Message Drop and Scheduling in DTNs: Theory and Practice

Amir Krifa, Chadi Barakat, *Senior Member, IEEE,* and Thrasyvoulos Spyropoulos, *Member, IEEE,*

Abstract—In order to achieve data delivery in Delay Tolerant Networks (DTN), researchers have proposed the use of *store-carry-and-forward* protocols: a node there may store a message in its buffer and carry it along for long periods of time, until an appropriate forwarding opportunity arises. This way, messages can traverse disconnected parts of the network. Multiple message replicas are often propagated to further increase delivery probability. This combination of long-term storage and message replication imposes a high storage and bandwidth overhead. Thus, efficient scheduling and drop policies are necessary to: (i) decide on the order by which messages should be replicated when contact durations are limited, and (ii) which messages should be discarded when nodes' buffers operate close to their capacity.

In this paper, we propose a practical and efficient joint scheduling and drop policy that can optimize different performance metrics, such as average delay and delivery probability. We first use the theory of encounter-based message dissemination to derive the optimal policy based on global knowledge about the network. Then, we introduce a method that estimates all necessary parameters using locally collected statistics. Based on this, we derive a distributed scheduling and drop policy that can approximate the performance of the optimal policy in practice. Using simulations based on synthetic and real mobility traces, we show that our optimal policy and its distributed variant outperform existing resource allocation schemes for DTNs. Finally, we study how sampled statistics can reduce the *signaling* overhead of our algorithm and examine its behavior under different congestion regimes. Our results suggest that close to optimal performance can be achieved even when nodes sample a small percentage of the available statistics.

Index Terms—Delay Tolerant Network, Congestion, Drop Policy, Scheduling Policy

1 Introduction

Mobile ad hoc networks (MANETs) had been treated, until recently, as a connected graph over which end-to-end paths need to be established. This legacy view might no longer be appropriate for modelling existing and emerging wireless networks [1], [2], [3]. Wireless propagation phenomena, node mobility, power management, etc. often result in intermittent connectivity with end-to-end paths either lacking or rapidly changing. To allow some services to operate even under these challenging conditions, researchers have proposed a new networking paradigm, often referred to as Delay Tolerant Networking (DTN [4]), based on the *store-carry-and-forward* routing principle [1]. Nodes there, rather than dropping a session when no forwarding opportunity is available, store and carry messages until new communication opportunities arise.

Despite a large amount of effort invested in the design of efficient routing algorithms for DTNs, there has not been a similar focus on queue management and message scheduling. Yet, the combination of long-term storage and the, often expensive, message replication performed by many DTN routing protocols [5], [6] impose a high bandwidth and storage overhead on wireless nodes [7]. Moreover, the data units disseminated in this context, called *bundles*, are self-contained, application-level data units, which can often be large [4]. As a result, it is expected that nodes' buffers, in this context, will often operate at full capacity. Similarly, the available bandwidth during a contact could be insufficient to communicate all intended messages. Consequently, *regardless of the specific routing algorithm used*, it is important to have: (i) efficient drop policies to decide which message(s) should be discarded when a node's buffer is full, and (ii) efficient scheduling policies to decide which message(s) should be chosen to exchange with another encountered node when bandwidth is limited and in which order.

In this paper, we try to solve this problem in its foundation. We develop a theoretical framework based on Epidemic message dissemination [8], [9], [10], and propose an optimal joint scheduling and drop policy, GBSD (Global knowledge Based Scheduling and Drop) that can maximize the average delivery rate or minimize the average delivery delay. GBSD derives a per-message utility by taking into account all information that are relevant for message delivery, and manages messages accordingly. Yet, to derive these utilities, it requires *global* network information, making its implementation difficult in practice, especially given the intermittently connected nature of the targeted networks. In order to amend this, we propose a second policy, HBSD (History Based Scheduling and Drop), a distributed (*local*) algorithm based on statistical learning. HBSD uses network history to estimate the current state of required (*global*) network parameters and uses these estimates, rather than actual values (as in GBSD), to calculate message utilities for each performance target metric.

- *Amir Krifa and Chadi Barakat are with the Project-Team Planète, INRIA Sophia-Antipolis, France.*
  E-mail(s): Amir.Krifa@inria.fr, Chadi.Barakat@inria.fr
- *Thrasyvoulos Spyropoulos is with the Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.*
  E-mail: spyropoulos@tik.ee.ethz.ch

To our best knowledge, the recently proposed RAPID protocol [11] is the only effort aiming at scheduling (and to a lesser extend message drop) using a similar theoretical framework. Yet, the utilities derived there are sub-optimal, as we will explain later, and require global knowledge (as in GBSD), raising the same implementation concerns. Simulations using both synthetic mobility models and real traces show that our HSBD policy not only outperforms existing buffer management and scheduling policies (including RAPID), but can also approximate the performance of the reference GBSD policy, in all considered scenarios.

Furthermore, we look deeper into our distributed statistics collection solution and attempt to identify the available tradeoffs between the collection overhead and the resulting performance. Aggressively collecting statistics and exchanging them with every encountered node allows estimates to converge faster, but it can potentially result in high energy and bandwidth consumption, and also interfere with data transmissions. Our results suggest that close to optimal performance can still be achieved even when the signaling overhead is forced (through sampling) to take only a small percentage of the contact bandwidth.

Finally, we examine how our algorithm behaves under different congestion regimes. Interestingly, we find that (i) at low to moderately congested regimes, the optimal policy is simply equivalent to dropping the message with the oldest age (similarly to the findings of [12]), while (ii) at highly congested regimes, the optimal policy is not linear on message age; some young messages have to be dropped, as a means of indirect admission control, to allow older messages to create enough replicas and have a chance to be delivered. Hence, our framework can also explain what popular heuristic policies are doing, in this context, relative to the optimal one.

The rest of this paper is organized as follows. Section 2 describes the current state-of-the art in terms of buffer management and scheduling in DTNs. In Section 3, we describe the "reference", optimal joint scheduling and drop policy that uses global knowledge about the network. Then, we present in Section 4 a learning process that enables us to approximate the global network state required by the reference policy. Section 5 discusses our evaluation setup and presents performance results for both policies (GBSD and HBSD) using synthetic and real mobility traces. In Section 6, we examine in detail our mechanism to collect and maintain network history statistics, and evaluate the signaling-performance tradeoff. Section 7 studies the behavior of our HBSD policy in different congestion regimes. Finally, we conclude this paper and discuss future work in Section 8.

2 State of the art

A number of sophisticated solutions have been proposed to handle routing in DTNs. Yet, the impact of buffer management and scheduling policies on the performance of the system has been largely disregarded, in comparison, by the DTN community.

In [13], Zhang et al. present an analysis of buffer constrained *Epidemic* routing, and evaluate some simple drop policies like drop-front and drop-tail. The authors conclude that drop-front, and a variant of it giving priority to source messages, outperform drop-tail in the DTN context. A somewhat more extensive set of combinations of *heuristic* buffer management policies and routing protocols for DTNs is evaluated in [12], confirming the performance of drop-front. In [14], Dohyung et al. present a drop policy which discards a message with the largest expected number of copies first to minimize the impact of message drop. However, all these policies are heuristic, i.e. not explicitly designed for optimality in the DTN context. Also, these works do not address scheduling. In a different work [15], we address the problem of optimal drop policy only (i.e. no bandwidth or scheduling concerns) using a similar analytical framework, and have compared it extensively against the policies described in [13] and [12]. Due to space limitations, we do not repeat these results here. We rather focus on the more general *joint scheduling and drop problem*, for which we believe the RAPID protocol [11] represents the state-of-the-art.

RAPID is the first protocol to explicitly assume both bandwidth and (to a lesser extent) buffer constraints exist, and to handle the DTN routing problem as an optimal resource allocation problem, given some assumption regarding node mobility. As such, it is the most related to our proposal, and we will compare directly against it. Despite the elegance of the approach, and performance benefits demonstrated compared to well-known routing protocols, RAPID suffers from the following drawbacks: (i) its policy is based on suboptimal message utilities (more on this in Section 3); (ii) in order to derive these utilities, RAPID requires the flooding of information about all replicas of a given message in the queues of all nodes in the network; yet, the information propagated across the network might arrive stale to nodes (a problem that the authors also note) due to change in the number of replicas, change in the number of messages and nodes, or if the message is delivered but acknowledgements have not yet propagated in the network; and (iii) RAPID does not address the issue of signalling overhead. Indeed, in [11], the authors showed that whenever the congested level of the network starts increasing, their meta-data channel consumes more bandwidth. This is rather undesirable, as meta-data exchange can start interfering with data transmissions amplifying the effects of congestion. In another work [16], Yong et al. present a buffer management schema similar to RAPID. However they do not address the scheduling issue nor the tradeoff between the control channel overhead and system performance. In this paper, we successfully address all these three issues.

3 Optimal Joint Scheduling and Drop Policy

In this section, we first describe our problem setting and the assumptions for our theoretical framework. We then use this framework to identify the optimal policy, GBSD (Global Knowledge based Scheduling and Drop). This policy uses global knowledge about the state of each message in the network (number of replicas). Hence, it is difficult to implement it in a real world scenario, and will only serve as reference. In the next section, we will propose a distributed algorithm that can successfully approximate the performance of the optimal policy.

3.1 Assumptions and Problem Description

We assume there are L total nodes in the network. Each of these nodes has a buffer, in which it can store up to B messages in transit, either messages belonging to other nodes or messages generated by itself. Each message has a Time-To-Live (TTL) value, after which the message is no more useful to the application and should be dropped by its source and all intermediate nodes. The message can also be dropped when a notification of delivery is received, or if an "anti-packet" mechanism is implemented [13].

