An Optimal Joint Scheduling and Drop Policy for Delay Tolerant Networks

Amir Krifa, Chadi Barakat
Project-Team Planète, INRIA Sophia-Antipolis, France
Thrasyvoulos Spyropoulos
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 (DTN) are wireless networks where disconnections may occur frequently. In order to achieve data delivery in DTNs, 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. Multiple message replicas are often propagated to increase delivery probability. This combination of long-term storage and 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 an efficient joint scheduling and drop policy that can optimize different performance metrics, like average delay and delivery probability. Using the theory of encounter-based message dissemination, we first propose an optimal policy based on global knowledge about the network. Then, we introduce a distributed algorithm that can approximate the performance of the optimal algorithm, in practice. Using simulations based on a synthetic mobility model and a real mobility trace, we show that our optimal policy and its distributed variant outperform existing resource allocation schemes for DTNs, such as the RAPID protocol [4], both in terms of average delivery ratio and delivery delay.

1 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, etc., connectivity in many wireless networks is often intermittent. 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 [2]), based on the *store-carry-and-forward* routing principle [7].

Despite a large amount of effort invested in the design of efficient routing algorithms for DTNs, there has not been a similar focus on drop and scheduling policies. Yet, the combination of long-term storage and the, often expensive, message replication performed by many DTN routing protocols [15, 9] impose a high bandwidth and storage overhead on wireless nodes [13]. Moreover, the data units disseminated in this context, called *bundles*, are self-contained, application-level data units, which can often be large [2]. It is evident that, in this context, node buffers will very likely run out of capacity. For the same reasons, when mobility results in short contacts between nodes, available bandwidth could be insufficient to communicate all intended messages. Consequently, efficient drop policies are necessary to decide which message(s) should be discarded when a node's buffer is full, together with efficient scheduling policies to decide which messages should be chosen when bandwidth is limited, *regardless of the specific routing algorithm used*.

In this paper, we try to solve this problem in its foundation. We develop a theoretical framework based on Epidemic message dissemination [6, 11, 12] that takes into account all information that are relevant for message delivery. Based on this theory, we first propose an optimal joint scheduling and drop policy, GBSD (Global khnowledge Based Scheduling and Drop) that can maximize the average delivery rate or minimize the average delivery delay by deriving a per-message utility and managing messages based on it. GBSD uses global information about the network to derive the per-message utility for a given routing metric, and thus is difficult to implement in practice. In order to amend this, we propose a second policy, HBSD(History Based Scheduling and Drop), employing a distributed (local) algorithm based on statistical learning from network history to estimate information about the current global status of the network that can be used later to calculate mes-

978-1-4244-2100-8/08/$25.00

sage utility. To our best knowledge, the recently proposed RAPID protocol [4] is the only effort aiming at scheduling (and to a lesser extend message drop) using such a theoretical framework, but is sub-optimal in a number of respects, as we will explain later.

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 describes the experimental setup and the results of our performance evaluation. Finally, we conclude this paper and discuss future work in Section 6.

2 Related Work

Several solutions have been proposed to handle routing in DTNs. Yet, an important issue that has been largely disregarded by the DTN community is the impact of buffer management and scheduling policies on the performance of the system[1]. In [16], 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 [10], confirming the performance of drop-front. However, all these policies are simple and/or heuristic that neither aim at optimality in the DTN context nor do they address scheduling. In a different work [8], 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 [16] and [10]. Due to space limitations, the comparison between various drop policies is not repeated here. We rather focus on the more general *joint scheduling and drop problem*, for which we believe the RAPID protocol [4] 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 own 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 two main drawbacks: (i) its policy is based on sub-optimal message utilities (more on this in Section 3); and (ii) in order to derive these utilities, RAPID requires the flooding of information about all the replicas of a given message in the queues of all nodes in the network; Yet, information propagated thus may be stale (a problem that the authors also note) due to change in the number of replicas, change in delivery delays, or if the message is delivered but acknowledgements have not yet propagated. In this paper, we propose a policy that fixes both (i) and (ii), and hence outperforms RAPID and other policies.

[1]These two problems have often been studied in somewhat different contexts, see for instance [5] which focuses on ad-hoc networks.

3 Optimal Joint Scheduling and Drop Policy

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 [3, 8][2]. To formulate the optimal policy problem, we do not make any specific assumption about the used mobility model. 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 [3], Random Waypoint and Random Direction [12] have such a property. Moreover, it has recently been argued that meeting and inter-meeting times observed in many traces may also exhibit an exponential tail [14].

Given the above problem setting and a routing metric, our policy GBSD derives a per-message utility. This utility 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: (i) *Scheduling*—at a limited transfer opportunity, a node should replicate messages in decreasing order of their utilities, and (ii) *Drop*—when a node exhausts all available storage, it should drop the message with the smallest utility among all buffered messages including the new received one(except source messages). We derive here such a per-message utility for two popular metrics: maximizing the average delivery rate, and minimizing the average delivery delay. In Table 1, we summarize the various quantities and notations we will use throughout the paper.

