Disruption-Tolerant Spatial Dissemination

Bo Xing, Sharad Mehrotra and Nalini Venkatasubramanian
Department of Computer Science, University of California, Irvine
{bxing, sharad, nalini}@ics.uci.edu

***Abstract*— Spatial dissemination is a specific form of information dissemination that enables mobile users to send information to other mobile users who are or will appear at a specific location (a user-defined region). Such geo-messaging services are on the rise; they typically are built upon centralized solutions and require users to have reliable access to a stable backend infrastructure for storing and communicating content. In this paper, we develop a distributed solution to spatial dissemination, that can work without the need for such an infrastructure. Our solution utilizes the concepts from disruption-tolerant networking to build a flexible/best-effort service that leverages the intermittent ad-hoc connectivity between users. We propose *Sticker*, a spatial dissemination protocol that aims to maximize delivery reliability without incurring significant storage/transmission overheads. Sticker employs the store-carry-and-forward model, and strives to optimize dissemination performance by addressing three sub-problems - replication, forwarding and purging. Our experiments show that, Sticker achieves delivery ratios that are close to the maximum possible values; as compared to existing techniques, it either cuts down storage/transmission overheads by over 50%, or greatly enhances both delivery reliability and storage efficiency.**

I. Introduction

With advances in mobile computing, the recent years have witnessed a remarkable increase in location-based services. These services exploit geographic positions of mobile users to provide them with customized information; an increasing amount of such information is originally created by mobile users as well. Spatial dissemination is one such location-based service. It allows mobile users to post geo-referenced messages so that others who are or will be at the same place will get the messages. In other words, a message is not intended for a set of selected recipients, but rather is attached to a physical location. This is useful in scenarios where user-created information is of strong location relevance, and the information producer has no knowledge about the number and identity of the recipients. In effect, the actual recipients perceive a physical space that is digitally augmented with rich context-related information [1].

Commercial spatial dissemination services have emerged for a variety of applications, e.g., greeting friends when they are in one's neighborhood (JotYou [2]), getting alerted while driving if (as reported by other users) there likely are law enforcement personnel issuing speeding tickets (Trapster [3]), discovering entertainment facilities in the vicinity as recommended by other users (SocialLight [4]). Efforts in the research community dedicated to developing such services include GeoNotes [5], Floating Note [6] and Digital Graffiti [7].

These systems work in similar manners, once specific implementation details are abstracted out. A central server, typically residing on the Internet, maintains a database of all live messages. Mobile users publish messages by uploading them to the server. Users periodically report their positions to the server; if a user's location matches the location profile of a message in the database, the server delivers the message to the user. While this approach usually works in the presence of stable Internet access, it becomes less effective where/when Internet access is intermittent or unavailable (this is typically the case at rescue sites, in military situations, or in regions that have sparse/spotty connectivity to external worlds, e.g., in national parks, on cruise ships, etc.), and/or short-term and easily deployable services are needed. In fact, we believe that spatial dissemination will be of great use in such scenarios too. As an example, during disaster relief, a first responder with limited infrastructure support could post an "area clear" message (less mission-critical information) to the location he/she has just inspected, so that responders arriving later do not repeat the same job. Therefore, a distributed solution would be a beneficial complement to existing systems (although unable to achieve the same level of reliability) – it would make spatial dissemination services ubiquitous and facilitate easy/impromptu deployments (more application scenarios illustrated in Fig. 1).

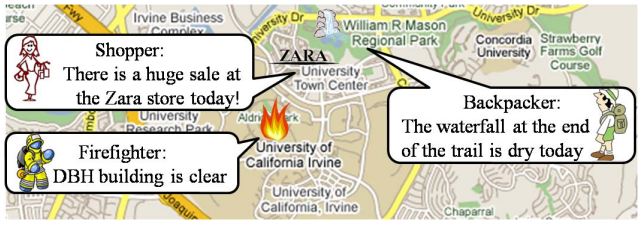


Fig. 1. Disruption-Tolerant Spatial Dissemination: Application Scenarios

Developing a distributed solution to spatial dissemination poses challenges. First, in absence of infrastructure, communications need to be carried out via the networks formed directly by mobile devices. However, mobile users likely are not connected as a network cloud at all times. Their mobility is uncontrollable, and the connectivity between them is highly dynamic. Second, in contrast to a centralized solution, in addition to content producers and consumers, mobile users also need to serve as content carriers. The messages originally stored in the database now need to be distributed to mobile devices. Despite increases in on-board storage on mobile devices, the amount of data that can be accommodated on the devices is dependant on various factors such as usage conditions and user willingness. Determining where, when and how to distribute the contents among mobile devices is a non-trivial task.

In this paper, we leverage the concepts from the disruption-tolerant networking (DTN) domain [8] to enable spatial dissemination services that tolerate communication disruptions. To accommodate the intermittent connectivity between users, we adopt the store-carry-and-forward model and exploit human mobility to facilitate message forwarding/delivery. While the foremost goal is *best-effort reliability* (to reach as many intended recipients as possible), we also aim at a balance

between reliability and other aspects of performance (storage/transmission efficiency). We propose *Sticker*, a spatial dissemination protocol that takes an interesting angle in approaching the problem, and addresses its three subproblems by employing novel strategies that are tailored to the special needs of spatial dissemination. To assess its feasibility and investigate deployment issues, we have implemented Sticker on real mobile devices. Further, through extensive simulation-based experiments, we show that Sticker outperforms existing approaches and fulfils our design goals.

In the following sections, we formally define the problem and review the literature. We elaborate on the details of the Sticker protocol in Section IV. We then sketch its implementation in Section V. We evaluate the performance of Sticker in Section VI. Finally Section VII concludes the paper.

II. Problem Formulation

*1) **Problem Statement**:* The **spatial dissemination problem** we consider in this paper is stated as follows.

Given: (i) a message M generated by a source device at a geographical position P; (ii) M's location (L), which, without loss of generality, is a circle centered at P (M's *location center*) with user-specified radius R (M's *location range*); (iii) M's lifetime (T), a user-specified time frame during which M is valid, starting from when M is generated; and (iv) that M is intended to be received by the devices that are at L at any time point during T (called the *consumers* of M).

The following criteria are optimized for: First and foremost, *Reliability*: maximum number of consumers receive M; Second, *Storage Efficiency*: least amount of aggregate storage is used (so that residual storage can be used for other purposes); Third, *Transmission Efficiency*: least number of transmissions take place (to lower energy consumption and network traffic).

*2) **System Model**:* We adopt a system model similar to what has been widely used in the DTN literature. The source device, when publishing a message (with its lifetime and location specified), adds the message (and possibly its replicas) to its cache. A cache is a local storage space on each device, containing the messages the device carries for dissemination (generated by itself or by other devices); the cache has a capacity limit (the maximum number of bytes it can accommodate). When device N_i *encounters* device N_j, N_i examines its cache and finds the messages of which N_j is a consumer; N_i delivers these messages to N_j, which presents them to the user. Meanwhile, N_i forwards certain messages to N_j for caching; N_j adds these messages to its cache if there is room. All participating devices are assumed to be able to detect their own geographical coordinates through localization technologies (e.g., GPS).