Routing: Each message has a single destination (unicast) and is assumed to be routed using a replication-based scheme [7]. During a contact, the routing scheme used will create a list of messages to be replicated among the ones currently in the buffer. Thus, different routing schemes might choose different messages. For example, epidemic routing will replicate all messages not already present in the encountered node's buffer [5]. For the purposes of this paper, we will use epidemic routing as a case study, for the following reasons. First, its simplicity allows us to concentrate on the problem of resource allocation, which is the focus of this paper. Second, it consumes the most resources per message compared to any other scheme. As a result, it can be easily driven to medium or high congestion regimes, where the efficient resource allocation problem is most critical. Third, given the nature of random forwarding schemes, unless a buffer is found full or contact capacity is not enough to transfer all messages, epidemic forwarding is optimal in terms of delay and delivery probability. Consequently, epidemic routing along with appropriate scheduling and message drop policies, can be viewed as a new routing scheme that optimally adapts to available resources [11]. Finally, we note that our framework could be used to treat other types of traffic (e.g. multicast), as well. However, we focus on unicast traffic to elucidate the basic ideas behind our approach, and defer the treatment of multi-point traffic to future work.

Mobility Model: Another important element in our analytical framework is the impact of mobility. In the DTN context, message transmissions occur only when nodes encounter each other. Thus, *the time elapsed between node meetings is the basic delay component*. The meeting time distribution is a basic property of the mobility model assumed [10], [9][1]. To formulate the optimal policy problem, we will first assume a class of mobility models that has the following properties:

- A.1 Meeting times are exponentially distributed or have at least an *exponential tail*;
- A.2 Nodes move *independently* of each other;
- A.3 Mobility is homogeneous, that is, all node pairs have the same meeting rate λ.

Regarding, the first assumption, it has been shown that many simple synthetic mobility models like Random Walk, Random Waypoint and Random Direction [10], [9] have such a property. Furthermore, it is a known result in the theory of random walks on graphs that hitting times on subsets of vertices usually have an exponential tail [19]. Finally, it has recently been argued that meeting and inter-meeting times observed in many traces also exhibit an exponential tail [20]. In our framework, *we sample the remaining meeting time only when a drop or scheduling decision needs to be taken*, in order to calculate the drop probability of Eq.(2). In a sparse network (as in our case), it can be shown that, at this time, the two nodes in question have already *mixed* with high probability. Thus, the quantity sampled can be approximated by the meeting time from stationarity, or the tail of the inter-meeting time distribution, which, as explained, is often exponential [**?**]. In other words, it is not required to make the stronger assumption of Poisson distributed inter-meeting times, as often done in related literature.

Regarding the second assumption, although it might not always hold in some scenarios, it turns out to be a useful approximation. In fact, one could use a mean-field analysis argument to show that independence is not required, in the limit of large number of nodes, for the analytical formulas derived to hold (see e.g. [21]).

Finally, in Section 3.4, we discuss how to remove assumption [A.3] and generalize our framework to heterogenous mobility models.

Buffer Management and Scheduling: Let us consider a time instant when a new contact occurs between nodes i and j. The following resource allocation problem arises when nodes are confronted with limited resources (i.e. contact bandwidth and buffer space)[2].

(*Scheduling Problem*) If i has X messages in its *local* buffer that it should forward to j (chosen by the routing algorithm), but does not know if the contact will last long enough to forward all messages, which ones should it send first, so as to maximize the *global* delivery probability for *all* messages currently in the network?

1. By *meeting time* we refer to the time until two nodes starting from the stationary distribution come within range ("first meeting-time"). If some of the nodes in the network are static, then one needs to use *hitting times* between mobile and static nodes. Our theory can be easily modified to account for static nodes by considering, for example, two classes of nodes with different meeting rates (see e.g. [18]).

2. We note that, by "limited resources", we do not imply that our focus is only small, resource-limited nodes (e.g. wireless sensors), but rather that the offered forwarding or storage load exceeds the available capacity. In other words, we are interested in congestion regimes.

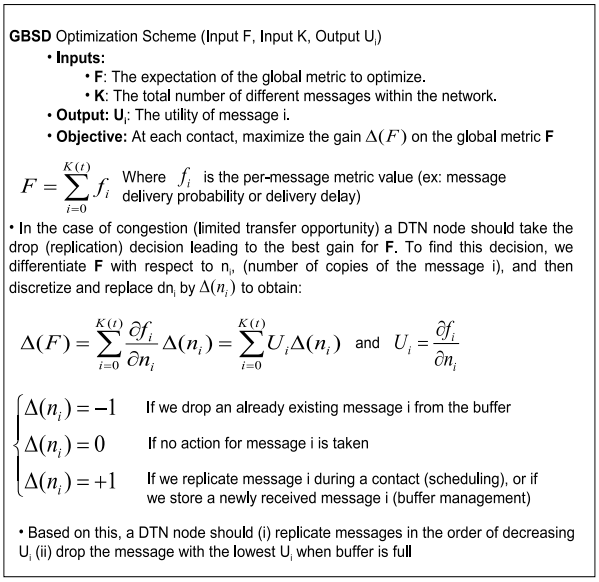

Fig. 1. GBSD Global optimization policy

(*Buffer Management Problem*) If one (or more) of these messages arrive at j's buffer and find it full, what is the best message j should drop among the ones already in its buffer (*locally*) and the newly arrived one, in order to maximize, let's say, the average delivery rate among all messages in the network (*globally*)?

To address these two questions, we propose the following policy. Given a routing metric to optimize, *our policy, GBSD, derives a per-message utility that captures the* marginal value *of a given message copy, with respect to the chosen optimization metric*. Based on this utility, two main functions are performed:

1) *Scheduling*: at each contact, a node should replicate messages in decreasing order of their utilities.
2) *Drop*: when a new message arrives at a node with a full buffer, this node should drop the message with the smallest utility among the one just received and the buffered messages.

We will derive next such a per-message utility for two popular metrics: maximizing the average delivery probability (rate), and minimizing the average delivery delay. Table 1 contains some useful notation that we will use throughout the paper. Finally, the GBSD optimization policy is summarized in Figure 1.

3.2 Maximizing the average delivery rate

We first look into a scenario where each message has a *finite* TTL value. The source of the message keeps a copy of it during the whole TTL duration, while intermediate nodes are not obliged to do so. To maximize the average delivery probability among all messages in the network, the optimal policy must use the per message utility derived in the following theorem, in order to perform scheduling and buffer management.

Theorem 3.1. *Let us assume that there are K messages in the network, with elapsed time T_i for the message i. For each message $i \in [1, K]$, let $n_i(T_i)$ be the number of nodes who have a copy of the message at this time instant, and $m_i(T_i)$ those that have "seen" the message (excluding the source) since its creation*[3] *($n_i(T_i) \leqslant m_i(T_i) + 1$). To maximize the average delivery rate of all messages, a DTN node should apply the GBSD policy using the following utility per message i:*

$$U_i(DR) = (1 - \frac{m_i(T_i)}{L-1})\lambda R_i \exp(-\lambda n_i(T_i) R_i). \quad (1)$$

Proof: The probability that a copy of a message i will not be delivered by a node is given by the probability that the next meeting time with the destination is greater than R_i, the remaining lifetime of a message ($R_i = TTL - T_i$). This is equal to $\exp(-\lambda R_i)$ under our assumptions.

Knowing that message i has $n_i(T_i)$ copies in the network, and assuming that the message has not yet been delivered, we can derive the probability that the message itself will not be delivered (i.e. none of the n_i copies gets delivered):

$$P\{\text{message } i \text{ not delivered} \mid \text{not delivered yet}\} = \prod_{k=1}^{n_i(T_i)} \exp(-\lambda R_i) = \exp(-\lambda n_i(T_i) R_i). \quad (2)$$

Here, we have not taken into account that more copies of a given message i may be created in the future through new node encounters. We have also not taken into account that a copy of message i could be dropped within R_i (and thus this policy is to some extent "greedy" or "locally optimal", with respect to the time dimension). Predicting future encounters and the effect of further replicas created complicates the problem significantly. Nevertheless, the same assumptions are applied for all messages equally and thus can justify the relative comparison between the delivery probabilities.

