Optimal Buffer Management Policies for Delay Tolerant Networks

Amir Krifa*†, Chadi Barakat†, Thrasyvoulos Spyropoulos†‡
†Project-Team Planète, INRIA Sophia-Antipolis, France
*National School of Computer Sciences (ENSI), Tunisia
‡Swiss Federal Institute of Technology (ETH), Zurich, Switzerland
Emails: {Amir.Krifa, Chadi.Barakat}@sophia.inria.fr, spyropoulos@tik.ee.ethz.ch

***Abstract*—Delay Tolerant Networks are wireless networks where disconnections may occur frequently due to propagation phenomena, node mobility, and power outages. Propagation delays may also be long due to the operational environment (e.g. deep space, underwater). In order to achieve data delivery in such challenging networking environments, researchers have proposed the use of *store-carry-and-forward* protocols: there, a node may store a message in its buffer and carry it along for long periods of time, until an appropriate forwarding opportunity arises. Additionally, multiple message replicas are often propagated to increase delivery probability. This combination of long-term storage and replication imposes a high storage overhead on untethered nodes (e.g. handhelds). Thus, efficient buffer management policies are necessary to decide which messages should be discarded, when node buffers are operated close to their capacity.**

In this paper, we propose efficient buffer management policies for delay tolerant networks. We show that traditional buffer management policies like *drop-tail* or *drop-front* fail to consider all relevant information in this context and are, thus, sub-optimal. Using the theory of encounter-based message dissemination, we propose an optimal buffer management policy based on global knowledge about the network. Our policy can be tuned either to minimize the average delivery delay or to maximize the average delivery rate. Finally, we introduce a distributed algorithm that uses statistical learning to approximate the global knowledge required by the the optimal algorithm, in practice. Using simulations based on a synthetic mobility model and real mobility traces, we show that our buffer management policy based on statistical learning successfully approximates the performance of the optimal policy in all considered scenarios. At the same time, our policy outperforms existing ones in terms of both average delivery rate and delivery delay.

I. Introduction

The traditional view of a network as a connected graph over which end-to-end paths need to be established might not be appropriate for modeling existing and emerging wireless networks. Due to wireless propagation phenomena, node mobility, low power nodes periodically shutting down, etc., connectivity in many wireless networks is, more often than not, intermittent. Despite this limited or episodic connectivity, many emerging wireless applications could still be supported. Some examples are the low-cost Internet provision in remote or developing communities [1], [2], vehicular networks (VANETs) for dissemination of location-dependent information (e.g. local ads, traffic reports, parking information, etc) [3], *pocket-switched* wireless networks to extend and sometimes bypass access point connectivity to the Internet [4], [5], [6], underwater networks [7], etc.

To enable some services to operate even under these challenging conditions, researchers have proposed a new networking paradigm, often referred to as Delay Tolerant Networking (DTN [8]). To route messages in DTNs, *store-carry-and-forward* protocols are proposed, where a node may store a message in its buffer and carry it along for long periods of time, until it can forward it further. This routing may happen randomly, be based on statistical information [9], or even other relevant information about the destination (e.g. social links, affiliation, etc.). Furthermore, due to the inherent uncertainty caused by the lack of complete (or any) information about other nodes in the network, many replicas of the same message may be propagated to increase probability of delivery. For example, one of the first and most popular routing protocols in this context, namely *Epidemic* routing [10], disseminates a message replica to *every* node in the network.

Although a large amount of effort has been invested in the design of efficient routing algorithms for DTNs, there has not been a similar focus on buffer management policies. Yet, the combination of long-term storage and the, often extensive, message replication performed by many DTN routing protocols [10], [9] imposes a high storage overhead on wireless nodes (e.g. small handhelds, sensors, etc.). Moreover, the data units disseminated in this context, called *bundles*, are self-contained, atomic application-level data units, which can often be large [8]. It is evident that, in this context, node buffers will very likely run out of capacity and, thus, efficient buffer management policies are necessary to decide which message(s) should be discarded when a node's buffer is full.

It has been demonstrated that buffer constraints can severely affect the relative and absolute performance of DTN routing schemes and consequently applications. For example, a number of studies have clearly shown that *Epidemic* routing has minimum delivery delay under no buffer constraints (and no bandwidth constraints), but performs poorly when buffer sizes are limited [11], [12]. However, it is less clear what the right buffer management policy is, in this context. For example, the simple *drop-tail* policy, used in many networks, has been shown to perform poorly in the DTN context [12]. Although some improvement can be achieved, for example, using other

policies like *drop-front* [12], existing policies fail to take into account the intrinsic characteristics and requirements of delay tolerant networking and store-carry-and-forward routing. For example, one of our results in this paper is that the best choice of the message to drop is strongly dependent on the number of copies in the network of the different messages existing in the DTN node's buffer. None of the existing policies takes this network-wide statistic into account.