*3) **Core Questions**:* The above system model raises several questions, and accordingly yields a set of subproblems to be addressed. (i) *Replication*: How many copies should a message have? This further reduces to two sub-questions: (a) how many copies are generated for a message initially? (b) after a device forwards a message copy to another device, does it retain the copy? This question reflects the tradeoff between reliability and efficiency (more copies lead to higher reliability due to redundancy, but sacrifice storage and transmission efficiency). (ii) *Forwarding*: When two devices meet, should a message copy be forwarded from one to the other for caching? In other words, which device will more likely deliver the message to consumers? (iii) *Purging*: If a device accepting an incoming message copy would cause cache overflow on the device (i.e., violating the cache capacity constraint), should the device remove some message copies in its cache to make room for the new one, or decline it? In later sections we will elaborate on how these core questions are addressed in Sticker.

III. Related Work

Since the DTN concept was introduced, there have been many research efforts in developing routing protocols for connectivity challenged environments. While unicast routing has drawn the most attention, multicast routing [9] [10] and content-based routing (for publish/subscribe) [11] [12] were also explored. Protocols have been proposed for various application contexts, including transportation networks [13] [14], social networks [15] [16] and vehicular networks [17] [18].

In specific contexts, mobility is predictable to some extent (e.g., bus networks). RAPID [13] learns mobility patterns on the go, and models routing as a resource allocation problem. DHR [14] assumes full network knowledge, and reduces routing to a shortest path problem on a spatial-temporal graph. Solutions such as message ferries [19] and Throwboxes [20] do not make assumptions on mobility, but rather introduce special stations with non-random mobility into the system to facilitate routing.

A larger class of DTN routing protocols have been designed with general mobility (without specific patterns) in mind, the emphasis being on the design of appropriate replication and forwarding strategies. Existing replication strategies mainly include: (i) *Single copy*: initially one copy is generated for a message; a device does not retain a message copy after forwarding it (e.g., MV [21], MoVe [17], GeOpps [18], CAR [22], SimBet [15], Bubble Rap [16]); (ii) *Growing number of copies*: initially one copy is generated; a device retains a copy after forwarding it (e.g., Epidemic [23], Prophet [24], MV [21], MobySpace [25], Delegation [26], MaxHop [27], Greedy [28]); (iii) *Fixed number of identical Copies*: initially multiple copies are replicated; a device does not retain a copy after forwarding it (e.g., Spray&Wait [29], SocialCast [12]).

A forwarding strategy is typically centered around a utility, which captures the likelihood that a device will have a message delivered to destination(s). When two devices meet, each device calculates a utility value for each message; a message is forwarded for caching to the device with a higher utility value. Existing forwarding strategies vary in how they define the utility, tailored to their target application scenarios (general, vehicular, or social). Some base it on past encounters with the destination(s) (e.g., Prophet [24], MV [21], MaxHop [27], Greedy [28]); some utilize mobility matching and prediction (e.g., MobySpace [25], MoVe [17], GeOpps [18], GeoMobCast [30]); some others incorporate social analysis (e.g., SimBet [15], Bubble Rap [16], SocialCast [12]).

Only a few protocols assume finite cache capacity and consider purging. Prophet [24] uses a FIFO strategy and removes the message earliest added to cache; in Delegation [26], the message with the most copies is removed; SocialCast [12] removes a message when it exceeds its time-to-live value; RAPID [13] removes the message with the lowest utility value.

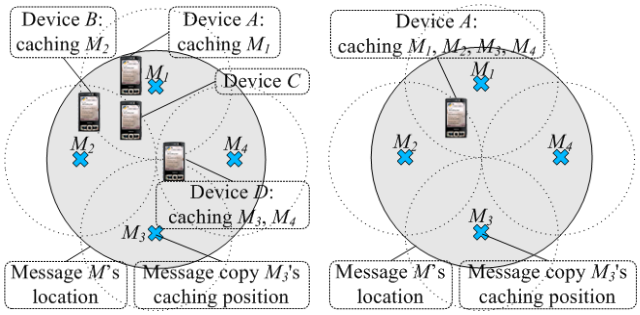


Fig. 2. Sticker's Approach to Spatial Dissemination: Examples

Existing work on spatial dissemination is fairly limited. It has been studied in the context of vehicular networks, with the assumption that nodes move in clusters with navigated routes on two-directional freeways [31]. As far as general spatial dissemination is concerned, two approaches have been dominant: flooding and scoped flooding. Epidemic [23], a general DTN routing scheme, takes a flooding-based approach – initially one copy is generated for a message; a device forwards all messages in its cache to every other device it encounters. While potentially achieving the best delivery, Epidemic incurs high overhead and could overwhelm devices with irrelevant messages. A less costly approach is scoped flooding, a representative being the scheme introduced in [32] (which we call InRange). Initially one copy is generated for a message; a device forwards a message to an encountered device only if the latter is within the message's location. InRange has two disadvantages: (1) a message can disappear if, at a time point, no device is present at the message's location; (2) when many devices are densely co-located there, all of them caching the message is highly redundant and inefficient.

IV. Sticker: The Protocol

A. Overview

Sticker's design follows two guiding principles. First, given that the mobility of devices are uncontrollable, the mobility of messages needs to be manipulated so that they stay close to the places they are intended for. Second, based on the observation that existing approaches are subject to "copy overflow" (yielding excessive copies for a message, e.g., Epidemic) or "copy underflow" (causing discontinuity to the dissemination of a message, e.g., InRange), the number of copies that a message can have needs to be well controlled throughout its lifetime.

To describe the rationale behind Sticker, we start with a simple observation – if a device caches a message, all other devices within its transmission range should be able to receive the message. Now imagine how we would approach the problem if we were given a group of stationary stations all carrying the message – we would place them in such a way that their transmission coverage altogether fully covers the location of the message; as a result, all consumers (either at the location when the message is generated, or entering the location thereafter) would be able to receive the message. In reality, unfortunately no such stations exist, but messages have to be carried by mobile devices. Nevertheless, we can mimic the scenario by *generating multiple copies for a message each tagged with the position of an imaginary station, and instructing each copy to stick to that position to the extent possible (i.e., move to or stay with the mobile device that is nearest that position).*

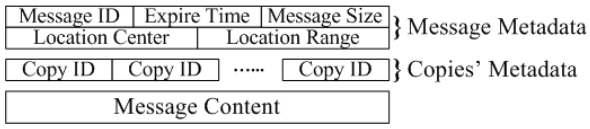


Fig. 3. Format of a Message in Cache (field sizes not to scale)

This leads to Sticker's approach to spatial dissemination, realized through its replication and forwarding strategies. For replication, Sticker adopts a *fixed number of distinct copies* strategy. That is, initially multiple copies are generated for a message, each being tagged uniquely, with its *caching position* (i.e., the ideal position of the device that caches this message copy); a device, after forwarding a message copy for caching, does not retain the copy. In concert with that, Sticker's forwarding strategy is based on location-stickiness (a utility, to be defined later) – a message copy is cached by the device that stays closer to the message copy's caching position.