We need to also take into consideration what has happened in the network since the message generation, in the absence of an explicit delivery notification (this

TABLE 1
Notation

Variable	Description
L	Number of nodes in the network
K(t)	Number of distinct messages in the network at time t
TTL_i	Initial Time To Live for message i
R_i	Remaining Time To Live for message i
$T_i = TTL_i - R_i$	Elapsed Time for message i. It measures the time since this message was generated by its source
$n_i(T_i)$	Number of copies of message i in the network after elapsed time T_i
$m_i(T_i)$	Number of nodes (excluding source) that have *seen* message i since its creation until elapsed time T_i
λ	Meeting *rate* between two nodes; $\lambda = \frac{1}{E[H]}$ where $E[H]$ is the average meeting time

3. We say that a node A has "seen" a message i, when A had received a copy of message i in the past, regardless of whether it still has the copy or has already removed it from its buffer.

part is not considered in RAPID [11], making the utility function derived there suboptimal). Given that all nodes including the destination have the same chance to see the message, the probability that a message i has been already delivered is equal to:

$$P\{\text{message } i \text{ already delivered}\} = \quad m_i(T_i)/(L-1). \tag{3}$$

Combining Eq.(2) and Eq.(3), the probability that a message i will get delivered before its TTL expires is:

$$\begin{aligned} P_i &= P\{\text{message } i \text{ not delivered yet}\} * (1 - \exp(-\lambda n_i(T_i)R_i)) \\ &\quad + P\{\text{message } i \text{ already delivered}\} \\ &= (1 - \frac{m_i(T_i)}{L-1}) * (1 - \exp(-\lambda n_i(T_i)R_i)) + \frac{m_i(T_i)}{L-1}. \end{aligned}$$

So, if we take at instant t a snapshot of the network, the global delivery rate for the whole network will be:

$$DR = \sum_{i=1}^{K(t)} \left[(1 - \frac{m_i(T_i)}{L-1}) * (1 - \exp(-\lambda n_i(T_i)R_i)) + \frac{m_i(T_i)}{L-1} \right].$$

In case of a full buffer or limited transfer opportunity, a DTN node should take respectively a drop or replication decision that leads to the best gain in the global delivery rate DR. To define this optimal decision, we differentiate DR with respect to $n_i(T_i)$,

$$\begin{aligned} \Delta(DR) &= \sum_{i=1}^{K(t)} \frac{\partial P_i}{\partial n_i(T_i)} * \triangle n_i(T_i) \\ &= \sum_{i=1}^{K(t)} \left[(1 - \frac{m_i(T_i)}{L-1}) \lambda R_i \exp(-\lambda n_i(T_i)R_i) * \triangle n_i(T_i) \right] \end{aligned}$$

Our aim is to maximize $\Delta(DR)$. In the case of message drop, for example, we know that: $\Delta n_i(T_i) = -1$ if we drop an already existing message i from the buffer, $\Delta n_i(T_i) = 0$ if we don't drop an already existing message i from the buffer, and $\Delta n_i(T_i) = +1$ if we keep and store the newly-received message i. Based on this, GBSD ranks messages using the per message utility in Eq.(1), then schedules and drops them accordingly. This utility can be viewed as the *marginal utility value* for a copy of a message i with respect to the total delivery rate. The value of this utility is a function of the global state of the message i (n_i and m_i) in the network. □

3.3 Minimizing the average delivery delay

We next turn our attention to minimizing the average delivery delay. We now assume that all messages generated have infinite TTL or at least a TTL value large enough to ensure a delivery probability close to 1. The following Theorem derives the optimal per-message utility, for the same setting and assumptions as Theorem 3.1.

Theorem 3.2. *To minimize the average delivery delay of all messages, a DTN node should apply the GBSD policy using the following utility for each message i:*

$$U_i(DD) = \frac{1}{n_i(T_i)^2 \lambda} (1 - \frac{m_i(T_i)}{L-1}). \tag{4}$$

Proof: Let us denote the delivery delay for message i with random variable X_i. This delay is set to 0 (or any other constant value) if the message has been already delivered. Then, the total expected delivery delay (DD) for all messages for which copies still exist in the network is given by,

$$DD = \sum_{i=1}^{K(t)} \left[\frac{m_i(T_i)}{L-1} * 0 + (1 - \frac{m_i(T_i)}{L-1}) * E[X_i | X_i > T_i] \right]. \tag{5}$$

We know that the time until the first copy of the message i reaches the destination follows an exponential distribution with mean $1/(n_i(T_i)\lambda)$. It follows that,

$$E[X_i | X_i > T_i] = T_i + \frac{1}{n_i(T_i)\lambda}. \tag{6}$$

Substituting Eq.(6) in Eq.(5), we get,

$$DD = \sum_{i=1}^{K(t)} (1 - \frac{m_i(T_i)}{L-1})(T_i + \frac{1}{n_i(T_i)\lambda}).$$

Now, we differentiate D with respect to $n_i(T_i)$ to find the policy that maximizes the improvement in D,

$$\Delta(DD) = \sum_{i=1}^{K(t)} \frac{1}{n_i(T_i)^2 \lambda} (\frac{m_i(T_i)}{L-1} - 1) * \Delta n_i(T_i).$$

The best drop or forwarding decision will be the one that maximizes $|\Delta(DD)|$ (or $-\Delta(DD)$). This leads to the per message utility in Eq.(4).

□

Note that, the per-message utility with respect to delivery delay is different than the one for the delivery rate. This implies (naturally) that both metrics cannot be optimized concurrently.

3.4 The Case of Non-Homogeneous Mobility

Throughout our analysis, we have so far assumed homogeneous node mobility. Recent measurement studies have revealed that, often, different node pairs might have different meeting rates. We extend here our analytical framework, in order to derive per-message utilities that maximize the global performance metric, in face of such heterogeneous mobility scenarios. We illustrate the extension with the delivery rate [4]. Specifically, we assume that meetings between a given node pair are exponentially distributed with meeting rate $\tilde{\lambda}$, where $\tilde{\lambda}$ is a *random variable* such that:

$$\tilde{\lambda} \in [0, \infty), \text{distributed as } f(\tilde{\lambda}).$$

$f(\tilde{\lambda})$ is a probability distribution that models the heterogeneous meeting rates between nodes, and can be any

4. The treatment of delivery delay utilities does not involve Laplace transforms, but poses no extra difficulties. We thus omit it here, due to space limitations

function integrable in $[0, \infty)$, capturing thus a very large range of conceivable mobility models.

The analysis of Theorem 3.2 is thus modified as follows. Let's assume that message i has n_i copies in the network, and that the n_i carriers have (unknown) meeting rates $\tilde{\lambda_1}, \tilde{\lambda_2}, \ldots, \tilde{\lambda_{n_i}}$, respectively. Eq.(2) becomes:

$$P\{\text{message } i \text{ not delivered} \mid \text{not delivered yet}\} = E_{\tilde{\lambda_1}, \tilde{\lambda_2}, \ldots, \tilde{\lambda_{n_i}}}[\prod_{j=1}^{n_i} \exp(-\tilde{\lambda_j} R_i)] = \quad (7)$$

$$\prod_{j=1}^{n_i} \int_0^{\infty} \exp(-\tilde{\lambda_j} R_i) f(\tilde{\lambda_j}) d\lambda_j = (F_{\mathcal{L}}(R_i))^{n_i}, \quad (8)$$

where $F_{\mathcal{L}}(R_i)$ is the Laplace transform of distribution $f(x)$ evaluated at R_i. Continuing as in the proof of Theorem 3.2, we get the unconditional probability of delivery P_i:

$$P_i = (1 - \frac{m_i}{L-1}) * (F_{\mathcal{L}}(R_i))^{n_i} + \frac{m_i}{L-1}.$$

Differentiating P_i with respect to n_i, we derive the following generic marginal utility per message:

$$(1 - \frac{m_i}{L-1}) * \ln(F_{\mathcal{L}}(R_i)) * (F_{\mathcal{L}}(R_i))^{n_i}. \quad (9)$$

We now consider some example distributions for node meeting rates, and derive the respective marginal utility.

Dirac delta funtion: Let $f(\tilde{\lambda}) = \delta(\tilde{\lambda} - \lambda)$, where $\delta(x)$ is an impulse function (Dirac's delta function). This corresponds to the case of homogeneous mobility, considered earlier, with average meeting rates for all nodes equal to λ. The laplace distribution of $f(\tilde{\lambda})$ is then equal to $F_{\mathcal{L}}(R_i) = \exp(-\lambda R_i)$. Replacing this in Eq.(9), the generic marginal utility, gives us Eq.(1), the utility for homogeneous mobility, as expected.

Exponential distribution: Let $f(\tilde{\lambda}) = \lambda \exp(-\tilde{\lambda}\lambda_0)$, for $\tilde{\lambda} \geq 0$. This corresponds to a mobility model, where individual rates between pairs differ, but the variance of these rates is not high and their average is equal to λ_0. The laplace transform of $f(\tilde{\lambda})$ is

$$F_{\mathcal{L}}(R_i) = \frac{1}{(R_i + \lambda_0)^2}.$$

Replacing this in Eq.(9) gives us the marginal utility per message that should be used:

$$(1 - \frac{m_i}{L-1}) * \ln(\frac{1}{(R_i + \lambda_0)^2}) * \frac{1}{(R_i + \lambda_0)^{2n_i}}. \quad (10)$$

Unknown distribution in large networks: If the actual probability distribution of meeting rates is not known, the following approximation could be made in order to derive marginal utilities per message and use them for buffer management. Let us assume that the meeting rates come from an unknown distribution with first and second moments $\bar{\lambda}$ and σ^2, respectively. Let us further assume that there is a large number of nodes, such that n_i, the number of copies of message i at steady state, is large. Using the central limit theorem, we have:

$$Prob(\sum_{j=1}^{n_i} \tilde{\lambda_j} \leq \lambda) \underset{n_i \to \infty}{\sim} \mathcal{N}(n_i\bar{\lambda}, \sigma\sqrt{n_i}), \quad (11)$$

that is, the sum of meeting rates with the destination of the n_i relays for message i is approximately (normally) distributed. Replacing this in Eq.(8), we get the (unconditional) delivery probability P_i

$$P_i = (1 - \frac{m_i}{L-1}) * F_{\mathcal{L}}(R_i) + \frac{m_i}{L-1},$$

where $F_{\mathcal{L}}(R_i)$ is the Laplace transform of the above normal distribution[5]. After some algebraic manipulations we get that

$$F_{\mathcal{L}}(R_i) = \frac{\exp(\frac{n_i(\bar{\lambda})^2}{\sigma^2} + \frac{R_i^2}{4})}{\sqrt{8n_i}\sigma^2} \text{erfc}(\frac{R_i}{2}),$$

$erfc(x)$ is the complementary error function.