3.1 Maximizing the average delivery rate

To maximize the average delivery rate, the per-message utility used by GBSD is defined by the following theorem:

[2]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, as well. Although in this work we consider unicast transmissions where both sources and destinations are mobile, our theory can be easily modified to account for static nodes also.

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

Theorem 3.1. *Let us assume there are K messages in the network with elapsed time T_i for message i at the moment when the drop or replication decision by a 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*[3] *(excluding the source), and those who have a copy of it at this moment ($n_i(T_i) \leqslant m_i(T_i) + 1$), respectively. To maximize the average delivery rate of all messages, a DTN node should apply the GBSD policy using the following utility per message i:*

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

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 the remaining time 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):

$$\prod_{i=1}^{n_i(T_i)} \exp(-\lambda R_i) = \exp(-\lambda n_i(T_i)R_i).$$

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 the effect of future encounters complicates the problem significantly. Nevertheless, the same assumption is performed for all messages equally and thus can justify the relative comparison between the delivery probabilities for different messages. Unlike RAPID [4], we take into consideration what has happened in the network since the message generation, in the absence of an explicit delivery notification. 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:

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

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

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

In case of congestion or limited transfert 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 find this decision, we differentiate DR with respect to $n_i(T_i)$, then we discretize and replace dn by $\Delta(n)$ to obtain:

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

Our aim is to maximize $\Delta(DR)$. 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 or replicate and forward an already buffered message i to another node. Based on that, GBSD ranks messages using the per-message utility in Eq.(1), then schedules and drops them accordingly.

3.2 Minimizing the average delivery delay

We now turn our attention to minimizing the expected delivery delay over all messages in the network. 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:*

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

[3] 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 if it has already removed it from its buffer.

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) is given by,

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

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,

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

Then, as for the delivery ratio, we differentiate D with respect to $n_i(T_i)$ and find Eq.(2). 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.

4 Using network history to approximate global knowledge in practice

In order to optimize a specific routing metric using GBSD, we need global information about the network and the "spread" of messages. 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 the case of RAPID [4], it is assumed that this global view is obtained by flooding the above information (or a through secondary, "instantaneous" channel). However, our experiments show that the impact of the flooding delay is non 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 much better gain can be realized if we find estimators for the metrics involved in the calculation of message utilities, namely m and n.

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. More details on the learning process can be found in [8].

4.1 Calculating estimators $\hat{n}(T)$ and $\hat{m}(T)$ for the average delivery rate per-message utility

When the 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. We want our estimators for m and n of a message to be unbiased in terms of the average delivery rate, that is, we want to obtain the same expression for the average delivery rate when m and n are substituted by their estimations. In the framework of the GBSD, this can be written as:

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

By plugging in the per-message utility in Eq.(1) any values of $\hat{n}(T)$ and $\hat{m}(T)$ that verify this equality, 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., $\hat{m}(T) = \bar{m}(T) = E[M(T)]$ (see [8] for a justification of this choice). Then, we extract $\hat{n}(T)$ from the above equality and we replace in Eq.(1) to obtain the following per-message utility:

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

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. Unlike Eq.(1), this new per-message utility is a function of past history of messages and so can be calculated locally. It maximizes the average message delivery rate calculated over a large number of messages. Except when the number of messages is not large 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. This will be illustrated later by our simulation results.

4.2 Calculating estimators $\hat{n}(T)$ and $\hat{m}(T)$ for the average delivery delay per-message 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,

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

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

To evaluate our new scheme, we have added an implementation of the DTN architecture to the Network Simulator NS-2. This implementation includes (i) the Epidemic routing protocol with *FIFO* and *drop-tail* for scheduling and message drop in case of congestion, respectively, (ii) the RAPID routing protocol based on flooding (i.e. no side-channel) as described, to our best understanding, in [4], (iii) a new version of the Epidemic routing protocol enhanced with our optimal joint scheduling and drop policy (GBSD), and another version using our statistical learning distributed algorithm (HBSD). The VACCINE mechanism described in [16] is used with all solutions to "clean up" the network after message delivery[4]. In our simulations, each node uses 802.11b protocol to communicate, with rate 1Mbits/s. 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 two mobility patterns, a synthetic one based on the Random Waypoint model, and a real-world mobility trace that tracks San Francisco's Yellow Cab taxis [1]. Please see [8] for more details about how we used this trace.

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 85KB for a randomly chosen destination. Other message sizes were also considered but are not presented here due to space limitations. Unless otherwise stated, each node maintains a buffer with a capacity of 10 messages. 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. Note, that the evaluation of the HBSD policy requires to wait until the different nodes collect enough history to be able to calculate their estimators, and thus include an initial "warm-up" period before starting to account for HBSD. As a final note, the results presented here are averages from 20 simulation runs, which is enough to ensure convergence.

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

5.1 Performance evaluation for delivery rate

First, we compare the delivery rate of all protocols for the two scenarios shown in Table 2. Figures 1 and Figures 2 show the delivery rate for the Taxi trace for the case of both limited bandwidth and buffer, and the case of limited bandwidth and unlimited buffer, respectively. The number of sources is changed to cover different congestion levels.