The beauty of Sticker's approach lies firstly in that its functioning is irrespective of the geographical distribution of mobile devices, and thereby does not require particular device mobility patterns. Although its design philosophy originates from the imaginary scenario, Sticker does not really attempt to cover a message's entire location as the stationary stations do; rather, it covers the areas (within the message's location) where consumer devices are (this is good because to reach consumers is the ultimate goal). As a result, *a consumer device is likely either within the transmission coverage of a device caching a copy of the message, or is caching a copy itself.*

Secondly, Sticker's approach is resilient to the variations in device density, and is free to "copy overflow" and "copy underflow". When many devices are densely located within a message's location, rather than all of them caching the message, only some of them do. On the other hand, if no device is present at a message's location, the message will not disappear. These merits of Sticker can be demonstrated using the examples in Fig. 2, where mobile devices are unevenly distributed with varying density. Each copy of message M (M_1, M_2, M_3 and M_4) is cached by the device nearest its caching position. Depending on device density, some devices may cache multiple copies of M at the same time. Although M's location is not fully covered, every consumer receives M, retrieving it either from adjacent devices or its own cache. Note that, when multiple copies of a message are cached on the same device, minimum extra cache space is taken. The message content is not replicated; each copy is represented using its compact metadata – containing only an ID (in short integer), from which its caching position can be inferred (to be described later). The format of a cached message is depicted in Fig. 3.

The Sticker protocol consists of three major components, each dealing with a subproblem of spatial dissemination (Fig. 4). In addition to replication and forwarding, Sticker explicitly addresses purging (which rises when many messages have been published concurrently). To meet the special needs of spatial dissemination, it determines the admission/rejection and removal of message copies by defining aliveness-significance (a utility metric, to be defined later) and solving a variant of the 0-1 knapsack problem. In the following, we elaborate on the

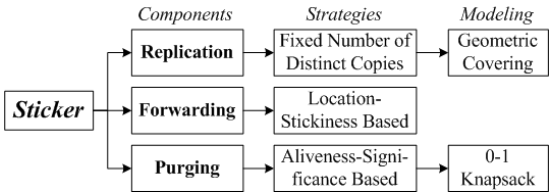


Fig. 4. An Overview of the Design of Sticker

major components of Sticker respectively.

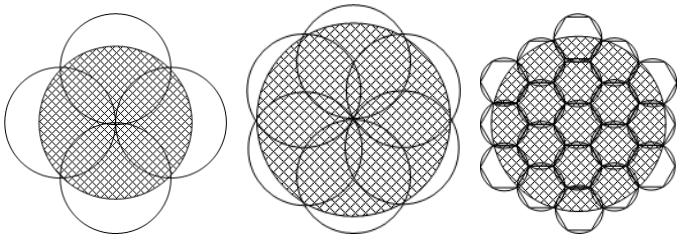
B. Replication

Sticker's replication component essentially deals with determining, for a certain message, how many copies should be generated and what are their caching positions. The goal is to minimize the number of copies, so that potential storage and transmission overheads are lowered. Based on Sticker's rationale, this translates to placing the smallest set of imaginary stations that collectively cover the location of the message. Given that a message's location is a circle of radius R, and that each device's transmission coverage can be modeled as a circle, this task reduces to the *geometric covering problem*: finding the minimum number of circles of radius r (r is the minimum transmission range of devices) and their arrangements on the plane, so that a circle of radius R is completely covered (can be generalized to arbitrary shapes). Hence, Sticker addresses replication by solving the geometric covering problem.

The geometric covering problem (and its variants) has been applied in a variety of application contexts before, e.g., coverage and monitoring scheduling for sensors [33], broadcast in sensor networks [34], etc.. Its optimal solutions are in many cases hard to find, and need to be approximated. Kershner showed in [35] that, no arrangement of circles could cover a 2-dimensional plane more efficiently than the hexagonal lattice arrangement (each hexagon is circumscribed by a circle of radius r). Moreover, it has been proved that the tight lower bound for the number of r-radius circles covering an area A is $\frac{2\sqrt{3}A}{9r^2}$; this limit is asymptotically achieved by the hexagonal lattice arrangement (when A/r^2 increases).

While being an attractive approach, the hexagonal lattice technique has its limitation – its approximation of solutions is loose when A/r^2 is small (e.g., in the circle covering circle case, if R/r is slightly larger than 1, where the optimal is 3 circles, it will find it to be 7). It is hence not a one-size-fits-all solution (especially considering that users may often post messages to small regions). Fortunately, for small A/r^2, it is not as hard to find the optimal (or near-optimal) solutions, which have been widely exposed [36].

Therefore, Sticker tackles the geometric covering problem by combining the hexagonal lattice technique and the well-known optimal solutions. It finds the optimal (or near-optimal) arrangements when R/r is small ($R/r \leq \sqrt{3}$), and using hexagonal lattice to approximate solutions when R/r is large ($R/r > \sqrt{3}$) (some example solutions are depicted in Fig. 5). By doing that, Sticker is able to keep the *approximation factor* (i.e., the ratio of the approximated number of r-radius circles over the optimal number) below 2. Based on that, a device, when publishing a message, determines the number of copies and their caching positions, and creates metadata for each copy.



(a) $\frac{2}{\sqrt{3}} < \frac{R}{r} \leq \sqrt{2}$ (b) $\frac{1}{2}+\frac{1}{2}\sqrt{5} < \frac{R}{r} \leq \sqrt{3}$ (c) $\frac{R}{r} > \sqrt{3}$

Fig. 5. Sticker's Solutions to Geometric Covering: Examples

The ID of each message copy (Fig. 3) is hence the index of either a hexagon or a circle in optimal (near-optimal) solutions.

C. Forwarding

Ideally a message copy would be cached by the device that stays closest to, or best covers its *intended region* (the r-radius circle centered at its caching position). However, there exists no oracle that has global knowledge of where each device is/moves and can find the optimal solution (as to which device should cache the message copy at each time point). With the goal being to maximize the chances that message consumers will be reached, Sticker makes greedy forwarding decisions during device encounters – a message copy is forwarded if the new carrier would better or more probably cover the copy's intended region. To this end, Sticker defines a utility metric that represents how good of a carrier a device is for a message copy, called *location-stickiness*.