Differentiating with respect to n_i gives us the new marginal utility for message i:

$$(1 - \frac{m_i}{L-1}) * \frac{(\bar{\lambda}^2\sqrt{8}(n_i)^{-\frac{1}{2}} + \sqrt{2}\sigma^2(n_i)^{-\frac{5}{2}}) \exp(n_i\frac{\bar{\lambda}^2}{\sigma^2} + \frac{R_i^2}{4})}{8\sigma^4} * \text{erfc}(\frac{R_i}{2}). \quad (12)$$

In a large enough network, even if the actual distribution of meeting rates is not known, a node could still derive good utility approximations, by measuring and maintaining an estimate for the first and second moments of observed or reported meeting rates. Due to space limitations, we do not look further into the issue of this estimation, but we note that techniques similar to the ones discussed in the next Section could be used. Furthermore, additional complexity in the mobility model (e.g. correlated meeting rates) could still be handled in our framework. However, we believe that such complexity comes at the expense of ease of interpretation (and thus usefulness) of the respective utilities. We will therefore consider the simple case of homogeneous mobility for the remainder of our discussion, in order to better elucidate some additional key issues related to buffer management in DTNs, and resort to a simulation-based validation under realistic mobility patterns.

3.5 Optimality of Gradient Ascent Policy

We finally turn our attention back to the distributed (*local*) buffer management policies of Sections 3.2 and 3.3, in order to further investigate their optimality. Let us observe our network at a random time instant, and assume there are K total undelivered messages, with remaining Times To Live $R_1, R_2, \ldots, R_K$, respectively. The centralized version of our buffer management problem then consists of assigning the available buffer space across the network (L nodes each able to store B message copies) among the copies of these messages, $n_1, n_2, \ldots, n_K$, so as to maximize the expected delivery probability for

5. Note that the Laplace transform is not raised anymore to the n_i^{th} power, as the distribution already corresponds to the sum of all rates.

all these messages (where the expectation is taken over mobility decisions of all nodes). This corresponds to the following optimization problem:

$$\max_{n_1,n_2,\ldots,n_K} \sum_{i=1}^{K} (1 - \exp(-\lambda n_i R_i) \quad (13)$$

$$\sum_{i=1}^{K} n_i - LB \leq 0 \quad (14)$$

$$n_i - L \leq 0, \forall i \quad (15)$$

$$n_i \geq 1, \forall i \quad (16)$$

This is a constrained optimization problem, with K variables and $2K + 1$ inequality constraints. The optimization function in Eq.(13) is a concave function in n_i. Constraint in Eq.(14) says that the total number of copies (for all messages) should not exceed the available buffer space in all L nodes, and is linear. Finally, the $2K$ constraints of Eq.(15) are also linear, and simply say that there is no point for any node to store two copies of the same message. Consequently, if we assume that n_i are real random variables (rather than integers), this is a *convex optimization* problem, which can be solved efficiently [22] (but not easily analytically).

Having found an optimal vector $\mathbf{n}$, a centralized optimal algorithm can easily assign the copies to different nodes (e.g. picking nodes sequentially and filling their buffers up with any non-duplicate copy, starting from the messages with highest assigned n_i — due to uniform mobility the choice of specific nodes does not matter). It is important to note that, given this assignment, *no further message replication or drop is needed. This is the optimal resource allocation averaged over all possible future node movements.* The optimal algorithm must perform the same process at every subsequent time step in order to account for new messages, messages delivered, and the smaller remaining times of undelivered messages.

Our local policies offer a *distributed implementation of a gradient ascent algorithm for this problem*. Gradient ascent algorithms look at the current state, i.e. vector $\mathbf{n}(k)$ at step k, and choose a neighboring vector $\mathbf{n}(k+1)$ that improves the optimization function in Eq.(13), and provably converge to the optimal solution [22]. In our case, a step corresponds to a *contact* between two nodes, and the neighboring states and permitted transitions depend on the messages in the buffers of the two nodes in contact. In other words, our gradient ascent algorithm is supposed to make enough steps to converge to the optimal copy vector $\mathbf{n}^*$, before the state of the network (i.e. number and ID of messages) changes enough for the optimal assignment to change significantly. This depends on the rate of update steps ($\approx \lambda L^2$) and the message TTL. If $TTL * \lambda * L^2 \gg 1$, then we expect the distributed, local policy to be able to closely follow the optimal solution at any time t. In Section 5.4, we use simulation to prove that this is indeed the case for the scenarios considered.

4 Using Network History to Approximate Global Knowledge in Practice

It is clear from the above description that the optimal policy (GBSD) requires global information about the network and the "spread" of messages, in order to optimize a specific routing metric. In particular, for each message present in a node's buffer, we need to know the values of $m_i(T_i)$ and $n_i(T_i)$. In related work [11], it has been suggested that this global view could be obtained through a secondary, "instantaneous" channel, if available, or by flooding ("in-band") all necessary meta-data. Regarding the former option, cellular network connections are known to be low bandwidth and high cost in terms of power and actual monetary cost per bit. In networks of more than a few nodes, the amount of signalling data might make this option prohibitive. Concerning flooding, our experiments show that the impact of the flooding delay on the performance of the algorithm is not negligible. In practice, intermittent network connectivity and the long time it takes to flood buffer status information across DTN nodes, make this approach inefficient.

A different, more robust approach is to find estimators for the unknown quantities involved in the calculation of message utilities, namely m and n. We do this by designing and implementing a learning process that permits a DTN node to gather knowledge about the global network state at different times in the past, by making in-band exchanges with other nodes. Each node maintains a list of encountered nodes and the state of each message carried by them as a function of time. Specifically, it logs whether a given message was present at a given time T in a node's buffer (counting towards n) or whether it was encountered earlier but is not anymore stored, e.g. it was dropped (counting towards m). In Section 6, we describe our statistics maintenance and collection method, in more detail, along with various optimizations to considerably reduce the signalling overhead.

Since global information gathered thus about a specific message might take a long time to propagate and hence might be obsolete when we calculate the utility of the message, we follow a different route. Rather than looking for the current value of $m_i(T)$ and $n_i(T)$ *for a specific message* i at an elapsed time T, we look at what happens, *on average, for all messages after an elapsed time* T. In other words, the $m_i(T)$ and $n_i(T)$ values for message i at elapsed time T are estimated using measurements of m and n for the same elapsed time T but *measured for (and averaged over) all other older messages*. These estimations are then used in the evaluation of the per-message utility.

Let's denote by $\hat{n}(T)$ and $\hat{m}(T)$ the estimators for $n_i(T)$ and $m_i(T)$ of message i. For the purpose of the analysis, we suppose that the variables $m_i(T)$ and $n_i(T)$ at elapsed time T are instances of the random variables $N(T)$ and $M(T)$. We develop our estimators $\hat{n}(T)$ and $\hat{m}(T)$ so that when plugged into the GBSD's delivery rate and delay per-message utilities calculated in Section 3,

we get two new per-message utilities that can be used by a DTN node without any need for global information about messages. This results in a new scheduling and drop policy, called HBSD (History Based Scheduling and Drop), a deployable variant of GBSD that uses the same algorithm, yet with per-message utility values calculated using estimates of m and n.

4.1 Estimators for the Delivery Rate Utility

When global information is unavailable, one can calculate the average delivery rate of a message over all possible values of $M(T)$ and $N(T)$, and then try to maximize it. In the framework of the GBSD policy, this is equivalent to choosing the estimators $\hat{n}\ (T)$ and $\hat{m}\ (T)$ so that the calculation of the average delivery rate is unbiased:

$$E[(1-\frac{M(T)}{L-1})*(1-\exp(-\lambda N(T)R_i))+\frac{M(T)}{L-1}] = (1-\frac{\hat{m}\ (T)}{L-1})*(1-\exp(-\lambda\ \hat{n}\ (T)R_i))+\frac{\hat{m}\ (T)}{L-1}$$

Plugging any values for $\hat{n}\ (T)$ and $\hat{m}\ (T)$ that verify this equality into the expression for the per-message utility of Eq.(1), one can make sure that the obtained policy maximizes the average delivery rate. This is exactly our purpose. Suppose now that the best estimator for $\hat{m}\ (T)$ is its average, i.e., $\hat{m}\ (T) = \bar{m}\ (T) = E[M(T)]$. This approximation is driven by the observation we made that the histogram of the random variable $M(T)$ can be approximated by a Gaussian distribution with good accuracy. To confirm this, we have applied the Lillie test [23], a robust version of the well known Kolmogorov-Smirnov goodness-of-fit test, to $M(T)$ for different elapsed times (T = 25%,50% and 75% of the TTL). This test led to acceptance for a 5% significance level. Consequently, the average of $M(T)$ is at the same time the unbiased estimator and the most frequent value among the vector $M(T)$. Then, solving for $\hat{n}\ (T)$ gives:

$$\hat{n}\ (T) = -\frac{1}{\lambda R_i}\ln(\frac{E[(1-\frac{M(T)}{L-1})\exp(-\lambda N(T)R_i)]}{(1-\frac{\bar{m}(T)}{L-1})}) \quad (17)$$

Substituting this expression into Eq.(1) we obtain the following new per message utility for our approximating HBSD policy:

$$\lambda R_i E[(1-\frac{M(T)}{L-1})\exp(-\lambda R_i N(T))] \quad (18)$$

The expectation in this expression is calculated by summing over all known values of $N(T)$ and $M(T)$ for past messages at elapsed time T. Unlike Eq.(1), this new per-message utility is a function of past history of messages and can be calculated locally. It maximizes the average message delivery rate calculated over a large number of messages. When the number of messages is large enough for the law of large numbers to work, our history based policy should give the same result as that of using the real global network information. Finally, we note that L, the number of nodes in the network, could also be calculated from the statistics maintained by each node in the network. In this work, we assume it to be fixed and known, but one could estimate it similar to n and m, or using different estimation algorithms like the ones proposed in [24].

