking data as it moves through space and time and intentionally routing data through space and time. As an exemplar of tracking, consider the need for data provenance in sensing aggregation. As an aggregate collects information sensed about a physical phenomenon, the aggregate may need to dynamically compute the data coverage by tracking where the aggregate has traveled and collected information [11]. As an exemplar of the routing challenge, consider a piece of data that measures the concentration of a gas leak. Users in the area where the gas is expected to dissipate should be warned; this can be accomplished by associating the data item with a route through space and time that captures this expected dissipation. Tracking and routing can also be combined; imagine a generic scenario in which a publisher generates a piece of data that tracks its movement en route to a subscriber. Upon receiving a publication, the subscriber sends a response that must follow the reverse path of the original publication. This generic situation is a stand-in for a variety of concrete applications. For example, the original publication may have reserved some resources along the routing path that the return response relies on. In the later sections of this paper, we use a concrete story behind this more general scenario. Specifically, we consider a maze traversal in which a prisoner in the maze sends a probe that tracks its path as it attempts to exit the maze. When it reaches the exit, the responder (e.g. a rescuer) sends a response packet back to the prisoner along the same path through the maze.

Our problem of tracking differs substantially from the goals of existing utilities, not only in terms of the transition from logical network hops to physical spaces but also because the route a bundle in a DTN takes may not be sta-

^{*}This work was funded in part by the Laboratory for Telecommunications Sciences, US Department of Defense. The opinions expressed in this paper reflect those of the authors, and do not necessarily represent those of the Department of Defense or US Federal Government.

ble (one bundle may pass through a given sequence of nodes, while a bundle sent just a minute later may take a different route). It therefore may often be important to track the route of *each* bundle. In our tracking facility, we track *both* logical network hops and the sequence of geographic locations it visits (whether because a device at one location transmits the bundle to a device at a different location or because the device holding the bundle moves). Besides being an illuminating diagnostic tool to understand the behavior of a DTN, tracking a bundle's geographic route can capture important meta-information related to the bundle's contents, as motivated above.

Source routing, in which a packet carries with it the specific network hops it must traverse, and geographic routing, where packet routing is based on physical locations, have been popular in mobile ad hoc networks [7, 8]. We combine these approaches into a geographically informed version of source routing. Previous approaches to geographic routing predominantly use a greedy approach in which locally optimal decisions are used to move a bundle incrementally closer to its destination. Our approach to geo-source routing differs in that it allows the sender to pre-specify a sequence of geo-locations that serve as routing waypoints. This style of approach can solve a variety of challenges associated with traditional geographic routing. For example by explicitly directing the geographic path of a bundle, our protocol can explicitly route around known dead-ends in the network or around known areas of congestion in the network. Combining this style of geo-routing with other approaches could, for example, ensure that network coded bundles take sufficiently diverse routes through the network.

We introduce the BREADCRUMB router, which implements geo-source routing of DTN bundles that pre-specify their delivery paths by providing a combination of geo-locations and logical network hops. Each geo-location has a margin of error, which allows the bundle to get near the specified location without having to exactly reach it. We also introduce two bundle extension blocks. The GeoRouting block holds the sequence of logical and geographical waypoints for geo-source routing. The GeoTracking block allows any bundle to collect a sequence of locations it visits in both logical and physical space. In our implementation, we rely on a GPS module to provide location information. In the paper, we use the phrases "GPS location" or "GPS coordinates" to refer to this information, but these phrases are stand-ins for any available, potentially very fine-grained location service. We describe our the BREADCRUMB router and the two extension blocks conceptually and show how we have implemented them in the IBR-DTN implementation of the bundle routing protocol [15]. We connect the BREADCRUMB router to our existing Java-based implementation of spatiotemporal trajectories [11], in which applications can perform expressive computations over data items given knowledge of their movements in space and time and demonstrate geo-tracking and geo-routing of bundles on a pair of mobility scenarios.