Location-stickiness captures how probably a device tends to cover a message copy's intended region during a projected time period. With regard to message copy C, at time t, device N_i's location-stickiness ($ls_{N_iC}(t)$) is the normalized aggregate value of N_i's expected coverage of C's intended region (reg_C):

$$ls_{N_iC}(t) = (\sum_{\tau=0}^{k} Ec_{t+\tau})/((k+1)A_{reg_C}) \quad (1)$$

where $Ec_{t+\tau}$ is the expected overlapping area between N_i's coverage and reg_C at time $t+\tau$; it is calculated based on the probability distribution of N_i's appearance at relevant locations at time $t+\tau$ ($P_{l,t+\tau}$) and the overlapping area between N_i's coverage and reg_C if N_i is at the locations ($O_{l,t+\tau}$):

$$Ec_{t+\tau} = \sum_{l} P_{l,t+\tau} O_{l,t+\tau} \quad (2)$$

k represents the number of time points projected into the future (the duration being shorter than C's residual lifetime). A_{reg_C} is reg_C's area: $A_{reg_C} = \pi(min(r,R))^2$, used for normalizing the location-stickiness value into the range $[0,1]$ (reaching 1 if the device fully covers the region during the considered time frame). This definition takes into account both the location and motion characteristics of devices – a device that is located closer to and/or moving towards the message copy's caching position has a higher location-stickiness.

In Equation (1), k is a parameter that can be tuned to balance the tradeoff between dissemination performance and computational cost. When $k = 0$, location-stickiness considers only the current location of a device. Whereas, for $k > 0$, it involves trajectory prediction – a device estimates its probabilities of being located around a message copy's intended region at future time points. This potentially enhances dissemination performance by applying knowledge of mobility, but it adds to

the computations at mobile devices. For trajectory prediction, a few candidate techniques exist (e.g., [37] [38]), with varying accuracy and costs. We will describe what we choose to use in our implementation of Sticker in Section V.

When two devices N_i and N_j meet, determining whether to forward a message copy C for caching from N_i to N_j is thus a matter of comparing their location-stickiness regarding C ($ls_{N_iC}(t)$ and $ls_{N_jC}(t)$), and possibly their distances to C's caching position ($d_{N_iC}(t)$ and $d_{N_jC}(t)$). C is forwarded, (i) if N_j has a considerably higher location-stickiness ($ls_{N_jC}(t) - ls_{N_iC}(t) > th_1$, th_1 is a pre-defined threshold), or (ii) if the devices are so far from C's caching position that they both have zero location-stickiness ($ls_{N_iC}(t) = ls_{N_jC}(t) = 0$), while N_j has a considerably shorter distance to C's caching position ($d_{N_iC}(t) - d_{N_jC}(t) > th_2$, th_2 is a pre-defined threshold).

Once N_i has decided which message copies to transmit to N_j (for caching and/or for delivery if N_j is a consumer), N_i schedules the order they will be transmitted, such that the transmissions that would contribute more to enhanced performance happen earlier. This is important because the duration of a encounter could be short and the later transmissions probably will not be completed. To do that, N_i sorts the message copies into three segments: (i) for both delivery and caching, (ii) for delivery only, and (iii) for caching only. Within segments (i) and (iii), message copies are further sorted in decreasing order of potential gain in location-stickiness if forwared.

D. Purging

In situations where users have published many messages at the same time, devices' caches oftentimes are heavily loaded, and might overflow if accepting additional message copies to be cached. Sticker in this case needs to decide whether to remove some in-cache message copies to accommodate an incoming message copy or reject it. The goal is to keep dissemination performance minimally undermined. To quantify the potential negative impact of a message copy being removed/rejeced on dissemination performance, Sticker further defines another utility, called *Aliveness-Significance (AS)*.

In contrast to location-stickiness, which captures how much of a message copy's intended region that a device probably covers, AS is the percentage of the whole location of the message that the device probably can cover. The AS of message copy C being cached on device N_i at time t is defined as:

$$AS_{N_iC}(t) = cr_C \times ls_{N_iC}(t) \quad (3)$$

where $ls_{N_iC}(t)$ is N_i's location-stickiness with regard to C, and cr_C is C's coverage ratio. Here, the *coverage ratio* (cr_C) of a copy C of message M is the percentage of M's location supposed to be covered by C; it indicates how important C is in the dissemination of M. AS thus incorporates both device-dependent and message-specific factors. As a result, a message copy that floats close to its caching position, and/or is the only copy of a message has a low chance of being purged.

When an incoming message copy (I) arrives, Sticker considers as purging candidates only the in-cache message copies whose AS values are lower than I's (i.e., which have less significance in retaining good dissemination performance). Within this candidate set, Sticker attempts to find a subset of message copies (to be purged), whose total size exceeds the space needed for accommodating I while the total AS is minimum (thus the dissemination performance is minimally compromised). Given that every message copy has an AS value and a size (including metadata and content), this reduces to the NP-hard 0-1 knapsack problem (packing items each with a weight and a value into a sack with weight limit), if all AS values and sizes are negated. The message copies in the solution to the problem will be purged; if no solution exists, I is rejected.

Sticker employs a greedy approximation algorithm to efficiently solve the knapsack problem – the general idea being to sort the items in decreasing order of value per unit of weight, and to pack them in this order until no more space is available [39]. In the purging scenario, the algorithm works as follows. Device N_i sorts the message copies in the candidate set (using FIFO as the tie-breaker) in increasing order of AS per unit of *space contribution* (i.e., the cache space that would be released if a message copy were purged). N_i then sequentially adds the sorted copies to a purging plan (initially an empty set) until the total size of the copies in the plan exceeds the size of the incoming copy I. If that never happens, I is rejected; otherwise, the message copies in the plan are purged.

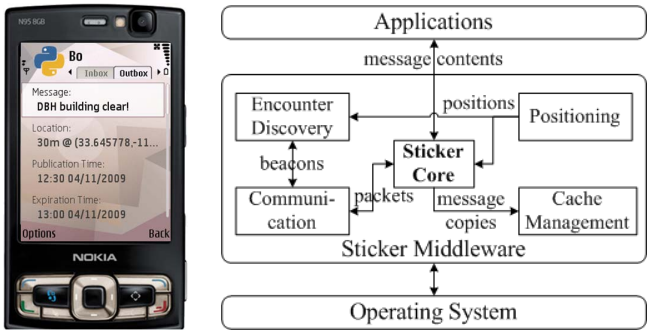
V. Implementation

Sticker has been implemented both in simulators and on real mobile devices. It is implemented using C in QualNet v4.0 [40] for simulation-based performance evaluation. It is also implemented on Nokia N95 smart phones [41] for feasibility/usability testing, using Python [42] atop the Symbian S60 platform [43].

In our implementation of Sticker's forwarding component, trajectory prediction (needed in location-stickiness measuring) is based on a hybrid scheme combining Markov chain based predication (similar to [37] and [38]) and linear prediction. The rationale is that a device's projected mobility is oftentimes dependent on its past motion history and/or current moving direction. To be able to estimate the probability distribution of its positions at future time points, each device observes its own position periodically, and builds up a Markov chain model (state space and transition probabilities); it switches to the linear predication technique for a coarse-grained estimation of moving direction when the Markov chain model alone does not suffice to make meaningful predications.