In this paper, we try to solve this problem in its foundation. We develop a theoretical framework based on *Epidemic* message dissemination [13], [14], [15] that takes into account all information that are relevant for encounter-based (or store-carry-and-forward) message delivery. Based on this theory, we first propose an optimal buffer management policy. This policy uses global information about the network either to maximize the average delivery rate or to minimize the average delivery delay. Then, we propose a distributed (local) algorithm that uses statistical learning in order to estimate information about the global state of the network, and uses this estimation to approximate the optimal algorithm in practice. Finally, to evaluate the performance of the optimal and our statistical learning algorithm against other buffer management policies, we have implemented a DTN framework including all policies in the network simulator NS-2 [16]. We have performed simulations for both the Random Waypoint model and two real-world mobility traces, the ZebraNet trace [17] and the San Francisco's Yellow Cab taxi trace [18]. Simulation results show that our statistical learning algorithm outperforms other policies achieving close-to-optimal performance, in all considered scenarios.

The rest of this paper is organized as follow. Section II gives the framework and assumptions of our study. In Section III, we establish theoretically an optimal, "reference" buffer management policy that uses global knowledge about the network. Then, we present in Section IV a learning process that enables us to approximate the global network state required by the reference policy. Section V describes the experimental setup and the results of our performance evaluation. Finally, we summarize our conclusions and discuss future work in Section VI.

II. Delay-Tolerant Routing and Buffer Management Policies

In this section, we briefly introduce the basic DTN routing scheme that we will use throughout the paper, describe some existing buffer management policies that we will compare our policies to, and go over some related work.

A. DTN routing

In the DTN context, when nodes encounter each other they perform pair-wise exchanges of messages with the goal that each message will *eventually* be delivered to its destination. An index of all messages carried by a node, called *summary vector*, is kept by each node, and when two nodes meet, they exchange summary vectors [10]. After this exchange, each node can determine if the peer node has any messages other than the ones stored locally, and based on this, it can decide which messages among them to forward to (or to request from) its peer. When no other node is currently within communication range, messages are buffered.

One of the simplest routing protocols that one could implement based on the above mechanism is the *Epidemic* routing protocol proposed in [10]. This protocol relies on the theory of *Epidemic* dissemination, where two nodes always exchange all messages they don't have in common when they encounter each other. Thus, if there is enough buffer space, messages will spread like an "epidemic" through the network, with every node eventually receiving (a copy of) the message[1]. As a result, *Epidemic* routing uses the maximum amount of resources and causes the highest amount of congestion.

More recently, a number of other routing protocols have been proposed, e.g. Spray and Wait [19], Prophet [9], etc., that aim at reducing the overhead of Epidemic routing. We have chosen to base our study on *Epidemic* routing due to its simplicity, its optimal performance in terms of delay and delivery rate when resources are unbounded, but also the high load it causes on network buffers. However, our policies described in Sections III and IV can apply to other multiple copy schemes, as well. The only difference is a higher number of drop decisions to be taken in the case of Epidemic routing.

B. Buffer management policies

A buffer management policy defines which message to drop if the buffer of a DTN node is full when a new message is to be accommodated. Each message i in the buffer (B messages in total) has a set of information S_i stored with it; S_i includes: the source id, the time since the message was generated, the Time-To-Live (TTL), etc. In the DTN architecture [20], the TTL value is a timeout value, which specifies when a message is no longer useful and should be deleted. Let a new message arrive at a buffer that is full. Then, usually, a buffer management policy is a function $f(S_1, S_2, \ldots, S_B, S_{new}) = j \in \{[1, B] \cup \{new\}\}$. This policy, decides on which message to drop among the ones already in the buffer and the new one, based on the information on all messages in the buffer.

To evaluate the performance of our proposed buffer management policies, we have chosen to compare them with a set of existing policies that have been used in related work [21]. This set includes: (i) *DL-Drop Last* (or "Droptail"), the most common among the set, simply removes the newly received message, (ii) *DF-Drop Front*, handles the queue in a FIFO order. The message that was first entered into the queue is the first message to be dropped when the buffer is full, (iii) *DO-Drop Oldest*, drops the message with the shortest remaining life time (closest to TTL expiration), and (iv) *DY-Drop Youngest*, drops the message with the longest remaining life time first.

[1]Finite bandwidth and unexpected interruptions may not allow a node to transmit all the messages it would like to forward. In such cases, the order in which messages are transmitted is important. In this work, we will assume that enough bandwidth is available for each contact, and thus buffer space is the only constrain on performance.

As an additional optimization, we consider that a node should not discard its own valid messages (called source messages) to create a place in its buffer for messages forwarded by other nodes. This ensures that at least one copy of each message stays in the network as long as its TTL does not expire. If all buffered messages are source ones, and the arriving message is also a source message, then we choose to delete the oldest one. This intuitive idea of giving priority to source messages has been proposed in [12] and was shown to improve the average delivery rate.

C. *Related work*

Several solutions have been proposed to handle routing in DTN. Yet, an important issue that has been largely disregarded by the DTN community is the impact of buffer management policies on the overall performance. In [12], Zhang et al. present an analysis of buffer-constrained *Epidemic* routing, and evaluate some of the simple buffer management policies previously described. The authors conclude that DF outperforms DL in terms of both delivery delay and delivery rate. Additionally, they notice that giving priority to source messages improves the delivery rate further, but makes messages spread slower, increasing hence their delay. In [21], Lindgren et al. evaluate a somewhat more extensive set of combinations of existing buffer management policies and routing protocols for DTNs. They show that Probabilistic routing [9] together with the right buffer management policy can result in better performance in terms of message delivery, overhead, and end-to-end delay. Specifically, in the context of *Epidemic* routing, the authors find that DF (with priority to source messages) gives the highest delivery rate, while DO gives the smallest end-to-end delay. Our own results support these findings.