2. MOTIVATION

Related Work. Our BREADCRUMB router combines geographic routing and source routing to enable nodes in a DTN to explicitly specify *waypoints* that a bundle should "hit" as it moves from a source to a destination. These waypoints (as well as the final destination) may be a combination of logical addresses (e.g., node identifiers) and physical locations. In

traditional source routing, e.g., [7], each packet carries the sequence of node identifiers (e.g., IP addresses) that completely specifies the exact path the packet should follow. On the other hand, in geographic-based routing, e.g., [4, 8, 12 packets are routed, usually greedily, with the destination being a physical coordinate of a geographic location. These greedy algorithms have been increasingly fortified to mitigate the impact of the non-optimality of the greedy decisions, for example to route around topology holes [17]. Some efforts have been made to combine source routing and geographic routing but with a focus on how one can assist the other, for example by using location to reduce the overhead of discovering source routes [1]. Our combination, on the other hand, is motivated by examples in which an application desires that the routing task forces the packet to follow a pre-specified route through physical space. In vehicular ad hoc networks (VANETs), routing algorithms combat the challenges of urban scenarios, e.g., in areas dense with buildings, packets must often be routed around buildings and other obstructions of radio signals. These VANET routing protocols use junctions as waypoints to help packets navigate around obstacles [6, 10] but rely on a priori city street maps or knowledge about vehicular traffic patterns to bootstrap effective communication.

Early work on trajectory-based forwarding focused on generating and following alternative routes around congested areas [13]. In our approach, packets follow a mixture of addresses and waypoints rather than pre-generated approximate path. The *Predict and Relay* approach [18] leverages the premise that nodes often revisit waypoints and uses this observation to improve end-to-end delivery in DTNs. This work is orthogonal to ours since it improves the likelihood that bundles will be forwarded to nodes that visit waypoints along our trajectory.

Our routing technique also shares features with rumor routing [2], which generates several paths to events (trajectories). Arbitrary nodes send bundles down random paths until they reach a node along the trajectory, at which point the bundle can be forwarded to the event. A key difference in our approach is that our bundles always make progress toward the next waypoint, and we do not presume that the path between any two waypoints is previously known or static (i.e., there is no event path).

Work targeting DTNs has incorporated geographic information into DTN bundle routing. GeoSpray [16] augments traditional greedy geographic forwarding with a store-andforward behavior that improves delivery success in intermittently connected networks. Other approaches use predictions based on navigation systems [3] or explicitly relax requirements associated with location knowledge to enable geographic based routing when only partial location information is available [9]. Finally, some DTN approaches explicitly rely on the support of known infrastructure (e.g., kiosks at bus stops) to reliably route based on position information [14]. In contrast to these approaches, we rely on completely distributed and ad hoc behavior. Further, as described next, our use cases are somewhat divergent in that we assume the need for the content carried by a bundle to reach specific waypoints, as opposed to only using the waypoints to mitigate routing challenges.

Use Cases. There are myriad uses for our bundle extensions and router. We briefly describe three examples.

Safe Data Sharing. Tracking bundles is clearly widely use-

ful for debugging and general spatiotemporal provenance. As a concrete example, a secure application may want to track the movement of bundles through space to ensure that bundles (and their content) never leave a pre-specified "safe" zone [11]. By placing a *GeoTracking* extension on each bundle, the application can compare the trajectory of the bundle against the safe zone to ensure the desired property.

An Oil Spill. Consider an environmental disaster, e.g., an oil spill, detected by one or more distributed sensors. Based on the location of the spill and simultaneously sensed ambient information (e.g., water currents, winds, etc.), the sensor can compute where the spill is likely to dissipate and generate bundles that can be explicitly routed along the trajectories of dissipation. As the bundles propagate, their routing paths (i.e., trajectories) can be updated by devices they pass through based on locally sensed ambient conditions (e.g., changes in water currents or winds). Our GeoRouting extension to the Bundle protocol's block format, combined with our BREADCRUMB router, can achieve this behavior.

A Maze. Consider a prisoner held in a maze that is patrolled by guards whose devices can act as intermediate DTN nodes. The prisoner sends a bundle with a destination of a known rescuer outside of the maze. This bundle tracks its trajectory, logging a successful path out of the maze. The rescuer can send a response back to the prisoner, routed along the geo-waypoints in the maze. The response may contain information about the perils of the exit path, sensed by the prisoner's original message. Using our BREADCRUMB router, the rescuer's bundle can reach the prisoner regardless of whether the particular guards in the maze change.

3. THE GEO- EXTENSION BLOCKS AND BREADCRUMB ROUTER

We next describe the design, implementation, and architectural considerations of the GeoTracking and GeoRouting blocks and our BREADCRUMB router. Our block formats are defined based on Bundle Protocol, and we have implemented our extensions in IBR-DTN [15] stack. Therefore some of the discussion is specific to this implementation. For example, the IBR-DTN design ethos dictates that we not create any RAM-based data structures that would grow with the number of bundles held. We connect these new constructs in IBR-DTN to our existing spatiotemporal trajectories implementation [11] using the IBR-DTN Java library. Within this Java bridge implementation, we create mirrored representations of the GeoTracking and GeoRouting blocks so that Java applications can easily process GeoTracking blocks and create GeoRouting blocks; in our specific case, the application is a spatiotemporal database that stores expressively space and time-tagged data items.