The implementation of Sticker on mobile devices is encapsulated in a middleware suite offering opportunistic messaging services to applications (a snapshot of a prototype application [44] is in Fig. 6(a)). The structure is portrayed in Fig. 6(b). Directly above the operating system is the Communication module, which deals with constructing sockets, and sending/receiving packets through wireless connectivity. Aside from Sticker Core (as detailed earlier), a set of auxiliary modules run in the background as separate threads. They interact with each other and with Sticker core through inter-thread communication mechanisms. The Encounter Discovery module is responsible for advertising presence and detecting neighboring devices. The Positioning module senses the device's position, and documents its motion history. The Cache Management module maintains the cache, physically residing in memory and/or flash storage, which is transparent to Sticker Core; In addition, it periodically inspects the cache and removes expired message copies.

Although supporting multiple connectivities for inter-device communications, our implementation primarily uses Wi-Fi ad-



(a) UI Snapshot (b) Implementation Structure

Fig. 6. Sticker: Implementation on Mobile Devices

hoc mode, as it best suits the needs of DTN-based applications. As we have experienced, other connectivities (Wi-Fi infrastructure mode and Bluetooth) have critical limitations. In case of Wi-Fi infrastructure mode, encounters are established by nearby devices connecting to a same access point and hearing each other beaconing (note that here the access point is not used for and does not need to have Internet access, but is just a relay for inter-device communications). However, many real-world access points are configured to forbid local broadcasts from devices (which is undetectable). In case of Bluetooth, encounters are discovered by scanning neighboring devices. That however is a tedious process; it takes up to 30 seconds but does not guarantee complete detections, and may prompt for pairing permissions. These observations validate our intuition in using Wi-Fi ad-hoc mode as the major connectivity.

VI. Performance Evaluation

Although we do test Sticker in real world settings, the experiments are constrained by the small number of devices we have and the limited mobility scenarios we can generate. In order to evaluate the performance of Sticker under a large variety of network scales, conditions and publication scenarios, we further examine it through simulation studies.

A. Experiment Methodology

*1) **Experiment Platform**:* We use QualNet v4.0 [40] as the simulation framework. Mobile devices are simulated by nodes that employ the IEEE802.11b MAC protocol (2Mbps bandwidth) and the two-ray propagation path-loss model. Their transmission ranges are tuned to 38m (typical for off-the-shelf devices). They use UDP as the transport protocol, and MAC-broadcast HELLO beacons periodically. Although many real mobility traces have been made available [45], none of them records both GPS coordinates and device encounters. We hence use synthetic mobility traces in our experiments. In a $1200m \times 1200m$ area, devices' movements follow the shortest path map-based mobility model, and are constrained to only valid paths (i.e., streets) on a predefined map (mimicking mobile users in real settings). Initially devices are placed at random map points (in streets). Each device selects a destination, which is a point of interest (where people tend to gather, such as parks, shopping malls and restaurants) by 0.5 probability, or a random map point by 0.5 proability. It moves towards the destination along the shortest map path (computed by the Dijkstra's algorithm), at a constant speed randomly chosen between 0.5m/s and the maximum speed. Once the destination is reached, the device waits for a pause time (a random value between 0 to 40 seconds) and selects a new destination. We generate mobility traces using the ONE simulator v1.3 [46] based on a cropped version of the included Helsinki downtown map, and convert the traces to the QualNet-recognizable format.

TABLE I
Variable Experiment Parameters

Parameter	Values	Default
Max normalized message range	1.0, 1.5, ..., 4.0	3.0
Number of devices	50, 60, ..., 110	80
Cache capacity	1KB, 2KB, ..., 64KB	64KB
Max device speed	2m/s, 4m/s, ..., 14m/s	2m/s
Max message lifetime	4min, 8min, 16min	8min
Max message size	80B, 160B, ..., 5120B	160B

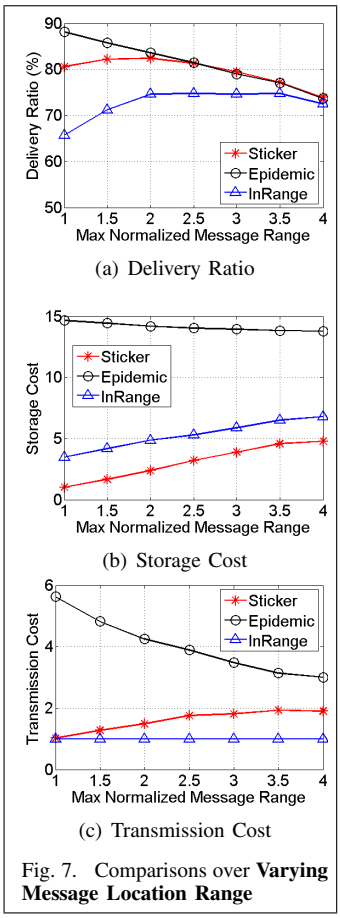
The scope and dimensions of our simulation study are summarized in Table I. In each simulation run (lasting 15 minutes), every device publishes messages following a non-homogeneous poisson process (the rate parameter λ changes over time). The mean interval ($\frac{1}{\lambda}$) between two message publications at a device is determined at runtime as a random value between 1 and 16 minutes. The attributes of a message are randomly generated. A message is minimally 10B in size, and has a lifetime of at least 1 minute; the minimum normalized location range (R/r) is 0.25. For each published message, the performance metrics (to be described shortly) are measured. Every result reported here is averaged over all published messages in simulation runs with 5 different device topologies/movements (e.g., under the default setting, every result is an average of 3210 measurements). All techniques being compared are tested with the same set of message publications under identical network conditions.

*2) **Performance Metrics**:* The performance metrics we measure capture reliability, storage efficiency and transmission efficiency, respectively. With regard to a particular message M, they are defined as follows. (i) *Delivery Ratio*: the percentage of devices receiving M out of the consumers of M (i.e., the devices that appear at M's location during M's lifetime); (ii) *Storage Cost*: the sum of the time that M is cached on each device, normalized over M's lifetime (a "single copy" replication strategy thus has storage cost of 1, serving as a baseline reference); (iii) *Transmission Cost*: the number of times M is transmitted (for delivery and forwarding), normalized over the number of devices receiving M (it is thus a value ≥ 1).

*3) **Experiment Design**:* Our first set of experiments compare Sticker with Epidemic and InRange. The goal is to examine whether Sticker can perform comparatively to or even outperforms existing approaches. Moreover, we observe Sticker's performance variation under different conditions. In the second set of experiments, we inspect the design components of Sticker. We take a closer look at its replication, forwarding and purging strategies, and compare them with other options to show the merits of Sticker's design. Our third set of experiments test Sticker dealing with content heterogeneity, where message size/lifetime is varied and Sticker's resilience is observed.