All the buffer management policies discussed so far fail to consider network-wide information, such as the number of replicas of each message, the number of nodes, etc. Yet, optimality cannot be achieved without this information. To our best knowledge, the only relevant work that takes encounter information into account is RAPID [22]. However, RAPID's focus is on scheduling messages under limited bandwidth, and also uses a suboptimal policy. We address the issue of optimal scheduling and compare ourselves against RAPID in [23].

III. Optimal Buffer Management Policy

In this section, we formalize the problem of choosing which message to discard when a node's buffer is full. We first make some assumptions regarding the routing protocol in hand, the service model and the mobility characteristics of the nodes. Then, we embark on finding theoretically the optimal buffer management policy, *GBD (Global Knowledge based Drop)*, based on global knowledge about the network state. As global knowledge is required, GBD is difficult to be implemented, thus, it will serve as a point of reference. In Section IV, we will show how to design a *local* buffer management policy that uses learning methods to estimate the global information about the network assumed by GBD, and can approximate the performance of the optimal policy in practice.

A. *Problem description (assumptions)*

We will 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 is destined to one of the nodes in the network, and has a Time-To-Live (TTL) value. After this time is elapsed, the message is no more useful to the application and should just be dropped by its source and all intermediate nodes. The message can also be dropped when a notification of delivery is received, if an "anti-packet" mechanism is implemented [12][2].

In the context of DTNs, message transmissions occur only when nodes encounter each other. The minimum time a node has to wait until it can forward a message further, is the time until it encounters another node which can act as a relay. Thus, *the time elapsed between node meetings is the basic delay component*.

This inter-encounter time between nodes depends on the value of a particular property of the mobility model assumed, namely the *meeting* time [24], [25][3]. We further consider that bandwidth is not an issue so when two nodes meet, there is enough time to exchange their messages. Messages are not fragmented and are transmitted in a FIFO order from one node to another during a contact. We consider bandwidth impact in [23].

Definition III.1. *Meeting Time: Let nodes i and j move according to some mobility process and let them start from their stationary distribution at time 0. Let further $X_i(t)$ and $X_j(t)$ describe the mobility process (position) of nodes i and j, respectively, at time t. The meeting time (U) between the two nodes is defined as the time it takes them to first come within transmission range (β) of each other, that is $U = \min_t\{t : \|X_i(t) - X_j(t)\| \leq \beta\}$.*

To formulate the optimal buffer management policy problem, we do not make any assumption about a specific mobility model used. Our only requirement is that *the meeting time of the mobility model is exponentially distributed or has at least an exponential tail, with parameter $\lambda = \frac{1}{E[U]}$*, where $E[X]$ denotes the expectation of a random variable X.

It has been shown that many popular mobility models like Random Walk [24], Random Waypoint and Random Direction [15], [14], as well as other more sophisticated synthetic models like the community model in [15] have such a property. In practice, there exist some recent studies based on traces collected from real-life mobility [6] that argue that inter-encounter and contact duration times may follow a power-law

[2]Once a node delivers a packet to the destination, it should delete the packet from its buffer to save storage space and prevent the node from infecting other nodes. Moreover, to avoid being reinfected by the packet, a node can keep track of packet delivery. We refer to this information stored at the node as "anti-packet"; various algorithms have been proposed to also propagate anti-packets to other infected and susceptible nodes [12].

[3]If some of the nodes in the network are static, then one needs to use the *hitting* time between a mobile node and a static node, instead. For simplicity, we assume here that all nodes are mobile and we refer only to meeting times thereafter. Our theory can be easily modified to account for static nodes.

distribution, instead. Yet, the authors in [26] show that even these traces in fact exhibit exponential tails after a cutoff point, and argue that for most mobility models that can be seen as a random walk on a graph, meeting times have an exponential tail. For this reason, we choose to stick with the exponential meeting time assumption, which makes our analysis tractable. Our trace-based evaluation further supports this assumption.

TABLE I
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; $\lambda = \frac{1}{E[U]}$ where $E[U]$ is the average meeting time

Given the above problem setting, a key question to answer is the following: if a node is congested, which message should it drop so as to optimize a specific routing metric? Our optimal buffer management policy derives a per-message *utility*, and then drops the message with the smallest utility value. This utility captures the *marginal value* of a given message copy for the overall routing process, and with respect to the chosen optimization metric. We derive here such a *utility* for two popular metrics: maximizing the average delivery rate, and minimizing the average delivery delay.

In Table I, we summarize the various quantities and notations we will use throughout the paper.

B. *Maximizing the average delivery rate*

We will first look into the following scenario. We assume that a number of messages are propagated in the network using replication (e.g. *Epidemic*), each of which 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. We consider a time instant where the network is *congested* and a new message copy arrives to a new node during an encounter, to find its buffer full. Assuming now that we know all messages in the network and the number of copies for each message at that time, the problem we would like to solve is: *what is the best message to be dropped (*locally*), among the ones already in the buffer of the given node and the newly arrived one, in order to maximize the average delivery rate among all messages in the network (*globally*)?* The answer is given in the following theorem.