3.1 The GeoTracking Block

We enabled per-bundle tracking with the GeoTracking extension block, which collects both the logical hops that the bundle traverses and the bundle's trajectory through physical space. A GeoTracking block is a series of tracking entries prefaced by a small header containing parameters for maintaining the block and counting the entries.

Figure 1 depicts the GeoTracking block's format. The Block Header fields are specified by RFC5050¹. Each GeoTracking block contains three mandatory fields:

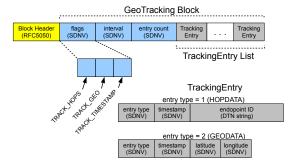


Figure 1: Format of the GeoTracking Block

Flags. The flags tell intermediate BPAs what information to append to the block. The flags are: TRACK_HOPS (0x01), TRACK_GEO (0x02), and TRACK_TIMESTAMP (0x04).

Interval. The interval (in seconds) tells intermediate BPAs the frequency with which to append a new GEODATA tracking entry to the entry list.

Entry Count. The entry count keeps track of the number of tracking entries contained in the block.

Keeping a GeoTracking block updated as the device storing the bundle moves is non-trivial for several reasons. First, there may be many bundles at a given node with GeoTracking blocks, and each bundle may have a different interval, requiring both responding to multiple timers and (in the case of IBR-DTN), reloading and storing each bundle from disk on every update. Instead, we maintain a global GPS log and update a bundle's associated GeoTracking block only when a bundle is serialized for sending. To completely satisfy any arbitrary tracking interval requirement would require recording the node's location at an interval of the GCD of all of the tracking intervals, which may not be known a priori. We assume a host-specific agent that logs GPS data to a file at a fixed global interval; bundles can request a less frequent update. Each time a GeoTracking block is serialized, we scan the log file for the necessary entries and creates the necessary tracking entries for the GeoTracking block.

This approach still has some drawbacks. First, it requires opening and reading a (potentially long) log file each time a GeoTracking block is serialized. Second, in IBR-DTN, because there is no function to "finalize" the contents of a block prior to serializing, the GPS log must be parsed twice: once when the block processor calculates the block's length, and again when the actual serialization takes place. This technically creates a race condition between these two calls, where the GPS log may get longer between the two functions. Resolving these issues completely may require some modifications to the serialization process of IBR-DTN and is reserved for future work.

Our extension blocks represent GPS coordinates in signed degrees format, where latitude ranges from -90° to 90° and longitude from -180° to 180° . However, since the self-delimiting numeric values (SDNVs)² cannot represent floating point numbers or negative values, we make two transformations to encode the values. If $\theta < 0$ we compute $\theta' = \theta + 360^{\circ}$. Then we scale all coordinates up by a factor of 1048576. This gives us at least 20 bits of precision, which is more than enough for meter-level resolution.

http://tools.ietf.org/html/rfc5050

²http://tools.ietf.org/html/draft-irtf-dtnrg-sdnv-09

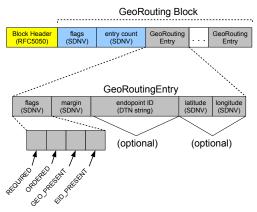


Figure 2: Format of the GeoRouting Block

We have implemented the *GeoTracking* block both in the core of IBR-DTN and within the Java API.

3.2 The GeoRouting Block

The GeoRouting extension block supports source-routing based on either intermediate geographic waypoints, logical hops (EIDs), or both. A GeoRouting block is essentially a list of geo-routing entries, each one specifying an intermediate routing goal. The GeoRouting block's format is shown in Figure 2; its main part contains only two fields: the flags (currently unused) and an entry count that specifies how many geo-routing entries follow. Each geo-routing entry has several fields. The flags specify the requirements and contents of the entry. The four flags are:

- **REQUIRED.** If set, then this entry *must* be satisfied for the bundle to be considered delivered. Otherwise the entry is considered optional.
- **ORDERED.** If set, then this entry *must* be satisfied before any successive entries can be considered. Otherwise an intermediate node can pop following entries off the list before this one is satisfied.
- **GEO_PRESENT.** If set, this entry contains a latitude/longitude pair to be used as a routing waypoint. To satisfy this entry, the bundle must visit a node that is within a specified margin of this coordinate.
- **EID_PRESENT.** If set, this entry contains an EID. To satisfy this entry, the bundle must, at some point, visit a node whose singleton EID matches the required EID.