4.2 Estimators for the Delivery Delay Utility

Similar to the case of delivery rate, we calculate the estimators $\hat{n}\ (T)$ and $\hat{m}\ (T)$ in such a way that the average delay is not affected by the estimation. This gives the following per-message utility specific to HBSD,

$$\frac{E[\frac{L-1-M(T)}{N(T)}]^2}{\lambda(L-1)(L-1-\bar{m}\ (T))} \quad (19)$$

This new per-message utility is only a function of the locally available history of old messages and is thus independent of the actual global network state. For large number of messages, it should lead to the same average delay as when the exact values for m and n are used.

5 Performance Evaluation

5.1 Experimental Setup

To evaluate our policies, we have implemented a DTN framework into the Network Simulator NS-2 [25]. This implementation includes (i) the Epidemic routing protocol with *FIFO* for scheduling messages queued during a contact and *drop-tail* for message drop, (ii) the RAPID routing protocol based on flooding (i.e. no side-channel) as described, to our best understanding, in [11], (iii) a new version of Epidemic routing enhanced with our optimal joint scheduling and drop policy (GBSD), (iv) another version using our statistical learning based distributed algorithm (HBSD), and (v) the VACCINE anti-packet mechanism described in [13][6].

In our simulations, each node uses the 802.11b protocol to communicate, with rate 11Mbits/sec. The transmission range is 100 meters, to obtain network scenarios that are neither fully connected (e.g. MANET) nor extremely sparse. Our simulations are based on five mobility scenarios: two synthetic mobility models and three real-world mobility traces.

Synthetic Mobility Models: We've considered both the Random Waypoint mobility model and the HCMM model [26]. The later is inspired from Watts' Caveman model that was shown to reproduce statistics of human mobility, such as inter-contact times and contact duration.

6. We have also performed simulations without any anti-packet mechanism, from which similar conclusions can be drawn.

Real Mobility Traces: The first *(i)* real trace is the one collected as part of the ZebraNet wildlife tracking experiment in Kenya and described in [27]. The second *(ii)* mobility trace tracks San Francisco's Yellow Cab taxis [28] and the third *(iii)* trace consists on the KAIST real mobility trace collected from a university campus (KAIST) in South Korea [29]. We consider a sample of the KAIST campus trace taken from 50 students, where the GPS receivers log their position at every 30 seconds.

To each source node, we have associated a CBR (Constant Bit Rate) application, which chooses randomly from [0, TTL] the time to start generating messages of 5KB for a randomly chosen destination. We have also considered other message sizes (see e.g. [15]), but found no significant differences in the qualitative and quantitative conclusions drawn regarding the relative performance of different schemes[7]. Unless otherwise stated, each node maintains a buffer with a capacity of 20 messages to be able to push the network towards a congested state without exceeding the processing and memory capabilities of our simulation cluster. We compare the performance of the various routing protocols using the following two metrics: the average delivery rate and average delivery delay of messages in the case of infinite TTL[8]. Finally, the results presented here are averages from 20 simulation runs, which we found enough to ensure convergence.

5.2 Performance evaluation for delivery rate

First, we compare the delivery rate of all policies for the three scenarios shown in Table 2.

TABLE 2
Simulation parameters

Mobility pattern:	RWP	ZebraNet	Taxis	KAIST	HCMM
Sim. Duration(h):	7	14	42	24	24
Sim. Area (km^2):	3*3	3*3	-	-	5*5
Nbr. of Nodes:	70	70	70	50	70
Avg. Speed (m/s):	2	-	-	-	-
$TTL(h)$:	1	2	6	4	4
CBR Interval(s):	360	720	2160	1440	1440

TABLE 3
Taxi Trace (70 CBR sources)

Policy:	GBSD	HBSD	RAPID	FIFO\DT
D. Probability:	0.72	0.66	0.44	0.34
D. Delay(s):	14244	15683	20915	36412

Figure 2 shows the delivery rate based on the random waypoint model. From this plot, it can be seen that: the GBSD policy plugged into Epidemic routing gives the best performance for all numbers of sources. When congestion-level decreases, so does the difference between GBSD and other protocols, as expected. Moreover, the HBSD policy also outperforms existing protocols (RAPID and Epidemic based on FIFO/drop-tail) and performs very close to the optimal GBSD. Specifically, for 70 sources, HBSD offers an almost 60% improvement in delivery rate compared to RAPID and is only 14% worse than GBSD. Similar conclusions can be also drawn for the case of the real Taxi trace, ZebraNet trace, KAIST trace or the HCMM model and 70 sources. Results for these cases are respectively summarized in Table 3, Table 4, Figure 3 and Table 5.

7. In future work, we intend to evaluate the effect of variable message size and its implications for our optimization framework. In general, utility-based scheduling problems with variable sized messages can often be mapped to Knapsack problems (see e.g. [30]).

8. By infinite TTL, we mean any value large enough to ensure almost all messages get delivered to their destination before the TTL expires.

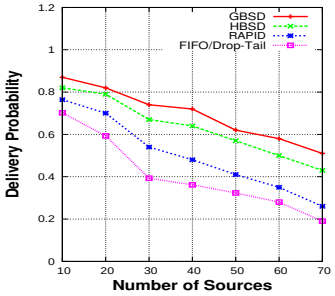

Fig. 2. Delivery Probability (R.W. mobility model).

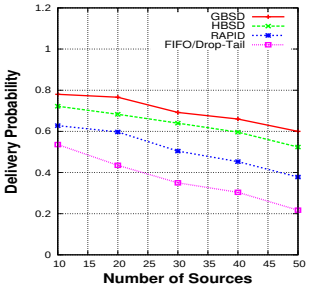

Fig. 3. Delivery Probability (KAIST mobility trace).

TABLE 4
ZebraNet Trace (70 CBR sources)

Policy:	GBSD	HBSD	RAPID	FIFO\DT
D. Probability:	0.68	0.59	0.41	0.29
D. Delay(s):	4306	4612	6705	8819

TABLE 5
HCMM Trace (70 CBR sources)

Policy:	GBSD	HBSD	RAPID	FIFO\DT
D. Probability:	0.62	0.55	0.38	0.23
D. Delay(s):	3920	4500	6650	8350

5.3 Performance evaluation for delivery delay

To study delays, we increase messages' TTL (and simulation duration), to ensure almost every message gets delivered. For the random waypoint mobility scenario, Figure 4 depicts the average delivery delay for the case of both limited buffer and bandwidth. As in the case of delivery rate, GBSD gives the best performance for all considered scenarios. Moreover, the HBSD policy outperforms the two routing protocols (Epidemic based on FIFO/drop-tail, and RAPID) and performs close to GBSD. Specifically, for 70 sources and both limited buffer and bandwidth, HBSD average delivery delay is 48% better than RAPID and only 9% worse than GBSD.

Table 3, Table 4, Figure 5 and Table 5 show that similar conclusions can be drawn for the delay under respectively the real Taxi(s), ZebraNet trace, KAIST trace and the HCMM model.

5.4 Optimality

Here, we show throught simulations results (based on the RW scenario 5.2) that our proposed policy (GBSD)

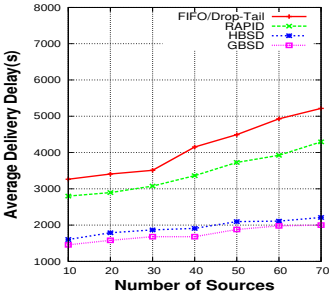

Fig. 4. Delivery Delay (R.W. mobility model).

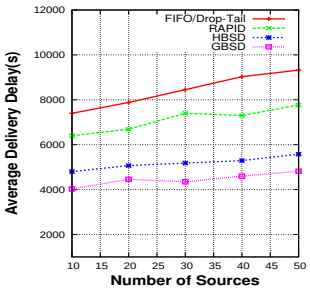

Fig. 5. Delivery Delay (KAIST mobility trace).

can keep up with the optimal algorithm described in Section 3.5. Indeed, Figure 6 plots the normalized Manhattan distance $d(X,Y) = \frac{\sum_{i=1}^{K}|x_i - y_j|}{K}$ betwenn two consecutive N vectors resulting from solving the optimal centralized version offline and shows that the later distance is very small which means that, when nodes meet frequently enough during the lifetime of messages, our distributed version (HBSD) has enough time to track the optimal behavior of the network. We belive the later result sufficiently justifies the claim to optimality with respect to a distributed implementation in this context. In addition to that, we compare throught Figure 7 the absolute difference between the number of copies of a given message while first applying GBSD during a simulation and second solving solving offline the optimal algorithm 3.5. We can see from the later result that for a randomly choosed set of messages, the difference in terms of number of copies is small and equal to 0 at some points. The later result comes to consolidate the optimality properties of our proposed policy.

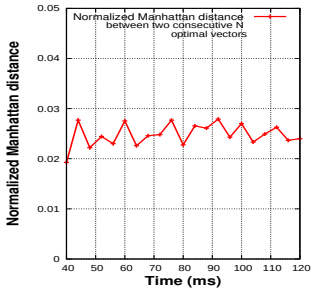

Fig. 6. Normalized Manhattan distance between two consecutive N optimal vectors.