Theorem III.1. *Delivery-Rate: Let us assume that there are K messages in the network, with elapsed time T_i for the message i at the moment when the drop decision by the node is to be taken. For each message $i \in [1, K]$, let $m_i(T_i)$ and $n_i(T_i)$ be the number of nodes that have "seen" the message since it's creation*[4] *(excluding the source) and those who have a copy of it at this instant ($n_i(T_i) \leqslant m_i(T_i) + 1$). The local optimal buffer management policy that maximizes the average delivery rate is to drop the message i_{min} satisfying:*

$$i_{min} = \underset{i}{argmin} \left[(1 - \frac{m_i(T_i)}{L-1}) \lambda R_i \exp(-\lambda n_i(T_i) R_i) \right] \quad (1)$$

Proof: We know that the meeting time between nodes is exponentially distributed with parameter λ. The probability that a copy of a message i will not be delivered by a node is then given by the probability that the next meeting time with the destination is greater than R_i. This is equal to $\exp(-\lambda R_i)$.

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):

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

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, also we have 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). Predicting future encounters and the effect of further replicas created complicates the problem significantly. Nevertheless, the same assumptions are performed for all messages equally and thus can justify the relative comparison between the delivery probabilities for different messages.

We should also take into consideration what has happened in the network since the message generation, in the absence of an explicit delivery notification. Since 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:

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

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

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

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

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

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

In case of congestion, a DTN node should take a drop decision, that leads to the best gain in the global delivery rate DR. To define this optimal decision, we differentiate DR with respect to $n_i(T_i)$, then we discretize and replace the dn by Δn to obtain:

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

The best drop decision is the one that maximizes $\Delta(DR)$. 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. Hence, the optimal buffer management policy that maximizes the future delivery rate is the one that drops message i having the smallest value of the following utility:

$$(1 - \frac{m_i(T_i)}{L-1}) \lambda R_i \exp(-\lambda n_i(T_i) R_i). \quad (4)$$

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 in the network.

■

C. *Minimizing the average delivery delay*

We now turn our attention to minimizing the average delivery delay. We assume that all messages generated have infinite TTL or at least a TTL value large enough to ensure a delivery probability close to 1. In this context, we will look for a buffer management policy that minimizes the expected delivery delay over all messages in the network.

Theorem III.2. *To minimize the average delivery delay of all messages, a DTN node should drop the message i_{min} satisfying:*

$$i_{min} = \underset{i}{argmin} \left[\frac{1}{n_i(T_i)^2 \lambda} (1 - \frac{m_i(T_i)}{L-1}) \right] \quad (5)$$

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 (D) for all messages for which copies still exist in the network (or if we want in the local buffer) is given by,

$$D = \sum_{i=1}^{K(t)} \left[\frac{m_i(T_i)}{L-1} * 0 + (1 - \frac{m_i(T_i)}{L-1}) * E[X_i | X_i > T_i] \right]. \quad (6)$$

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,

$$E[X_i | X_i > T_i] = T_i + \frac{1}{n_i(T_i)\lambda}. \quad (7)$$

Substituting Eq.(7) in Eq.(6), we get,

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

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

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

The best drop decision will be the one that maximizes $|\Delta(D)|$ (or $-\Delta(D)$). This corresponds to dropping the message i that minimizes the following utility metric,

$$\frac{1}{n_i(T_i)^2 \lambda} (1 - \frac{m_i(T_i)}{L-1}) \quad (8)$$

This per-message utility is different than the one for the delivery rate and can be seen as the *marginal utility value* of a copy of a message i regarding the average delivery delay. Again it is a function of the global state of this message across the network. ■

IV. USING LEARNING TO APPROXIMATE GLOBAL KNOWLEDGE IN PRACTICE

In order to optimize a specific routing metric using GBD, we need global information about the network and the "spread" of messages. In particular, for each message present in the node's buffer, we need to know the values of $m_i(T_i)$ and $n_i(T_i)$, the number of nodes that have seen the message and those that have a copy of it. Unfortunately, this is not feasible in practice due to intermittent network connectivity and the long time it takes to flood buffer status information across DTN nodes, which could make such info obsolete. Our proposed solution is to find appropriate estimators for these utilities.

We do this by designing and implementing a learning process that permits to a DTN node to gather knowledge about the global network state history 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, which could be 0 if the message was in the node's buffer at the specified time or 1 if the message was seen but deleted due to congestion as described in Figure 1. Note that each node maintains the time of the last list update and only sends the list if it has been updated since the last exchange. This way and after some time, all nodes will have the same global and accurate view about the network history. This history can be limited to some time duration if the network size is large.

Since the global information thus gathered on a specific message might take a long time to propagate (as mentioned earlier) and hence might be obsolete when we calculate the

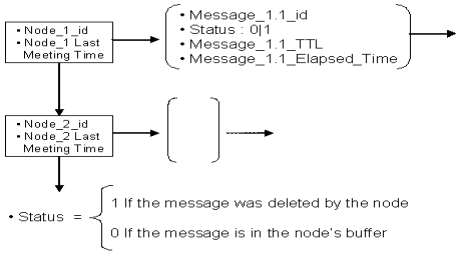

Fig. 1. List maintained by each DTN node.