If both GEO_PRESENT and EID_PRESENT are set, then the bundle must be carried by the node specified in the EID field to the location specified by the GPS coordinate. **REQUIRED** and **ORDERED** could be set to false if the sender intends to allow the bundle to take a shortcut if one is available; some entries could be discarded if the bundle finds itself able to skip ahead in the specified geo-trajectory. Our router's use of these fields is described below.

The block's **margin** specifies how close the block must come to each coordinate for the entry to be satisfied. The margin is given in absolute degrees. If m is the margin and x_0 and y_0 are the required longitude and latitude, then getting the bundle within the range $(x_0 \pm m, y_0 \pm m)$ is sufficient. This target area is roughly rectangular instead of circular, and a particular margin will result in different actual margins at different points on the globe. Interpreting the margin as a radius in meters would require more complicated geodetic calculations each time a node's location

changes. To represent the margin as an SDNV, we apply the same transformation as with GPS coordinates.

The GeoRouting block is easier to maintain for the block processor, but more complicated for the routing implementation. The details of these router updates are given in the next section. As with the GeoTracking extension block, we have implemented the GeoRouting block in both the IBR-DTN core and within the Java API; this allows Java-based applications to create geo-routed bundles directly.

3.3 The BREADCRUMB Router

The BREADCRUMB router performs geo-source routing using the *GeoRouting* extension block, the location of the node, and the location of neighboring peers. When it receives a bundle with a *GeoRouting* block, the router examines the entry at the top of the list, and, if the entry contains a geo entry, the router compares the entry's location against its location and the locations of neighboring peers and decides whether it or a peer is within the specified margin of the required coordinate. If the router finds a match, it pops the top entry from the list stored on the *GeoRouting* block. The router also forwards bundles without popping entries in the *GeoRouting* block if it encounters a peer that is closer to the next required waypoint.

The BREADCRUMB router's functions are divided into three tasks in the IBR-DTN routing task-queue structure:

- **SearchNextBundle.** Computes the next bundle to send to a peer; invoked when the router receives an event indicating a change in peer connectivity (e.g., a successful peer handshake or completed bundle transfer).
- **UpdateMyLocation.** Updates the router's knowledge of the node's location; queued periodically and anytime a bundle is received.

SearchNextBundle and UpdateMyLocation both require inspecting the information in each bundle's GeoRouting block, which is challenging to do efficiently in IBR-DTN since bundles are kept in persistent storage. On the other hand, maintaining a data structure in memory that contains all the geo information for each bundle is counter to the design of IBR-DTN. We devised a solution that does not require creating an additional data-structure and minimizes the retrieval of bundles from persistent storage. We rely on two IBR-DTN constructs: BUNDLE FILTERS and META BUNDLES. META BUNDLES are light-weight bundle representations that contain fields of particular interest. BUNDLE FILTERS query the storage for meta bundles that meet a set of criteria. We added three fields to the META BUNDLE:

hasgeoroute. A boolean identifier indicating that the bundle has a Georouting block.

nextgeohop. The last entry in the Georouting block. reacheddest. A boolean flag indicating that there are no more entries in the Georouting block (i.e., the bundle has reached its "final" destination)

We also created two BUNDLE FILTERS, one pertaining to each task that needs to inspect the bundles:

SearchNext. Determines which bundles to send to each peer; invoked during SearchNextBundle. For each META BUNDLE for which HASGEOROUTE is TRUE, the filter compares the location of each peer with the NEXTGEOHOP to see if the peer is closer to it than the host. If so, the meta bundle is added to the list.

UpdateLocation. Determines which Georouting blocks need updating; the decision is based on whether the location of the node is within the specified margin of error of the NEXTGEOHOP from the meta bundle.

Forwarding a bundle does not necessarily pop a geo entry off of the *GeoRouting* block; the BREADCRUMB router also greedily forwards bundles to nodes that are *closer* to the next geo waypoint, even if they are not within the specified margin of error of the waypoint.