B. Experiment Results

*1) **Comparing Sticker with Existing Techniques**:* We assess the performance of Sticker by first comparing it against

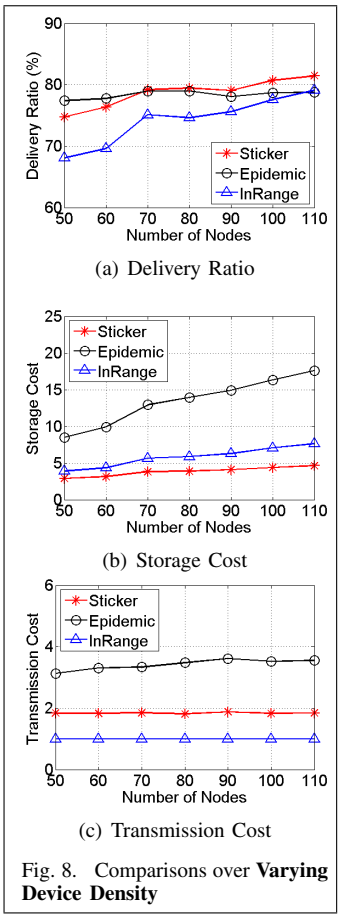


(a) Delivery Ratio

(b) Storage Cost

(c) Transmission Cost

Fig. 7. Comparisons over **Varying Message Location Range**

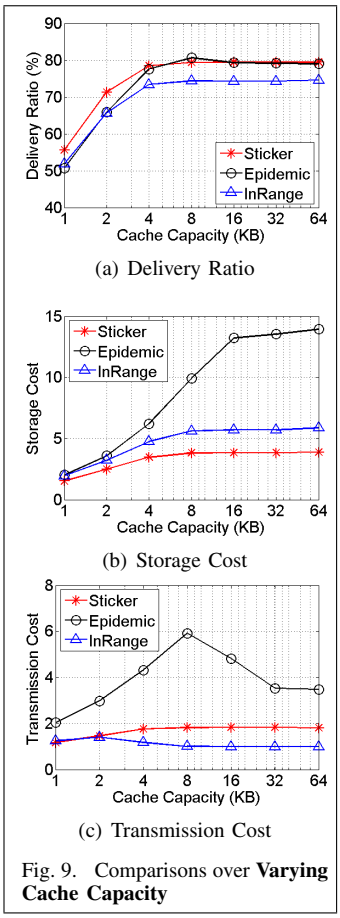


(a) Delivery Ratio

(b) Storage Cost

(c) Transmission Cost

Fig. 8. Comparisons over **Varying Device Density**

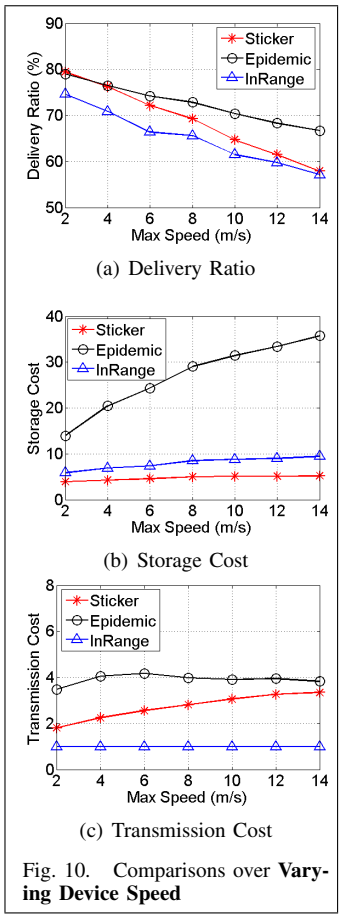


(a) Delivery Ratio

(b) Storage Cost

(c) Transmission Cost

Fig. 9. Comparisons over **Varying Cache Capacity**



(a) Delivery Ratio

(b) Storage Cost

(c) Transmission Cost

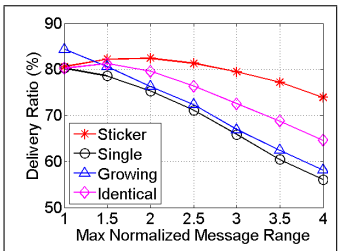
Fig. 10. Comparisons over **Varying Device Speed**

Epidemic and InRange (adopting the same purging strategy as Sticker). Recall that *Epidemic* forwards every message to every encountered device, whereas *InRange* forwards a message to a device for caching only if the device is at the message's location. They both employ the "growing number of copies" replication strategy. The comparisons are carried out along four dimensions: varying message location range(R/r) (Fig. 7), varying device density (or number of devices, Fig. 8), varying cache capacity (Fig. 9) and varying device speed (Fig. 10).

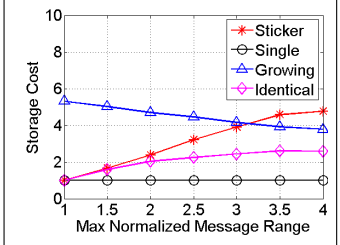
In general, Sticker achieves high delivery ratios that are very close to those achieved by **Epidemic** (which are typically maximum possible). It reaches almost all consumers that are connectivity-wise reachable by properly distributing messages to smaller set of devices (some consumers are missed due to the lower redundancy). While attaining the same level of reliability, Sticker consumes significantly lower storage and transmission costs than Epidemic, in many cases by over 50% (more when cache capacity becomes bottleneck (Fig. 9(c))). The reduction in costs largely originates from Sticker effectively preventing the number of message copies from growing overwhelmingly. As a result of the cost savings, in some scenarios (e.g., where device density is high), Sticker's delivery ratio is even higher than Epidemic (Fig. 8(a)) – Epidemic suffers from its excessive storage and transmission overheads so that many messages are dropped and a number of transmissions fail.

When compared to **InRange**, Sticker further demonstrates its advantage in reliability – its delivery ratio exceeds InRange in all tested cases. The performance gain is especially significant when message location range is small (Fig. 7(a)) and/or device density is low (Fig. 8(a)). In those cases, Sticker works consistently, whereas InRange is subject to "copy underflow" (i.e., at a point no device is present at a message's location, and all consumers are missed thereafter). Sticker also outperforms InRange in storage cost (by up to 50%) – it produces constant number of copies for a message regardless of device density, whereas InRange generates many redundant copies when devices are densely co-located. Further, Sticker consumes minimum extra space when caching multiple copies of a message. The tradeoff for Sticker's superiority in reliability and storage efficiency is its slightly more transmissions compared to InRange, whose transmission cost is constantly 1 (a message is forwarded only to consumers). This is however a cost that Sticker has to pay in favor of reliability, because to achieve better delivery there sometimes is the need for forwarding a message to non-consumers or to devices that had cached the message before.

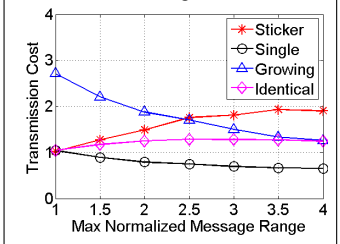
*2) **Sticker Performance under Varying Conditions**:* Observing the impact of various factors on Sticker's performance is insightful. When message location range becomes larger (Fig. 7), the storage/transmission costs go up, as more devices are involved in forwarding and larger number of messages are forwarded during device encounters. Because of that, higher ratio of transmission failures occur, leading to lower delivery ratios. With increasing device density (Fig. 8), delivery ratio gradually ascends while storage/transmission costs stay at the same level. The increase in delivery ratio is because a message's location is better covered when its carriers are more densely lo-



(a) Delivery Ratio

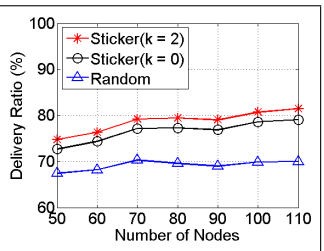


(b) Storage Cost

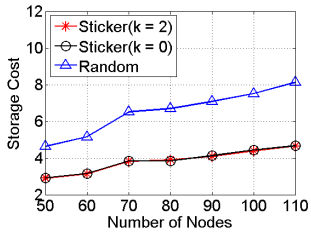


(c) Transmission Cost

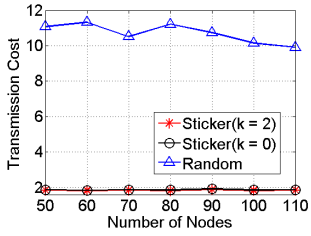
Fig. 11. Sticker **Replication** Strategy Compared with Other Options



(a) Delivery Ratio

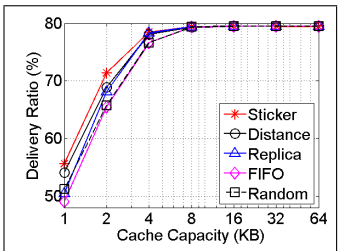


(b) Storage Cost

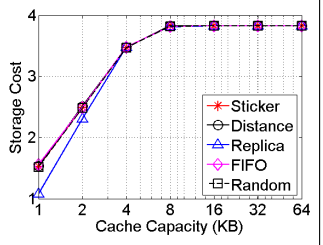


(c) Transmission Cost

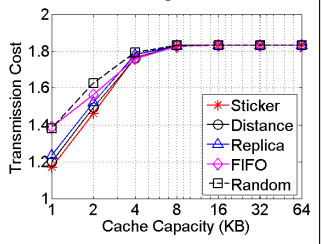
Fig. 12. Sticker **Forwarding** Strategy Compared with Other Options



(a) Delivery Ratio

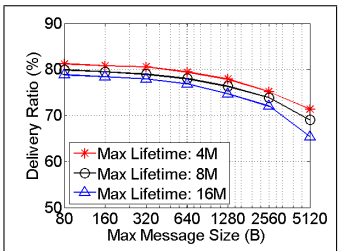


(b) Storage Cost

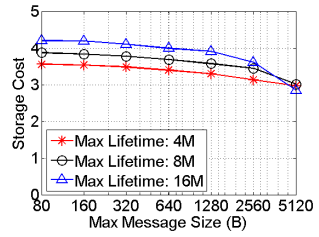


(c) Transmission Cost

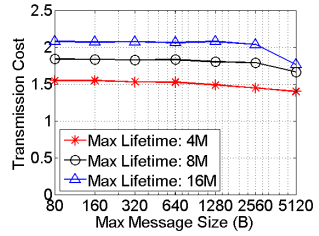
Fig. 13. Sticker **Purging** Strategy Compared with Other Options



(a) Delivery Ratio



(b) Storage Cost



(c) Transmission Cost

Fig. 14. Sticker Dealing with **Content Heterogeneity**

cated. As cache capacity is repeatedly halved, the performance of Sticker can degrade sharply (Fig. 9). This is unavoidable since many message copies cannot be accommodated in cache due to capacity limitation. However, Sticker mitigates this performance degradation as much as possible, as we will show later. Finally, compared to other techniques, Sticker is relatively susceptible to speed increases (Fig. 10), due to more frequent message forwarding. However, even when devices move as fast as 14m/s (50km/h), Sticker still outperforms InRange.

*3) **Sticker Components Design**:* Now we examine the design choices of Sticker's components. For replication, forwarding and purging, we respectively generate comparison techniques by replacing Sticker's design with alternative strategies and keeping the other two components unchanged.

Sticker's **replication** strategy – fixed number of distinct copies – is compared with single copy, growing number of copies and fixed number of identical copies (the number of copies equivalent to that in Sticker). Fig. 11 depicts the comparisons over varying message location range. As shown, Sticker reaches more consumers than other strategies in most cases (Fig. 11(a)). Only when $R/r < 1$, growing number of copies works best (because of its higher redundancy with significantly higher storage cost (Fig. 11(b)) and transmission cost (Fig. 11(c))), while all other strategies reduce to the single copy strategy. Sticker's win over single copy and growing number of copies (which has similar delivery ratio as single copy) is due to its adaptivity to the sizes of messages' locations. It outperforming fixed number of identical copies indicates the effectiveness of geo-tagging message copies differently.

The comparison technique for **forwarding** is *random forwarding*, which decides whether to forward a message copy for caching randomly by a probability. In addition, we compare the variants of Sticker to show the impact of the lookahead parameter k (in the definition of location-stickiness, Equation 1). As shown in Fig. 12, Sticker attains substantially higher delivery ratios than random forwarding (Fig. 12(a)) – Sticker makes smart selections of message carriers by using well defined utility. In terms of costs, Sticker also drastically reduces the need for storage (Fig. 12(b)) and transmissions (Fig. 12(c)). Comparing Sticker with and without trajectory prediction ($k = 2$ and $k = 0$), we see that Sticker further enhances reliability when additionally considering device mobility in making forwarding decisions (Fig. 12(a)). Trajectory prediction is helpful especially in scenarios where device mobility is restricted by e.g., streets/terrain and/or human behaviors.

Sticker's **purging** strategy is compared against the following alternatives – *Random purging*: repeatedly removing a random message copy (either an in-cache copy or the incoming copy) until cache capacity limit is not violated; *FIFO purging*: always removing the message copy that was added to cache earliest; *Location based purging*: removing (or rejecting) the message copy whose caching position is farthest from the device's position; *Replica based purging*: removing (or rejecting) the copy whose message has the largest number of copies.

The virtue of Sticker's purging strategy is exhibited in Fig. 13. When cache capacity constraints are stringent, among the

compared techniques, Sticker experiences the least degradation in delivery ratio (up to 17% less than FIFO, Fig. 13(a)) while consuming the lowest transmission cost (up to 14% lower than FIFO, Fig. 13(c)). This results from Sticker's efforts in carefully and comprehensively assessing the potential impact of purging individual message copies. Moreover, Sticker properly models purging as an optimization problem and effectively approximates the optimal solutions.

*4) **Sticker Dealing with Content Heterogeneity**:* As shown in Fig. 14, with the message size doubling up, Sticker's delivery ratio descends gradually but insignificantly. This implies that Sticker is resilient to heterogeneous content sizes under moderate device speeds. Sticker's reasonably low transmission overhead accounts for that – as small volume of forwarding takes place during encounters, growing content size does not lead to many transmission failures. Fig. 14 also reveals that message lifetime has a light impact on Sticker's performance. While it increases (more messages coexisting), delivery ratio tends to decline while costs rise. The slight performance degradation is due to the increased amount of forwarding and hence more transmission failures during encounters. Sticker demonstrates its flexibility when confronted with heavy workload.

C. Performance Evaluation Summary

In summary, our experiments have shown that Sticker meets our design goal in achieving the balance between reliability and other aspects of performance. The reliability level Sticker accomplishes is very close to the maximum possible as benchmarked by Epidemic; moreover, it reduces costs by over 50% in many cases. As compared to InRange, Sticker improves on both reliability and storage efficiency with a light tradeoff in transmission cost. Our experiments have also shown that the design of Sticker is effective – its components each outperforms alternative design options. Finally, Sticker is resilient to content heterogeneity; it best suits the scenarios where users move in moderate speeds (e.g., pedestrians), and works efficiently with messages posted to locations of all sizes.

VII. Concluding Remarks

This paper studies enabling spatial dissemination services in disruption-tolerant networking environments. We have developed a protocol that achieves high reliability while incurring low storage/transmission overheads. Undoubtedly a distributed solution to spatial dissemination does not provide guarantees on quality of service and cannot attain the reliability level achievable by centralized solutions. However, it does the best that can be done and offers a beneficial complement to centralized solutions. While we considered only the scenario where all mobile devices have localization capability, we are currently working on enabling Sticker to deal with device heterogeneity (where some devices are not localization-capable).

References

[1] T. Imielinski and B. Nath, "Wireless graffiti - data, data everywhere," in *VLDB*, 2002.
[2] "Jotyou," `http://www.jotyou.com/`.
[3] "Trapster," `http://www.trapster.com/`.
[4] "Sociallight," `http://socialight.com/`.
[5] F. Espinoza, P. Persson, and et. al, "Geonotes: Social and navigational aspects of location-based information systems," in *Ubicomp*, 2001.
[6] M. Multaharju, K. Koskinen, R. Spacil, J. Ikonen, and J. Porras, "Floating note - a location based messaging application," in *WAWC*, 2004.
[7] "Digital graffiti," `http://dg.jku.at/Digital_Graffiti/`.
[8] "Dtnrg," `http://www.dtnrg.org/`.
[9] W. Zhao, M. Ammar, and E. Zegura, "Multicasting in delay tolerant networks: Semantic models and routing algorithms," in *SigComm*, 2005.
[10] U. Lee, S. Y. Oh, K.-W. Lee, and M. Gerla, "Relaycast: Scalable multicast routing in delay tolerant networks," in *ICNP*, 2008.
[11] E. Yoneki, P. Hui, and et al., "A socio-aware overlay for publish/subscribe communication in delay tolerant networks," in *MSWIM*, 2007.
[12] P. Costa, C. Mascolo, M. Musolesi, and G. P. Picco, "Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks," *IEEE Journal On Selected Areas In Communications (JSAC)*, 2008.
[13] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "Dtn routing as a resource allocation problem," in *SIGCOMM*, 2007.
[14] C. Liu and J. Wu, "Scalable routing in delay tolerant networks," in *MobiHoc*, 2007.
[15] E. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *MobiHoc*, 2007.
[16] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay tolerant networks," in *MobiHoc*, 2008.
[17] J. LeBrun, C.-N. Chuah, and et al., "Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks," in *VTC*, 2005.
[18] I. Leontiadis and C. Mascolo, "Geopps: Geographical opportunistic routing for vehicular networks," in *AOC*, 2007.
[19] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *MobiHoc*, 2004.
[20] W. Zhao, Y. Chen, M. Ammar, M. Corner, B. Levine, and E. Zegura, "Capacity enhancement using throwboxes in dtns," in *MASS*, 2006.
[21] B. Burns, O. Brock, and B. N. Levine, "Mv routing and capacity building in disruption tolerant networks," in *INFOCOM*, 2005.
[22] M. Musolesi, S. Hailes, and C. Mascolo, "Adaptive routing for intermittently connected mobile ad hoc networks," in *WoWMoM*, 2005.
[23] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," Duke, Tech. Rep., 2000.
[24] A. Lindgreny, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," in *SAPIR*, 2004.
[25] J. Leguay, T. Friedman, and V. Conan, "Dtn routing in a mobility pattern space," in *Sigcomm Workshop on DTN*, 2005.
[26] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot, "Delegation forwarding," in *MobiHoc*, 2008.
[27] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *INFOCOM*, 2006.
[28] V. Erramilli, A. Chaintreau, M. Crovella, and C. Diot, "Diversity of forwarding paths in pocket switched networks," in *IMC*, 2007.
[29] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *Sigcomm Workshop on DTN*, 2005.
[30] M. Piorkowski, "Mobility-centric geocasting for mobile partitioned networks," in *ICNP*, 2008.
[31] I. Leontiadis, P. Costa, and C. Mascolo, "Persistent content-based information dissemination in hybrid vehicular networks," in *Percom*, 2009.
[32] D. Frey and G.-C. Roman, "Context-aware publish subscribe in mobile ad hoc networks," in *COORDINATION*, 2007.
[33] X. Bai, S. Kumar, Z. Yun, D. Xuan, and T. H. Lai, "Deploying wireless sensors to achieve both coverage and connectivity," in *MobiHoc*, 2006.
[34] A. Durresi, V. K. Paruchuri, and S. S. Iyengar, "Optimized broadcast protocol for sensor networks," *IEEE Transactions on Computers*, 2005.
[35] R. Kershner, "The number of circles covering a set," *American Journal of Mathematics*, 1939.
[36] "Circles covering circles," `http://www.stetson.edu/~efriedma/circovcir/`.
[37] M. Denko, "Mobility prediction schemes in wireless ad hoc networks," *Advanced Wired and Wireless Networks*, 2004.
[38] Q. Yuan, I. Cardei, and J. Wu, "Predict and relay: An efficient routing in disruption-tolerant networks," in *Mobihoc*, 2009.
[39] S. Sahni, "Approximation algorithms for the 0-1 knapsack problem," *Journal of ACM*, 1975.
[40] "Qualnet 4.0," `http://www.scalable-networks.com`.
[41] "Nokia n95 8gb," `https://www.nokiausa.com/A4513447`.
[42] "Pys60," `wiki.opensource.nokia.com/projects/PyS60`.
[43] "S60," `http://www.s60.com`.
[44] B. Xing, K. Seada, P. Boda, and N. Venkatasubramanian, "Passiton: An opportunistic messaging prototype on mobile devices," in *CCNC*, 2009.
[45] "Crawdad," `http://crawdad.cs.dartmouth.edu/`.
[46] A. Keränen, J. Ott, and T. Kärkkäinen, "The one simulator for dtn protocol evaluation," in *SIMUTools*, 2009.