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 assume that the quantities $m_i(T)$ and $n_i(T)$ at elapsed time T are distributed according to random variables $N(T)$ and $M(T)$, respectively. Further, let's denote by $\hat{n}(T)$ and $\hat{m}(T)$ estimators for $n_i(T)$ and $m_i(T)$. By finding appropriate estimators $\hat{n}(T)$ and $\hat{m}(T)$ and plugging them into the GBD's delivery rate and delay utility-metrics calculated in Section III, we get two new utility-metrics, which could be used by a DTN node without any need for global information about messages. This results in a new buffer management policy, called *HBD (History Based Drop)*, a deployable variant of GBD that uses the new utilities based on estimates of m and n. The estimation algorithms are described in paragraphs IV-A and IV-B.

As a final note, in order to justify our motivation for the history-based learning process described above, we introduce another buffer management policy, *FBD (Flood Based Drop)*. FBD accounts only for the global information collected using simple message flooding, that is, without considering past history or other messages. So, from the list (in Fig. 1), DTN nodes extract $n_i(T_i)$ value for message i simply by looking at the number of nodes that said they hold it and the $m_i(T_i)$ value by looking at those nodes that said they saw it, even if this information is obsolete. These values are then plugged into the GBD's delivery rate and delay utilities as in the case of HBD. Our results from Section V indicate that, unlike HBD, FBD approximates poorly GBD's performance for both routing metrics, and thus is not sufficient to infer the required global information in practice.

A. *Calculating estimators* $\hat{n}(T)$ *and* $\hat{m}(T)$ *for the average delivery rate utility*

When the global information is unavailable, one can calculate the average delivery rate of a message, by averaging over all possible values for random variables $M(T)$ and $N(T)$. Then, one can try to minimize this average. In the framework of the GDB, 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.

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

By plugging in the utility expression in Eq.(4) any values of $\hat{n}(T)$ and $\hat{m}(T)$ that verify this equality, one can be sure that the obtained policy minimizes the average delivery rate. This is exactly our purpose. Suppose now that the best estimator for $\hat{m}(T)$ is its average, i.e., $\hat{m}(T) = \bar{m}(T) = E[M(T)]$. A justification for this assumption will be given in paragraph IV-C. Then, we solve the equation for:

$$\hat{n}(T) = -\frac{1}{\lambda R_i}\ln(\frac{E[(1-\frac{M(T)}{L-1})\exp(-\lambda N(T)R_i)]}{(1-\frac{\bar{m}(T)}{L-1})}) \quad (9)$$

Substituting this expression into the GBD's delivery rate utility in Eq.(4), we get the following utility for HBD,

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

The expectation in this expression is calculated by summing over all values of $N(T)$ and $M(T)$ for past messages at elapsed time T. Note, that L, the number of nodes in the network, could be calculated from the list maintained by each node in the network. In this work, we assume it to be fixed and known, but one could estimate it as well in the same way we do for n and m, or using some additional estimation algorithm. We defer this for future work. Thus, unlike GBD's delivery rate utility, this new utility is a function only of past and accurate global network history and so can be calculated locally since it does not depend on the flooding time.

B. *Calculating estimators* $\hat{n}(T)$ *and* $\hat{m}(T)$ *for the average 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 calculation is unbiased.

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

Again, supposing that $\hat{m}(T) = \bar{m}(T) = E[M(T)]$ and simplifying this last expression further, we obtain:

$$\hat{n}(T) = \frac{L-1-\bar{m}(T)}{E[\frac{L-1-M(T)}{N(T)}]} \quad (10)$$

By substituting this value in the GBD's delivery delay utility in Eq.(8), we can find the delay utility specific to HBD,

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

Also, unlike GBD's delivery delay utility, this new utility is function of the locally available history of other messages.

C. *On the approximation of* $\hat{m}(T)$ *by* $E[M(T)]$

In paragraphs IV-A and IV-B, we have supposed that $\hat{m}(T) = E[M(T)]$. 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 [27], 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 $M(T)$.

V. Performance Evaluation

A. *Experimental setup*

To evaluate our policies, we have implemented a DTN framework into the Network Simulator NS-2. This implementation includes the *Epidemic* routing protocol, the buffer management policies described in Section II, against which we compare our policies, and the VACCINE anti-packet mechanism described in [12][5]. Each node uses a wireless communication channel 802.11b that has a range of 100 meters, to obtain network scenarios that are neither fully connected (e.g. MANET) nor extremely sparse. Our simulations are based on three mobility patterns, a synthetic one, based on the Random Waypoint model and two real-world mobility traces: the first trace was collected as part of the ZebraNet wildlife tracking experiment in Kenya described in [17]. The second mobility trace tracks San Francisco's Yellow Cab taxis. Many cab companies outfit their cabs with *GPS* to aid in rapidly dispatching cabs to their costumers. The Cabspotting system [18] talks to the Yellow Cab server and stores the data in a database. We have use an API provided by the Cabspotting system in order to extract mobility traces. Note that this trace describes taxi's positions according to the *GPS* cylindrical coordinates (*Longitude*, *Latitude*) and in order to uses these traces as input for the NS-2 simulator, we have implemented a tool based in the Mercator [28] cylindrical map projection which permit us to convert traces to plane coordinates.