By using the BUNDLE FILTERS, only one lookup into the persistent storage is required; it retrieves a list of META BUNDLES that need action. For SEARCHNEXTBUNDLE, the META BUNDLES contain the information necessary to determine which bundles to transfer. For UPDATEMYLOCATION, each META BUNDLE represents a bundle that needs to be pulled from persistent storage to have its *GeoRouting* block updated. This is a considerable improvement over retrieving every bundle just to inspect a *GeoRouting* block that, in the majority of cases, will not require updating.

The BREADCRUMB router defaults to epidemic routing when bundles do not contain Georouting blocks. We can therefore use a single router to build trajectories using GeoTracking blocks (i.e., to drop breadcrumbs) and to use GeoRouting blocks to return to the source (i.e., to follow breadcrumbs). The router supports single-copy routing, so that only a single copy of the bundle with a GeoRouting block persists in the network. For our prototype, the router provides ordered, geo-source routing, i.e., it assumes that all GeoRouting blocks contain entries that must be visited in order, and that each entry pertains to a particular geo location. Our GeoTracking and GeoRouting extension blocks apply more widely; supporting a heterogeneous series of entries would make the router more generalizable since it could be applied to scenarios where the order that locations are visited does not matter, or where a few specific nodes must be visited.

4. EXPERIMENTS

To demonstrate the functionality of our geo extension blocks abd BREADCRUMB router, we performed several experiments on the VirtualMeshTest (VMT) testbed [5]. VMT is an analog channel emulator based on an array of programmable attenuators. Given a desired physical arrangement of nodes, the system computes the expected path loss between nodes and dynamically programs the attenuators.

For each experiment, we define one node to be a sender and another to be a responder. Using the Java IBR-DTN bridge API, we create application-level bundles that we send through our router. The sender initially creates a probe bundle, to which it attaches a GeoTracking extension block. The sender then sends this bundle to the responder using his EID as the target of routing. When the responder receives this bundle at the application level, it creates a bundle with a GeoRouting tracking block that specifies as waypoints a subset of the locations visited by probe. The geo-routed bundle is sent back along the waypoints.

Crop Circles. In our first set of experiments, we created grids of nodes that move continuously in circles in either a clockwise or counter clockwise pattern. These *crop circles* consist of six nodes arranged as in Figure 3; the sender is the node in the lower left; the responder is the node in the upper right. Figure 4 shows the results of a single execution on the crop circles network; Figure 4(a) shows the

trajectory of the probe bundle sent from the sender to the responder, while Figure 4(b) shows the trajectory followed by the return (geo-routed) bundle. The small circles indicate the waypoints specified in the *GeoRouting* block of the responder's bundle (which were computed automatically by our application layer from the probe's *GeoTracking* block).

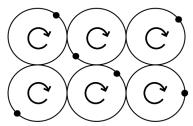


Figure 3: Crop circles mobility scenario

The track of the sender's probe bundle stops as soon as the responder (the node in the upper right corner) receives the bundle. When the responder generates its *GeoRouting* block for the return bundle, it inserts its location as the first waypoint. The track of the responder's bundle (Figure 4(b)) hits all of the waypoints specified in the *GeoRouting* bundle. In the figure, the final waypoint does not appear to quite be reached. This is because our experiments descrialize and log the tracking blocks only when a bundle arrives at a node; once the node in the top left corner hands the bundle to the node in the lower left corner, the latter node continues to move along the path shown in Figure 3, eventually crossing that final waypoint.

The Maze. Our final experiments mirror the motivating scenario of the prisoner in the maze. The maze consists of a series of connected hallways, each patrolled by a single guard who moves back and forth along hallway. Our maze is shown in Figure 5. The sender (i.e., the prisoner) is the node in the lower left corner, while the responder (i.e., the rescuer) is the node in the lower right corner.

Figure 6 shows the results of a single execution on the maze in Figure 5. The figure shows only the trajectory of the rescuer's bundle that is geo-routed back to the prisoner based on the tracked trajectory of the prisoner's probe bundle. While copies of the prisoner's probe bundle do wander down the maze's dead-end paths, the reply from the rescuer reflects only the successful path through the maze. In Figure 6, The responder's bundle successfully reaches all of the *GeoRouting* block's required waypoints without following any detours; the last waypoint is reached eventually, when the prisoner moves back along his corridor.