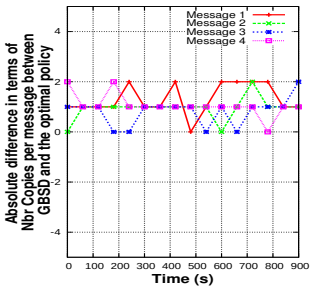

Fig. 7. Difference in terms of Nbr of copies.

6 Maintaining Network History

The results of the previous section clearly show that our distributed policy (HBSD) that uses estimators of global message state successfully approximates the performance of the optimal policy (GBSD). This is as an important step towards a practical implementation of efficient buffer management and scheduling algorithms on wireless devices. Nevertheless, in order to derive good estimators in a distributed manner, nodes need to exchange (a possibly large amount of) metadata during every node meeting. Potentially, each node needs to know the history of all messages having passed through a node's buffer, for every node in the network. In a small network, the amount of such "control" data might not be much, considering that large amounts of data transfers can be achieved between 802.11 transceivers during short contacts. Nevertheless, in larger networks, this method can quickly become unsalable and interfere with data transmissions, if statistics maintenance and collection is naively done.

In this section, we describe the type of statistics each node maintains towards calculating the HBSD utility for each message, and propose a number of mechanisms and optimizations to significantly reduce the amount of metadata exchanged during contacts. Finally, we explore the impact of reducing the amount of collected statistics on the performance of our buffer management and scheduling policy.

6.1 Maintaining Buffer State History

In order to keep track of the statistics about past messages necessary to take the appropriate transmission or dropping decision, we propose that each node maintains the data structure depicted in Figure 8. Each node maintains a list of messages whose history in the network it keeps track of. For each message, it maintains its ID, its TTL and the list of nodes that have *seen* it before. Then, for each of the nodes in the list, it maintains a data structure with the following data: (i) the node's ID, (ii) a boolean array $Copies\_Bin\_Array$, and (iii) the version $Stat\_Version$ associated to this array.

The $Copies\_Bin\_Array$ array (Fig. 9) enables nodes to maintain what each message experienced during its life time. For a given entry pair (message a and node b) in this list, the $Copies\_Bin\_Array[k]$ indicates if the node a had already stored or not a copy of message b in its buffer during Bin k. In other words, time is quantized into "bins" of size Bin_Size, and bin k correspond to the period of time between $k * Bin\_Size$ and $(k+1) * Bin\_Size$. As a result, the size of the $Copies\_Bin\_Array$ is equal to TTL/Bin_Size.

How should one choose Bin_Size? Clearly, the larger it is, the fewer the amount of data a node needs to maintain and to exchange during each meeting; however, the smaller is also the granularity of values the utility function can take and thus the higher the probability of an incorrect decision. As already described in Section 3, message transmissions can occur only when nodes encounter each other. This is also the time granularity at which buffer state changes occur. Hence, we believe that a good trade-off is to monitor the evolution of each message's state at a bin granularity in the order

of meeting times[9]. This *results in a big reduction of the size of statistics to maintain locally* (as opposed to tracking messages at seconds or milliseconds granularity), while still enabling us to infer the correct messages statistics.

Finally, the $Stat\_Version$ indicates the Bin at which the last update occurred. When the TTL for message a elapses, b sets the $Stat\_Version$ to TTL/Bin_Size, which also indicates that all information about the history of *this* message in *this* buffer is now available. The combination of how the $Copies\_Bin\_Array$ is maintained and the $Stat\_Version$ updated, ensures that only the minimum amount of necessary metadata *for this pair of (message, node)* is exchanged during a contact.

We note also that, in principle, a $Message\_Seen\_Bin\_Array$ could be maintained, indicating if a node a had *seen* (rather than *stored* a message b at time t, in order to estimate $m(T)$. However, it is easy to see that the $Message\_Seen\_Bin\_Array$ can be deduced directly from the $Copies\_Bin\_Array$, and thus no extra storage is required. Summarizing, based on this lists maintained by all nodes, any node can retrieve the vectors $N(T)$ and $M(T)$ and can calculate the HBSD per-message utilities described in Section 4 without a need for an *oracle*.

6.2 Collecting Network Statistics

We have seen so far what types of statistics each node maintains about each past (message ID, node ID) tuple it knows about. Each node is supposed to keep up-to-date the statistics related to the messages it stores locally. However, it can only update its knowledge about the state of a message a at a node b when it either meets b directly, or it meets a node that has more recent information about the (a, b) tuple. The goal of the statistics collection method is that, through such message exchanges, nodes converge to a unified view about the state of a given message at *any* buffer in the network, during its lifetime.

Sampling Messages to Keep Track of: We now look in more detail into what kind of metadata nodes should exchange. The first interesting question is: *should a node maintain global statistics for* every *message it has heard of or only a subset?* We argue that monitoring a dynamic subset of these messages is sufficient to quickly converge to the correct expectations we need for our utility estimators. This dynamic subset is illustrated in Figure 10 as being the Messages Under Monitoring, which are stored in the *MUM* buffer; it is dynamic because its size is kept fixed while messages inside it change. When a node decides to store a message for the first time, if there is space in its MUM buffer, it also inserts it there and will track its global state. The actual *sampling rate* depends on the size of the MUM buffer and the offered traffic load, and

9. According to the *Nyquist-Shannon* [31] sampling theorem, a good approximation of the size of a *Bin* would be equal to inter-meeting-time/2. A running average of the observed times between consecutive meetings could be maintained easily, in order to dynamically adjust the bin size [7].

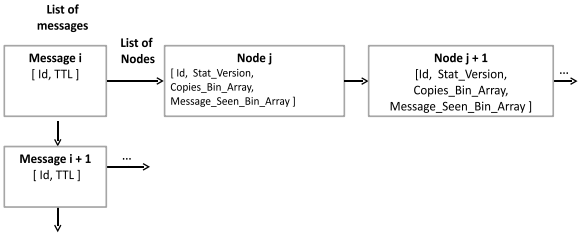

Fig. 8. Network History Data Structure

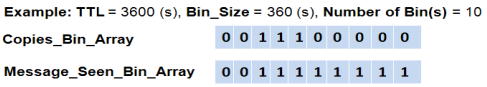

Fig. 9. Example of Bin arrays

results in significant further reduction in the amount of metadata exchanged. At the same time, a smaller MUM buffer might result to slower convergence (or even lack of). In Section 6.3 we study the impact of MUM buffer size on the performance of our algorithm.

Handling Converged Messages: Once the node collects an entire history of a given message, it removes it from the *MUM* buffer and pushes it to the buffer of Messages with a Complete History (*MCH*). A node considers that it has the complete history of a given message only when it gets the last version of the statistics entries related to all the nodes the message goes through during its TTL[10] Finally, note that, once a node decides to move a message to the *MCH* buffer, it only needs to maintain a short summary rather than the per node state as in Fig. 8.

Statistics Exchanged: Once a contact opportunity is present, both peers have to ask only for newer versions of the statistics entries (message ID, node ID) related to the set of messages buffered in their *MUM* buffer. This ensures that, even for the sampled set of messages, only new information is exchanged and no bandwidth is wasted while not introducing any extra latency in the convergence of our approximation scheme.

6.3 Performance Tradeoffs of Statistics Collection

We have presented a number of optimizations to reduce the amount of stored metadata and the amount of sig-

10. Note that there is a chance that a node might "miss" some information about a message it pushes in its MCH. This probability depends on the statistics of the meeting time (first and second moment) and the TTL value. Nevertheless, for many scenarios of interest, this probability is small and it may only lead to slightly underestimating the m and n values.

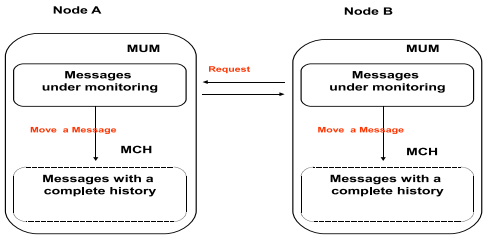

Fig. 10. Statistics Exchange and Maintenance.

nalling overhead. Here, we explore the trade-off between the signalling overhead, its impact on performance, and the dynamicity of a given scenario. Our goal is to identify operation points where the amount of signalling overhead is such that it interferes minimally with data transmission, while at the same time it suffices to ensure timely convergence of the required utility metrics per message. We will consider throughout the random waypoint scenario described in Section 5.2. We have observed similar behaviour for the trace-based scenarios.

Amount of Signalling Overhead per Contact: We start by studying the effect of varying the size of the *MUM* buffer on the average size of exchanged statistics per-meeting. Figure 11 compares the average size of statistics exchanged during a meeting between two nodes for three different sizes of the *MUM* buffer, as well as for the basic epidemic statistics exchange method (i.e. unlimited MUM). We vary the number of sources in order to cover different congestions regimes.

Our first observation is that increasing the traffic load results in decreasing the average amount of statistics exchanged per-meeting (except for the MUM size of 20 messages). This might be slightly counterintuitive, since a higher traffic load implies more messages to keep track of. However, note that a higher congestion level also implies that much fewer copies per message will coexist at any time (and new versions are less frequently created). As a result, much less metadata per message is maintained and exchanged, resulting in a downward trend. In the case of a MUM size of 20, it seems that these two effects balance each other out. In any case, the key property here is that, in contrast with the flooding-based method of [11], *our distributed collection method scales well, not increasing the amount of signalling overhead during high congestion.*

A second observation is that, using our statistics collection method, a node can reduce the amount of signalling overhead per meeting up to an order of magnitude, compared to the unlimited MUM case, even in this relatively small scenario of 70 nodes.