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 1KB[6] for a randomly chosen destination. Unless otherwise stated, we associate to each node a buffer with a capacity of 10 messages. Finally, we assume that each time two nodes meet, they have enough time (bandwidth) to exchange all data and control messages, so we take into consideration only buffer constraints.

We compare the performance of the various buffer management policies using the following two metrics: the message's average delivery rate and the delivery delay in the case of infinite TTL. Concerning the evaluation of the HBD policy, we suppose that different nodes are already in a converged state, so we start accounting for HBD's results $2*TTL$ seconds after simulation starts. The choice of this value will be justified in paragraph V-D. Note that the results presented here are averages from 20 simulation runs, which is enough to ensure convergence.

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

[6]In future work, we intend to evaluate the effect of message size with realistic wireless communication environments.

B. *Performance evaluation for delivery rate*

First, we compare the delivery rate of all policies for the three scenarios shown in Table II. Figures 2, 3 and 4 show respectively the delivery rate based on the Random Waypoint model, the ZebraNet trace, and the Taxi trace.

TABLE II
Simulation parameters

Mobility pattern:	RWP	Zebra's Traces	Taxi's Traces
Simulation's Duration(s):	5000	5000	36000
Simulation' Area (m^2):	1000*1000	1500*1500	-
Number of Nodes:	30	40	40
Average Speed (Km/H):	6	-	-
$TTL(s)$:	650	650	7200
CBR Interval(s):	200	200	2100

From these plots, it can be seen that the GBD policy gives the best performance for all numbers of sources. When congestion-level decreases, so does the difference between GBD and other policies, as expected. Moreover, the HBD policy outperforms existing policies (DF, DO, DL, DY) and performs very close to GBD. For example, for 30 sources and Random Waypoint mobility, HBD's delivery rate is 10% higher than Drop Front and only 2% worse than GBD. Similarly, for 40 sources and the ZebraNet traces, HBD delivers 12% more messages than Drop Front and 3% worse than GBD. Finally for the Taxi traces and 40 sources HBD performs 17% better than Drop Front and 3% worse than GBD.

Note that the fact that Drop Front gives a higher delivery rate than Drop Oldest could be deduced from our delivery rate utility in Eq.(4). Specifically, at an instant t, we have $R_{front} - R_{oldest} >> |n_{front} - n_{oldest}|$, which implies that our delivery rate utility gives a smaller value for the message at the front of the queue than for the oldest message. Hence, our utility predicts that dropping the message at the head of the queue will increase the delivery rate more than dropping the oldest.

In order to justify our motivation for the learning process we also compare our HBD policy explicitly to FBD that uses only collected information per message. Figure 5 shows that when the congestion level increases the difference between FBD and GBD becomes significant, unlike the case of HBD. For example, for 30 CBR sources the difference is about 19% while HBD differs from GBD only by 2%. These results further underline the importance of the history-based learning process in order to implement GBD in practice.

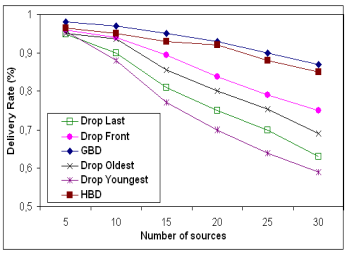

Fig. 2. Average delivery rate for Random Waypoint mobility.

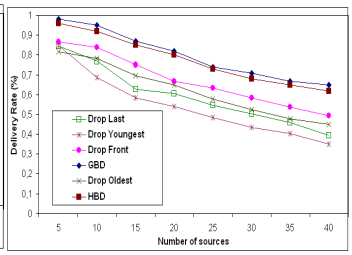

Fig. 3. Average delivery rate for the ZebraNet trace.

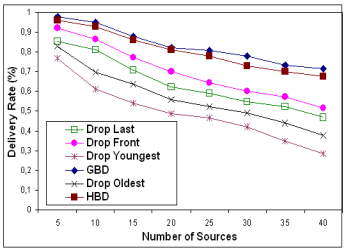

Fig. 4. Average delivery rate for the Taxi trace.

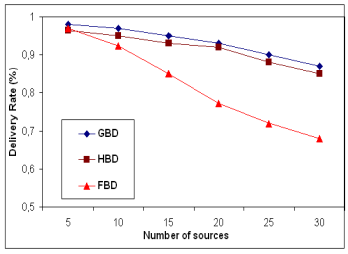

Fig. 5. Comparison of HBD and FBD delivery rate for Random Waypoint mobility.

So far we assumed the same TTL value for all generated messages in different scenarios. In our last scenario, we fix the number of sources to 30 and arrange them in three groups of 10 nodes. Nodes in each group generate messages with TTL values equal to 250, 450 and 650 seconds, respectively. We range here the buffer size from 10 to 35 messages. (Note that in this scenario we start accounting for HBD's results $2 * 650 = 1300$ seconds after simulations starts, for convergence.) Figure 6 shows that even for this scenario, HBD outperforms the four existing buffer management policies (DF, DO, DL and DY) and performs close to GBD. For example, for buffer size equal to 10 messages, HBD delivers 14% messages more than Drop Front and 3% messages less than GBD.