Table 2. Simulation parameters

Mobility pattern:	RWP	Taxi Trace
Simulation's Duration(s):	5000	36000
Simulation' Area (m^2):	1500*1500	-
Number of Nodes:	40	40
Average Speed (Km/H):	6	-
TTL(s):	750	7200
CBR Interval(s):	200	2100

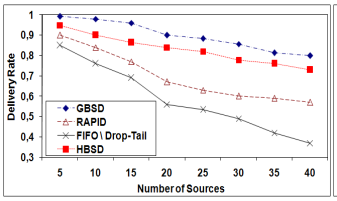

Figure 1. Limited buffer and bandwidth.

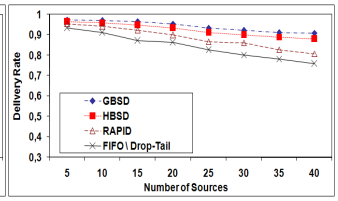

Figure 2. Unlimited buffer and limited bandwidth.

Table 3. Random Waypoint & unlimited buffer and limited bandwidth

Policy:	GBSD	HBSD	RAPID	FIFO\DT
D. Rate(%):	83	77	65	54
D. Delay(s):	519,75	532	682	775

Table 4. Random Waypoint & both limited buffer and bandwidth

Policy:	GBSD	HBSD	RAPID	FIFO\DT
D. Rate(%):	55	50	36	23
D. Delay(s):	1469,5	1507,47	1690,7	1970,45

From these plots, 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. For example, for 40 sources, and in the case of

limited bandwidth and buffer, HBSD's delivery rate is 15% higher than RAPID and only 6% worse than GBSD. Similar conclusions can be also drawn for the case of Random Waypoint mobility and 40 sources. Results for this case are summarized in Table 4 and Table 3.

5.2 Performance evaluation for delivery delay

We keep the same simulation duration and message generation rate as in Section 5.1. For the taxi mobility scenario, Figures 3 and 4 depict the average delivery delay for the the case of both limited buffer and bandwidth, and the case of unlimited buffer but limited bandwidth, respectively. 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 40 sources and both limited buffer and bandwidth, HBSD's average delivery delay is 17% better than RAPID and only 7% worse than GBSD. For the case of unlimited buffer and limited bandwidth, HBSD performs 13% better than RAPID and 8% worse than GBSD. Table 3 and 4 show that similar conclusions can be drawn for the delay under Random Waypoint also, with a gain up to 28% compared to RAPID.

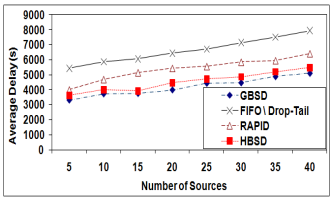

Figure 3. Limited buffer and bandwidth.

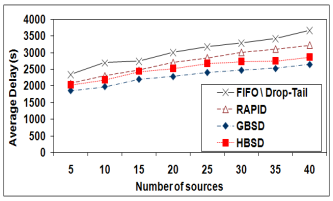

Figure 4. Unlimited buffer and limited bandwidth.

6 Conclusion and Future Work

In this work, we investigated both the problems of scheduling and buffer management in delay tolerant networks. First, we proposed an optimal joint scheduling and buffer management policy based on global knowledge about the network state. Then, we 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 a real mobility trace, we showed that our policy based on statistical learning successfully approximates the performance of the optimal algorithm in all considered scenarios. Finally, both policies (GBSD and HBSD) plugged into the Epidemic routing protocol outperform current state-of-the-art protocols like RAPID [4] with respect to both 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 policies that take into account different message sizes. The consideration of routing protocols other than Epidemic is also an interesting direction to explore.

References

[1] Cabspotting project. *http://cabspotting.org/*.

[2] Delay tolerant networking research group. *http://www.dtnrg.org*.

[3] 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*.

[4] A. Balasubramanian, B. N. Levine, and A. Venkataramani. DTN Routing as a Resource Allocation Problem. In *Proceedings of ACM SIGCOMM*, 2007.

[5] A. E. Fawal, J.-Y. L. Boudec, and K. Salamatian. Multi-hop broadcast from theory to reality: Practical design for ad hoc networks. In *Proceedings of ACM Autonomics*, 2007.

[6] Z. J. Haas and T. Small. A new networking model for biological applications of ad hoc sensor networks. *IEEE/ACM Transactions on Networking*, 14(1):27–40, 2006.

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

[8] A. Krifa, C. Barakat, and T. Spyropoulos. Optimal buffer management policies for delay tolerant networks. In *Proceedings of IEEE SECON 2008*, June 2008.

[9] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mobile Computing and Communication Review*, 7(3), 2003.

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

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

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

[13] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Efficient routing in intermittently connected mobile networks: The multiple-copy case. *ACM/IEEE Transactions on Networking*, Feb. 2008.

[14] M. V. Thomas Karagiannis, Jean-Yves Le Boudec. Power law and exponential decay of inter contact times between mobile devices. In *Proceedings of ACM/IEEE MobiCom*, 2007.

[15] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, 2000.

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