5. CONCLUSION

Our BREADCRUMB router implements geo-source routing by allowing nodes to specify waypoints in the form of either logical addresses or physical locations. The *GeoRouting* and *GeoTracking* extension blocks track the bundle's movement to be successfully routed through the waypoints, as demonstrated in our experiments. These results indicate that trajectory-based geo-source routing is promising as an area for future research. However, there are clearly security and privacy concerns that must be addressed. This could involve allowing nodes to turn off the bundle tracking, or encrypting or anonymizing the tracking data. We have also not yet explored how application data (such as sensor readings) could alter the trajectory that bundles take.

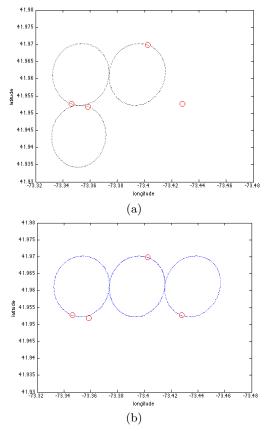


Figure 4: Crop circles tracking and trajectories. (a) The tracked trajectory of the probe bundle. (b) The tracked trajectory of the geo-routed bundle.

6. REFERENCES

- S. Basagni, I. Chlamtac, and V. Syrotiuk. Dynamic source routing for ad hoc networks using the global positioning system. In *Proc. of WCNC*, 1999.
- [2] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proc. of WSNA*, 2002.
- [3] P. Cheng, K. Lee, M. Gerla, and J. Härri. GeoDTN+Nav: Geographic DTN routing with navigator prediction for urban vehicular environments. *Mobile Networks and Applications*, 15(1):61–82, 2010.
- [4] M. Florian, S. Andreev, and I. Baumgart. OverDrive: An overlay-based geocast service for smart traffic applications. In *Proc. of MobiCom*, 2013.
- [5] D. Hahn, G. Lee, B. Walker, M. Beecher, and P. Mundur. Using virtualization and live migration in a scalable mobile wireless testbed. SIGMETRICS Perform. Eval. Rev., 38:21–25, January 2011.
- [6] M. Jerbi, S.-M. Senouci, R. Meraihi, and Y. Ghamri-Doudane. An improved vehicular routing protocol for city environments. In *Proc. of ICC*, 2007.
- [7] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, 353:153–181, 1996.
- [8] B. Karp and H. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. of MobiCom*, 2000.
- [9] E. Kuiper and S. Nadjm-Tehrani. Geographical routing with location service in intermittently

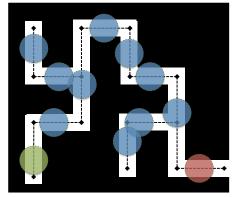


Figure 5: Maze mobility scenario

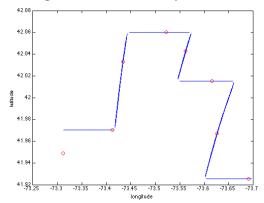


Figure 6: Maze routing trajectory

connected MANETs. *IEEE Trans. on Vehicular Technology*, 60(2):592–604, 2011.

- [10] C. Lochert, M. Mauve, H. Füssler, and H. Hartenstein. Geographic routing in city scenarios. ACM SIGMOBILE Mobile Computing and Communications Review, 9(1):69–72, 2005.
- [11] J. Michel, C. Julien, J. Payton, and G.-C. Roman. A spatiotemporal model for ephemeral data in pervasive computing networks. In *Proc. of PerHot*, 2012.
- [12] J. Navas and T. Imielinski. GeoCast: Geographic addressing and routing. In Proc. of MobiCom, 1997.
- [13] D. Niculescu and B. Nath. Trajectory based forwarding and its applications. In *Proc. of MobiCom*, 2003
- [14] H.-S. Park, J.-H. Jang, S.-H. Lee, and J.-D. Kim. Position-based DTN routing in metropolitan bus network. In *Proc. of ICSAI*, 2012.
- [15] S. Schildt, J. Morgenroth, W.-B. Pöttner, and L. Wolf. Ibr-dtn: A lightweight, modular and highly portable bundle protocol implementation. *Electronic Communications of the EASST*, 37:1–11, 2011.
- [16] V. Soares, J. Rodrigues, and F. Farahmand. GeoSpray: A geographical routing protocol for vehicular delay-tolerant networks. *Information Fusion*, 15:102–113, 2014.
- [17] J. Tian, L. Han, and K. Rothermel. Spatially aware packet routing for mobile ad hoc inter-vehicle radio networks. In *Proc. of ITS*, 2003.
- [18] Q. Yuan, I. Cardei, and J. Wu. Predict and relay: An efficient routing in disruption-tolerant networks. In *Proc. of MobiHoc*, 2009.