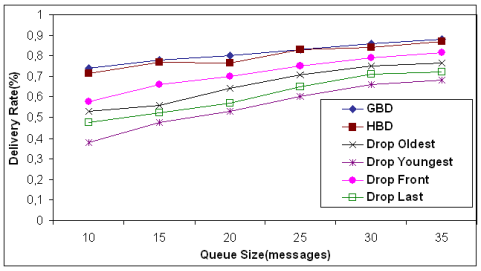

Fig. 6. Average delivery rate for Random Waypoint mobility and different $TTL(s)$.

C. *Performance evaluation for delivery delay*

To evaluate the average delivery delay metric, we keep the same simulations durations and messages generation rates as those used for the delivery rate. Figures 7, 8 and 9 depict the average delivery delay for the Random Waypoint model, the ZebraNet trace, and the Taxi trace, respectively. As in the case of delivery rate, GBD gives the best performance for all three mobility patterns. Moreover, the HBD policy outperforms the four buffer management policies (DF, DO, DL, DY) and performs close to GBD. Specifically, for 30 sources and Random Waypoint mobility, HBD's average delivery delay is 24% better than Drop Oldest and 6% worse than GBD. For the ZebraNet traces, HBD performs 15% better than Drop Oldest and 5% worse than GBD. Finally the highest improvement was observed for the Taxi trace, where HBD performs 29% better than Drop Oldest and only 3% worse than GBD.

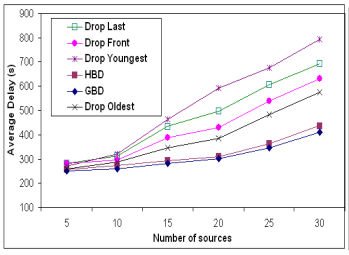

Fig. 7. average delivery delay for the Random Waypoint mobility.

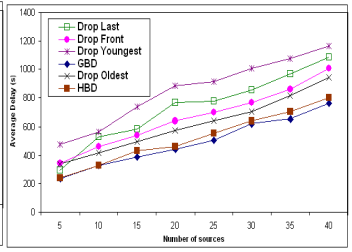

Fig. 8. average delivery delay for the ZebraNet trace.

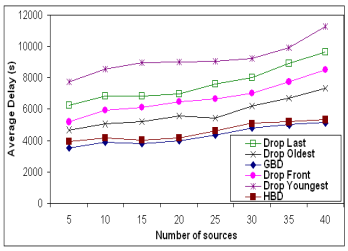

Fig. 9. average delivery delay for the Taxi trace.

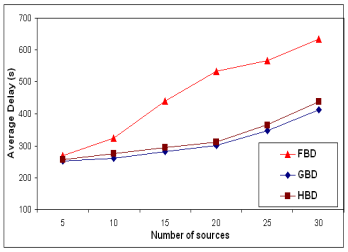

Fig. 10. Comparison of HBD and FBD delivery delay for Random Waypoint mobility.

Once more, the fact that Drop Oldest has smaller delivery delay than Drop Front can be justified based on our delay utility in Eq.(8). Specifically, in most cases the oldest messages have a greatest number of copies, on average, than the message at the front of the queue. Thus, if we apply our delay utility to the oldest message, we will get a smaller value than for the messages in the front of the queue ($\frac{1}{n^2_{oldest}(t)} << \frac{1}{n^2_{front}(t)}$), which explains why dropping the oldest message gives a smaller average delivery delay than dropping the message in the front of the queue.

Finally, in order to further emphasize on the importance of the learning process we compare again the HBD policy to GND and FBD, in terms of delivery delay also. Figure 10 shows that, when congestion level increases, the difference between FBD and GBD becomes significant which is not the case of HBD. For 30 CBR sources, the difference is 30% while HBD differs from GBD is 6%.

D. *HBD's Convergence*

In this last part, we look at the time taken by the learning process to converge. We consider the same simulations parameters as in Table II fixing the number of sources to 15. For the HBD policy and for different randomly chosen nodes, Figure 11 shows that as the number of measurements at

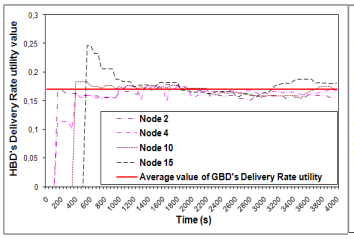

Fig. 11. HBD's delivery rate utility convergence for different randomly chosen nodes.

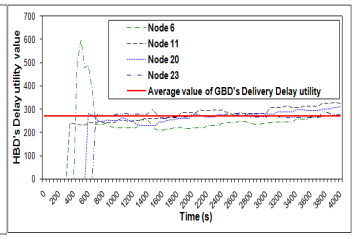

Fig. 12. HBD's delivery delay utility convergence for different randomly chosen nodes.

elapsed time $T = 100$ increases, the delivery rate utility's value increases and converges to the average value of the GBD's delivery rate utility, which is equal to 0.17. From Figure 11, one can also extract the convergence times of the delivery rate utility. These values are illustrated in Table III. In Figure 12, we depict similar convergence results for delivery delay. The average delivery delay utility converges to 271, the average value of the HBD's utility. The different convergence times are described in Table IV. These results justify the choice

TABLE III
THE TIME OF CONVERGENCE OF HBD'S DELIVERY RATE UTILITY

Node 2	1100 seconds
Node 4	1000 seconds
Node 10	600 seconds
Node 15	1150 seconds

TABLE IV
THE TIME OF CONVERGENCE OF HBD'S DELIVERY DELAY UTILITY

Node 6	700 seconds
Node 11	600 seconds
Node 20	750 seconds
Node 23	700 seconds

of the time of convergence in V-A. Indeed, in the different simulations scenarios described above, the HBD's utilities take less than $2 * TTL$ seconds to converge to the GBD's average utilities values for a fixed elapsed time.

VI. CONCLUSION AND FUTURE WORK

In this work, we investigated the problem of buffer management in delay tolerant networks. First, we proposed an optimal buffer management policy based on global knowledge about the network state. Then, we have introduced a distributed algorithm that uses statistical learning to approximate the required global knowledge of the optimal algorithm. Using simulations based on a synthetic mobility model (Random Waypoint), and two real mobility traces (ZebraNet and San Francisco taxi traces), we showed that our buffer management policy based on statistical learning successfully approximates the performance of the optimal algorithm in all considered scenarios. Finally, both policies outperform existing policies with respect to delivery rate and delivery delay, in all considered scenarios.

Note that in this work, we considered that all messages have the same size. It would be interesting to define buffer management policies that take into account different messages sizes. For example, in case of congestion, the end-to-end delay versus message delivery trade-off could be influenced by the choice of dropping several small messages or one large message that occupies the entire node's buffer.

ACKNOWLEDGMENTS

We thank Michal Piorkowski and Wei-Jen Hsu for pointing us out to the San Francisco's taxi cab mobility traces.

REFERENCES

[1] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *Proceedings of ACM SIGCOMM*, Aug. 2004.
[2] A. Pentland, R. Fletcher, and A. Hasson, "Daknet: Rethinking connectivity in developing nations," *IEEE Computer*, 2004.
[3] P. Basu and T. Little, "Networked parking spaces: architecture and applications," in *IEEE Vehicular Technology Conference (VTC)*, 2002.
[4] M. Papadopouli and H. Schulzrinne, "Seven degrees of separation in mobile ad hoc networks," in *Proceedings of IEEE GLOBECOM*, 2000.
[5] M. Motani, V. Srinivasan, and P. S. Nuggehalli, "Peoplenet: engineering a wireless virtual social network," in *Proceedings of ACM/IEEE MobiCom*, 2005.
[6] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of human mobility on the design of opportunistic forwarding algorithms," in *Proceedings of IEEE INFOCOM*, 2006.
[7] J. Heidemann, W. Ye, J. Wills, A. Syed, and Y. Li, "Research challenges and applications for underwater sensor networking," in *Proceedings of the IEEE Wireless Communications and Networking Conference*, 2006.
[8] "Delay tolerant networking research group," *http://www.dtnrg.org*.
[9] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mobile Computing and Communication Review*, vol. 7, no. 3, 2003.
[10] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke University, Tech. Rep. CS-200006, 2000.
[11] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Efficient routing in intermittently connected mobile networks: The multiple-copy case," *to appear in Transactions on Networking*, Feb. 2008.
[12] X. Zhang, G. Neglia, J. Kurose, and D. Towsley, "Performance modeling of epidemic routing," in *Proceedings of IFIP Networking*, 2006.
[13] 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.
[14] R.Groenevelt, G. Koole, and P. Nain, "Message delay in manet (extended abstract)," in *Proc. ACM Sigmetrics*, 2005.
[15] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Performance analysis of mobility-assisted routing," in *Proceedings of ACM/IEEE MOBIHOC*, 2006.
[16] "The network simulator ns-2," *http://www.isi.edu/nsnam/ns/*.
[17] Y. Wang, P. Zhang, T. Liu, C. Sadler, and M. Martonosi, "Movement data traces from princeton zebranet deployments," CRAWDAD Database.
[18] "Cabspotting project," *http://cabspotting.org/*.
[19] K. P. Thrasyvoulos Spyropoulos and C. Raghavendra, "An efficient routing scheme for intermittently connected mobile networks," *ACM SIGCOMM workshop on Delay Tolerant Networking (WDTN-05)*, 2005.
[20] K. Scott and S. Burleigh, "Bundle protocol specification." RFC 5050, November 2007.
[21] A. Lindgren and K. S. Phanse, "Evaluation of queuing policies and forwarding strategies for routing in intermittently connected networks," in *Proc. of IEEE COMSWARE*, January 2006.
[22] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "Dtn routing as a resource allocation problem," in *Proc. ACM SIGCOMM*, August 2007.
[23] A. Krifa, C. Barakat, and T. Spyropoulos, "An optimal joint scheduling and drop policy for delay tolerant networks," in *to appear in WoWMoM workshop on Autonomic and Opportunistic Communication (AOC)*, June 2008.
[24] 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*.
[25] R. Durrett, *Probability: Theory and Examples*, 2nd ed. Duxbury Press, 1995.
[26] M. V. Thomas Karagiannis, Jean-Yves Le Boudec, "Power law and exponential decay of inter contact times between mobile devices," in *Proc. of ACM/IEEE MobiCom*, 2007.
[27] 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, June 1967.
[28] "The mercator projection," *http://en.wikipedia.org/wiki/Mercator_projection/*.