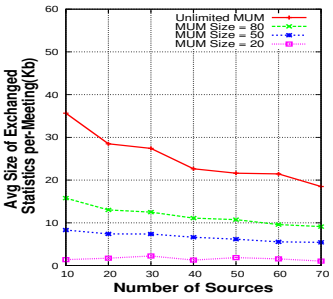

Fig. 11. Signalling overhead (per contact) resulting from HBSD statistics collection.

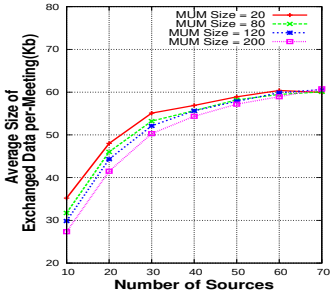

Fig. 12. Average size of exchanged (non-signalling) data per contact.

Finally, we plot in Figure 12 the average size of exchanged (non-signalling) data per-meeting. We can observe that increasing the size of the *MUM* buffer results in a slight decrease of the data exchanged. This is due to the priority we give to statistics exchange during a contact. We note also that this effect becomes less pronounced when congestion increases (in line with Fig. 11). Finally, in the scenario considered, we can observe that, for MUM sizes less than 50, signalling does not interfere with data transmissions (remember that packet size is 5KB). This suggests that, in this scenario, a MUM size of 50 messages represents a good choice with respect to the resulting signalling overhead. In practice, a node could find this value online, by dynamically adjusting its MUM size and comparing the resulting signalling overhead with average data transfer. It is beyond the scope of this paper to propose such an algorithm. Instead, we are interested in exposing the various tradeoffs and choices involved in efficient distributed estimation of statistics. Towards this goal, we explore next the effect of the MUM sizes considered on the performance of our HBSD algorithm.

Convergence of Utilities and Performance of the HBSD Policy : In this last part, we fix the number of sources to 50 and we look at the impact of the size of the *MUM* buffer on (i) the time it takes the HBSD delivery rate utility to converge, and (ii) its accuracy. We use the *mean relative square error* to measure the accuracy of the HBSD delivery rate utility, defined as follows:

$$\frac{1}{\#Bins} * \sum_{Bins} \frac{(A-B)^2}{B^2},$$

where, for each *bin*, A is the estimated utility value of Eq. (18) (calculated using the approximate values of m and n, collected with the method described previously) and B is the utility value calculated using the real values of m and n.

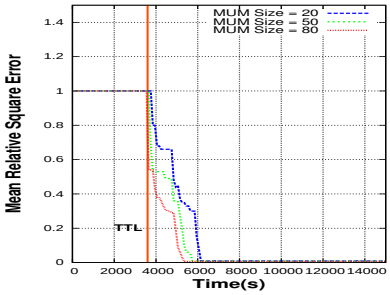

Fig. 13. Mean relative square errors for HBSD delivery rate utility.

Figure 13 plots the *mean relative square errors* for the HBSD delivery rate utility, as a function of time. We can observe that, increasing the size of the *MUM* buffer results in faster reduction of the *mean relative square error* function. With a *MUM* buffer of 80 messages, the delivery rate utility estimate converges 800 seconds faster than using an *MUM* buffer of 20 messages. Indeed, the more messages a node tracks in parallel, the faster it can collect a working history of past messages that it can use to calculate utilities for new messages considered

for drop or transmission. We observe also that all plots converge to the same very small error value [11]. Note also that it is not the absolute value of the utility function that we care about, but rather the *shape* of this function, whether it is increasing or decreasing, and the relative utility values.

In fact, we are more interested in the end performance of our HBSD, as a function of how "aggressively" nodes collect message history. In Figures 14 and 15, we plot the delivery rate and delay of HBSD, respectively, for different MUM sizes. These results correspond to the scenario described in Section 5.2, where we have a fixed number of CBR sources. As is evident from these figures, regardless of the size of the *MUM* buffer sizes, nodes eventually gather enough past message history to ensure an accurate estimation of per message utilities, and a close-to-optimal performance. In such scenarios, where traffic intensity is relatively stable, even a rather small MUM size suffices to achieve good performance. This is not necessarily the case when traffic load experiences significant fluctuations.

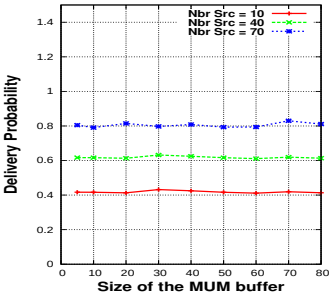

Fig. 14. Delivery Probability for HBSD with statistics collection (static traffic load).

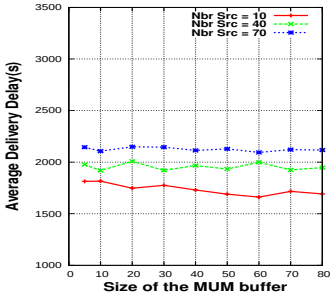

Fig. 15. Deliver Delay for HBSD with statistics collection (static traffic load).

When the offered traffic load changes frequently, convergence speed becomes important. The bigger the *MUM* buffer the faster our HBSD policy react to changing congestion levels. We illustrate this with the following experiment. We maintain the same simulation scenario, but we vary the number of CBR sources among each two consecutive TTL(s), from 10 to 70 sources (i.e. the first and second TTL window we have 10 sources, the third and fourth window 70 sources, etc. — this is close to a *worst case* scenario, as there is a sevenfold increase in traffic intensity within a time window barely higher than a TTL, which is the minimum required interval to collect any statistics). Furthermore, to ensure nodes use non-obsolete statistics towards calculating utilities, we force nodes to apply a *sliding window* of one *TTL* to the messages with complete history stored in the *MCH* buffer, and to delete messages out of this *sliding window*.

Figures 16 and 17 again plot the HBSD policy delivery rate and delay, respectively, as a function of MUM buffer size. Unlike the constant load case, it is easy to see there that, increasing the size of the *MUM* buffer, results in considerable performance improvement. Nevertheless, even in this rather dynamic scenario, nodes manage to keep up and produce good utility estimates, with only a modest increase on the amount of signalling overhead required.

11. We speculate that this remaining error might be due to slightly underestimating m and n, as explained earlier.

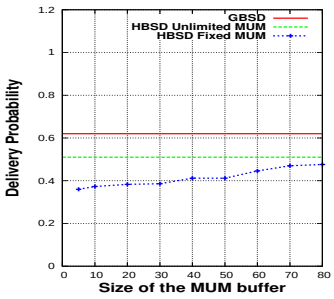

Fig. 16. Deliver Probability for HBSD with statistics collection (dynamic traffic load).

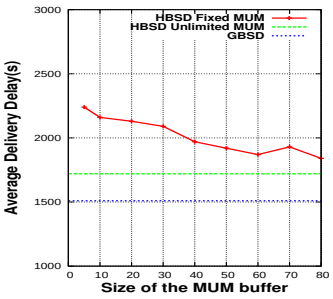

Fig. 17. Deliver Delay for HBSD with statistics collection (dynamic traffic load).

7 Distribution of HBSD Utilities

We have described how to efficiently collect the necessary statistics in practice, and derive good estimates for the HBSD utility distribution during the lifetime of a message. In this last section, we turn our attention to the utility distributions themselves. First, we are interested whether the resulting distributions for HBSD delivery rate and delivery delay utilities react differently to different congestion levels, that is, if the priority given to messages of different ages shifts based on the offered load. Furthermore, we are interested whether the resulting utility shape (and respective optimal policy) could be approximated by simple(r) policies, in some congestion regimes.

We consider again the simulation scenario used in Section 5.2 and Section 6.3. First, we fix the number of sources to 50, corresponding to a *high congestion regime*. In Figure 18 and Figure 19, we plot the distribution of the HBSD delivery rate and delivery delay utilities described in Sections 4.1 and 4.2. It is evident there that the optimal utility distribution has a non-trivial shape for both optimization metrics, resulting in a complex optimal scheduling and drop policy.

Next, we consider a scenario with low congestion. We reduce the number of sources to 15, keep the buffer size of 20 messages, but we also decrease the CBR rate of sources from 10 to 2 messages/TTL. In Figures 20 and 21, we plot the distribution of the HBSD delivery rate and delivery delay utilities, respectively, for this low congestion scenario. Surprisingly, our HBSD policy behaves very differently now, with both utility functions decaying monotonically as a function of time (albeit not at constant rate). This suggests that the optimal policy in low congestion regimes could be approximated by the simpler "Drop Oldest Message" (or schedule

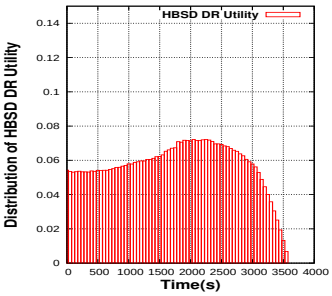

Fig. 18. Distribution of HBSD DR utility in a congested network.

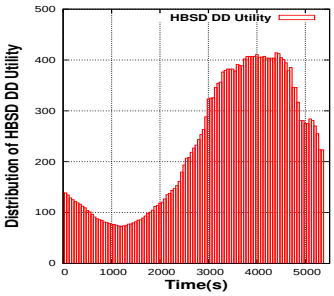

Fig. 19. Distribution of HBSD DD utility in a congested network.

younger messages first) policy, which does not require any signalling and statistics collection between nodes.

To test this, in Tables 6 and 7, we compare the performance of the HBSD policy against a simple combination of "Drop Oldest Message" (for Buffer Management) and "Transmit Youngest Message First" (for Scheduling during a contact). We observe, that in the low congestion regime, the two policies indeed have similar performance (4% and 5% difference in delivery rate and delivery delay, respectively). However, in the case of a congested network, HBSD clearly outperforms the simple policy combination.

We can look more carefully at Figures 18 and 19, to understand what is happening in high congestion regimes. The number of copies per message created at steady state depends on the total number of messages co-existing at any time instant, and the aggregate buffer capacity. When too many messages exist in the network, uniformly assigning the available messages to the existing buffers, would imply that every message can have only a few copies created. Specifically, for congestion higher than some level, the average number of copies per message allowed is so low that most messages cannot reach their destination during their TTL. *Uniformly assigning resources between nodes is no more optimal*. Instead, to ensure that at least some messages can be delivered on time, the optimal policy gives higher priority to older messages that have managed to survive long enough (and have probably created enough copies), and "kills" some of the new ones being generated. This is evident by the values assigned at different bins (especially in the delivery delay case). In other words, when congestion is excessive *our policy performs an indirect admission control function.*

Contrary to this, when the offered load is low enough to ensure that all messages can on average create enough copies to ensure delivery, the optimal policy simply performs a fair (i.e. equal) distribution of resources.

The above findings suggest that it would be quite useful to find a generic way to signal the congestion level and identify the threshold based on which nodes can decide to either activate our HBSD scheme or just use a simple Drop/Scheduling policy. Suspending a complex Drop/Scheduling mechanism and its underlying statistics collection and maintenance methods, whenever not needed, can help nodes save an important amount of resources (e.g. energy), while maintaining the same end performance. Finally, we believe that the indirect signalling provided by the behaviour of the utility function during congestion, could provide the basis for an end-to-end flow control mechanism, a problem remaining largely not addressed in the DTN context.

TABLE 6
HBSD vs. "Schedule Younger First\Drop-Oldest" in a congested network.

Policies:	HBSD	"Schedule Younger First\Drop-Oldest"
D. Rate(%):	54	29
D. Delay(s):	1967	3443

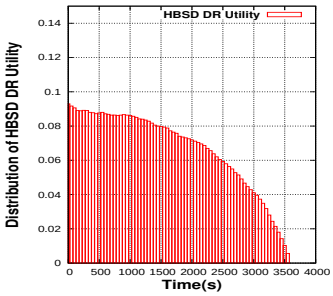

Fig. 20. Distribution of HBSD DR utility in a low congested network.

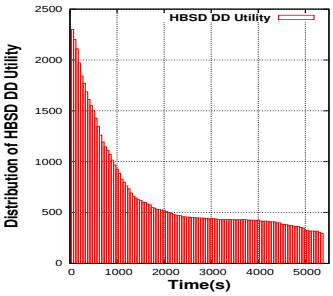

Fig. 21. Distribution of HBSD DD utility in a low congested network.

TABLE 7
HBSD vs "Schedule Younger First\Drop-Oldest" in a low congested network.

Policies:	HBSD	"Schedule Younger First\Drop-Oldest"
D. Rate(%):	87	83
D. Delay(s):	1530	1618

8 Conclusion

In this work, we investigated both the problems of scheduling and buffer management in DTNs. First, we proposed an optimal joint scheduling and buffer management policy based on global knowledge about the network state. Then, we introduced an approximation scheme for the required global knowledge of the optimal algorithm. Using simulations based on a synthetic mobility model (Random Waypoint), and real mobility traces, we showed that our policy based on statistical learning successfully approximates the performance of the optimal algorithm. Both policies (GBSD and HBSD) plugged into the Epidemic routing protocol outperform current state-of-the-art protocols like RAPID [11] with respect to both delivery rate and delivery delay, in all considered scenarios. Moreover, we discussed how to implement our HBSD policy in practice, by using a distributed statistics collection method, illustrating that our approach is realistic and effective. We showed also that unlike many works [11], [16] that also relied on the use

of an in-band control channel to propagate metadata, our statistics collection method scales well, not increasing the amount of signalling overhead during high congestion.

Finally, we carried a study of the distributions of HBSD' utilities under different congestion levels and we showed that: when congestion is excessive, HBSD performs an indirect admission control function and has a non-trivial shape for both optimization metrics, resulting in a complex optimal scheduling and drop policy. However, when the offered load is low enough, HBSD can be approximated by a simple policy that does not require any signalling and statistics collection between nodes. The above findings suggest that it would be quite useful to find a generic way to signal the congestion level and identify the threshold based on which nodes can decide to either activate our HBSD scheme or just use a simple Drop/Scheduling policy. Suspending a complex Drop/Scheduling, whenever not needed, can help nodes save an important amount of resources, while maintaining the same end performance.

REFERENCES

[1] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *Proceedings of ACM SIGCOMM*, Aug. 2004.

[2] S. Jain, M. Demmer, R. Patra, and K. Fall, "Using redundancy to cope with failures in a delay tolerant network," in *Proceedings of ACM SIGCOMM*, 2005.

[3] N. Glance, D. Snowdon, and J.-L. Meunier, "Pollen: using people as a communication medium," *Computer Networks*, vol. 35, no. 4, Mar. 2001.

[4] "Delay tolerant networking research group," *http://www.dtnrg.org*.

[5] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke University, Tech. Rep. CS-200006, 2000.

[6] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mobile Computing and Communication Review*, vol. 7, no. 3, 2003.

[7] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Efficient routing in intermittently connected mobile networks: The multiple-copy case," *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 77–90, 2008.

[8] Z. J. Haas and T. Small, "A new networking model for biological applications of ad hoc sensor networks." *IEEE/ACM Transactions on Networking*, vol. 14, no. 1, pp. 27–40, 2006.

[9] R.Groenevelt, G. Koole, and P. Nain, "Message delay in manet (extended abstract)," in *Proc. ACM Sigmetrics*, 2005.

[10] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Performance analysis of mobility-assisted routing," in *Proceedings of ACM/IEEE MOBIHOC*, 2006.

[11] A. Balasubramanian, B. Levine, and A. Venkataramani, "Dtn routing as a resource allocation problem," in *Proceedings of ACM SIGCOMM*, 2007.

[12] A. Lindgren and K. S. Phanse, "Evaluation of queuing policies and forwarding strategies for routing in intermittently connected networks," in *Proceedings of IEEE COMSWARE*, 2006.

[13] X. Zhang, G. Neglia, J. Kurose, and D. Towsley, "Performance modeling of epidemic routing," in *Proceedings of IFIP Networking*, 2006.

[14] H. P. Dohyung Kim and I. Yeom, "Minimizing the impact of buffer overflow in dtn," in *Proceedings International Conference on Future Internet Technologies (CFI)*, 2008.

[15] A. Krifa, C. Barakat, and T. Spyropoulos, "Optimal buffer management policies for delay tolerant networks," in *IEEE SECON*, 2008.

[16] D. J. L. S. L. Z. Yong L., Mengjiong Q., "Adaptive optimal buffer management policies for realistic dtn," in *IEEE GLOBECOM*, 2009.

[17] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks," in *Proc. IEEE INFOCOM*, 2006.

[18] T. Spyropoulos, T. Turletti, and K. Obrazcka, "Routing in delay tolerant networks comprising heterogeneous populations of nodes," *IEEE Transactions on Mobile Computing*, 2009.

[19] D. Aldous and J. Fill, "Reversible markov chains and random walks on graphs. (monograph in preparation.)," *http://stat-www.berkeley.edu/users/aldous/RWG/book.html*.

[20] T. Karagiannis, J.-Y. Le Boudec, and M. Vojnović, "Power law and exponential decay of inter contact times between mobile devices," in *Proceedings of ACM MobiCom '07*, 2007.

[21] A. Chaintreau, J.-Y. Le Boudec, and N. Ristanovic, "The age of gossip: spatial mean field regime," in *Proceedings of ACM SIGMETRICS '09*, 2009.

[22] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press New York, NY, USA, 2004.

[23] H. Lilliefors, "On the kolmogorov-smirnov test for normality with mean and variance unknown," *Journal of the American Statistical Association, Vol. 62. pp. 399-402*, 1967.

[24] A. Guerrieri, A. Montresor, I. Carreras, F. D. Pellegrini, and D. Miorandi, "Distributed estimation of global parameters in delay-tolerant networks," in *in Proccedings of Autonomic and Opportunistic Communication (AOC) Workshop (colocated with WOWMOM*, 2009, pp. 1–7.

[25] DTN Architecture for NS-2. [Online]. Available: http://www-sop.inria.fr/members/Amir.Krifa/DTN

[26] C. Boldrini, M. Conti, and A. Passarella, "Users mobility models for opportunistic networks: the role of physical locations," in *Proc. of IEEE WRECOM*, 2007.

[27] Y. Wang, P. Zhang, T. Liu, C. Sadler, and M. Martonosi, "Movement data traces from princeton zebranet deployments," CRAWDAD Database. http://crawdad.cs.dartmouth.edu/, 2007.

[28] Cabspotting Project. [Online]. Available: http://cabspotting.org/

[29] "KAIST mobility traces," *http://research.csc.ncsu.edu/netsrv/?q=node/4*.

[30] C. Boldrini, M. Conti, and A. Passarella, "Contentplace: social-aware data dissemination in opportunistic networks," in *Proceedings of ACM MSWiM*, 2008.

[31] Nyquist Shannon sampling theorem. [Online]. Available: