Timely Data Delivery in a Realistic Bus Network

Utku Acer[§], Paolo Giaccone[†], David Hay[‡], Giovanni Neglia[*], and Saed Tarapiah[†]
[§]Bell Labs, Antwerp, Belgium
[†]Dipartimento di Elettronica, Politecnico di Torino, Turin, Italy
[‡]School of Computer Science and Engineering, Hebrew University, Jerusalem, Israel
[*]EPI Maestro, INRIA Sophia-Antipolis Méditerranée, France

***Abstract*—WiFi-enabled buses and stops may form the backbone of a metropolitan delay tolerant network, that exploits nearby communications, temporary storage at stops, and predictable bus mobility to deliver non-real time information. This paper studies the problem of how to route data from its source to its destination in order to maximize the delivery probability by a given deadline. We assume to know the bus schedule, but we take into account that randomness, due to road traffic conditions or passengers boarding and alighting, affects bus mobility. We propose a simple stochastic model for bus arrivals at stops, supported by a study of real-life traces collected in a large urban network. A succinct graph representation of this model allows us to devise an optimal (under our model) single-copy routing algorithm and then extend it to cases where several copies of the same data are permitted.**

Through an extensive simulation study, we compare the optimal routing algorithm with three other approaches: minimizing the expected traversal time over our graph, minimizing the number of hops a packet can travel, and a recently-proposed heuristic based on bus frequencies. Our optimal algorithm outperforms all of them, but most of the times it essentially reduces to minimizing the expected traversal time. For values of deadlines close to the expected delivery time, the multi-copy extension requires only 10 copies to reach almost the performance of the costly flooding approach.

I. Introduction

A bus-based network is a convenient solution as wireless backbone for delay tolerant applications in an urban scenario. In fact, a public transportation system provides access to a large set of users (e.g. the passengers themselves), and is already designed to guarantee a coverage of the urban area. Moreover, such a wireless backbone is not significantly constrained by power and/or memory limitations: a throwbox can be easily placed on a bus and connected to its power supply, or can be put in an appropriate place in bus stops, which are usually already connected to the power grid to provide lights and electronic displays. Finally, travel times can be predicted from the transportation system time-table. Even if the actual times are affected by varying road traffic conditions and passengers' boarding and alighting times, such a backbone still provides strong probabilistic guarantees on data delivery time that are not common in opportunistic networks.

Given this scenario, this paper explores the basic question: *"how to route data over a bus-based network, from a given source to a given destination, so that the delivery probability by a given deadline is maximized?"*. We rely on the knowledge of bus schedule information and some stochastic characterization of bus mobility, supported by real data traces.

Most prior work exploits the contacts between the buses. In this paper we consider the alternative approach of relying only on bus-stop contacts and we study both single copy and multiple copies options to route packets to their destinations.

We start with a simple mobility model for buses (Sec. III) that is supported by the statistical analysis of a set of real traces of the public transportation system of Turin in Italy. This model allows us to represent the transportation system appropriately in terms of a graph with independent random weights, that we call the *stop-line graph* (Sec. IV). Under this representation, our original problem to identify routes maximizing the delivery probability by a given deadline (or maximizing the *on-time delivery probability*) becomes equivalent to a specific stochastic shortest path problem on the stop-line graph. We are able to find an optimal algorithm, called On-Time, for the single-copy case (Sec. IV-A) and then to extend it for the multi-copy case through a greedy approach (Sec. IV-C). We compare the performance of these proposed algorithms with three other heuristics (Sec. IV-B) that also operate on the stop-line graph: an adaptation of the routing algorithm proposed in [6] for bus-bus communications (we refer to it as Min-Headway), and the two naïve algorithms, Min-Delay, that determines the path with the least expected weight, and Min-Hops, that minimizes the number of times packets are forwarded until they are delivered to their destinations. Since the number of real-life traces we obtained is limited, the comparison (Sec. V) is based on simulations carried on a large set of synthetic traces generated on the basis of our bus mobility model and the schedule of Turin bus system.

In the companion technical report [1] our analysis is extended to the case when transmissions can fail.

II. Related Work

Most of the research on DTN routing has focused on bus-to-bus communications [2], [4], [5] with the following approach: Each vehicle learns at run time about its meeting process; then, the vehicles exchange their local view with other vehicles and use the information collected to decide how to route data. Unlike these studies, we mainly focus on *bus-to-stop data transfers* and derive a single-copy routing algorithm to maximize the delivery probability by a given deadline. We then extend the algorithm to address settings where several copies of the same data are permitted.

The use of fixed relay nodes was also considered in [3]. The authors report that the performance of a vehicular network is improved by adding some infrastructure, like base stations connected to the Internet, a mesh wireless backbone, or fixed relays (which are similar to our stops). They show that deploying some infrastructure has a much more significant effect on delivery delay than increasing the number of mobile nodes. These findings support our proposed architecture that relies on an opportunistic connectivity between vehicle nodes and fixed relays.

We observe that we use the bus network for data transfer as it is used for passenger transfer. Thus, one could expect that the same problem has already been addressed in the transportation literature (see [1] for more details). However, this is not the case: First, the possibility to exploit multi-copy is clearly absent in the transportation of people or merchandize. Second, the probability to miss a transfer opportunity is also not considered in transportation, while data transfer between two nodes may fail because of insufficient contact duration, channel noise or collisions. Third, even for single-copy routing, bus network passenger routes usually aim to *minimize the expected traversal time* (possibly limiting the maximum number of bus changes) and not to maximize the delivery probability by a given deadline.

In conclusion, to the best of our knowledge, this is the first paper that proposes an optimal routing algorithm that takes advantage of bus schedule information as well as a stochastic characterization of bus mobility, supported by real data traces.

III. Model Definitions and Assumptions

In this section, we formally define the terms and notation we use to describe a transportation system, following the terminology used in transportation literature.

A transportation system $\mathcal{T}$ has a set of stops, denoted by $\mathcal{S}$, and a set of vehicles (buses), denoted by $\mathcal{V}$, which travel between the stops according to a predetermined path and a predetermined schedule. For each vehicle $v \in \mathcal{V}$, the schedule allows us to determine its *trajectory*, denoted traj(v), which is the ordered sequence of stops the vehicle traverses: traj$(v) = (s_0, s_1, \ldots s_n)$. In addition, each vehicle v is associated with a *trip*, denoted trip(v), which is a time-stamped trajectory: trip$(v) = ((s_0, \tau_0), (s_1, \tau_1), \ldots (s_n, \tau_n))$, such that a vehicle v should arrive at stop s_i along its trajectory at time[1] $\tau_i = \tau(v, s_i)$. We distinguish between the scheduled time τ_i and the actual time $t_i = t(v, s_i)$, which is a random variable depending on road traffic fluctuations, passengers boarding and alighting, etc..

A key concept in bus networks is the notion of *lines*, which are basically different vehicles with the same trajectory (at different times). Let $\mathcal{L}$ denotes the set of lines. For each vehicle $v \in \mathcal{V}$ we denote its corresponding line by $line(v) = \{v' \in \mathcal{V} | \text{traj}(v) = \text{traj}(v')\}$. Note that lines introduce an important characteristic of a bus transportation system: if a passenger misses a specific vehicle v, she/he can still catch another vehicle v' in $line(v)$ and reach the same set of stops.

In the sequel, we will refer to the transportation system $\mathcal{T}$ as the quintuple $\langle \mathcal{S}, \mathcal{V}, \mathcal{L}, \tau(), t() \rangle$, where the function $\tau()$ is a way to represent the schedule and $t()$ denotes a characterization of the stochastic process of vehicle arrivals at the stops.

[1]We do not keep an explicit notation for the departure time of a bus from a stop (which is not given by our traces). Notice that departure times determine the duration of the transmission opportunities, however these are not important in our setting, which does not take into account bandwidth constraints.

The problem of maximizing the delivery probability by a given deadline requires a realistic statistical characterization of bus mobility patterns, which is also useful to generate a large set of synthetic traces and evaluate the performance of our routing algorithms.

We have performed a statistical analysis of a one-day trace, provided by Turin's public transportation company, with actual bus arrival times at their corresponding stops. Due to lack of space, the full details of this analysis appears only in the companion technical report [1]. Here, we only present the following two consequences of this analysis, and refer to them as Assumptions 1 and 2. These hypotheses are going to be kept for granted in the rest of the paper and will be fundamental to develop our routing algorithm.

Assumption 1: Bus travel times at consecutive stops are independent (but not necessarily identically distributed; in particular, their distribution will depend on the corresponding scheduled value).

Assumption 2: The distribution of the waiting time at a stop (when switching between buses of different lines) only depends on the stop and the characteristic of the *departing* bus line, not on the line of the arriving bus.

IV. Routing Algorithms in a Bus Network

As mentioned before, our routing algorithms aim to define an offline routing for the transportation system that maximizes data delivery probability by a given deadline.

Definition 1: Given a transportation system $\mathcal{T} = \langle \mathcal{S}, \mathcal{V}, \mathcal{L}, \tau(), t() \rangle$, a source stop s_s, a destination stop s_d, a start time t_{start}, and a deadline t_{stop}, the *on-time delivery problem* is to find a route between s_s and s_d that starts after time t_{start} and maximizes the on-time delivery probability; namely, $\Pr\{\text{data is delivered before time } t_{stop}\}$.

Next, we describe our *stop-line graph* for a transportation system. Note that in [1] we show that such a representation is needed and simpler representations are not suited for our problem. In the stop-line graph $G_{sl} = \langle V_{sl}, E_{sl} \rangle$, nodes are (s, ℓ) pairs, where s is a stop and ℓ is a line; $(s, \ell) \in V_{sl}$ if and only if line $\ell \in \mathcal{L}$ arrives at (or, equivalently, departs from) stop $s \in S$. In addition, we add two nodes $\hat{s}_s$ and $\hat{s}_d$ which are connected to all nodes that correspond to the source and destination stops. The edges of G_{sl} are defined as follows: An edge between (s, ℓ) and (s', ℓ') corresponds to a route between stops s and s' on line ℓ that continues from stop s' on line ℓ'. If $\ell = \ell'$ we call the edge a *travel edge*, while if $\ell \neq \ell'$ we call it a *travel-switch edge*. An example of G_{sl} appears in Fig. 1.

We now define the random variables associated to the edges in E_{sl}. The random variable of a travel edge describes the corresponding travel time between two stops: formally, a travel edge $e = ((s, \ell), (s', \ell))$ is associated with the random variable $w_e = tt(\ell, s, s')$ describing the travel time of a line ℓ bus from stop s to stop s'. The random variable of a travel-switch edge includes the travel time between the corresponding stops and the waiting time for the next line. Formally, a travel-switch

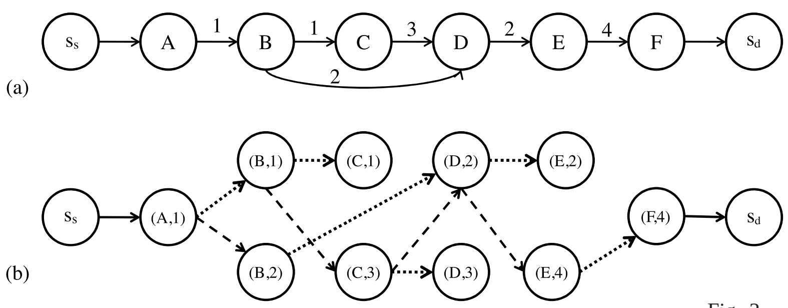

Fig. 1. (a) Example of bus network with $\mathcal{S} = \{A, B, C, D, E, F\}$ and $\mathcal{L} = \{1, 2, 3, 4\}$: the node corresponds to a stop and the label on the edge represents the line connecting the two stops. (b) The corresponding line-stop graph G_{sl}. Dotted edges are travel edges, while dashed edges are travel-switch edges.

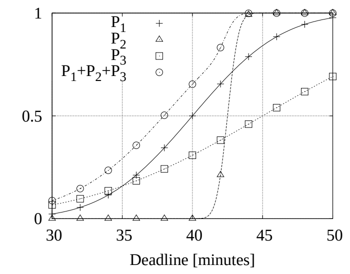

Fig. 2. Delivery probability CDFs of three disjoint paths P_1, P_2 and P_3, connecting a source and a destination with different traversal times. Path P_1 has the lowest expected traversal time; the variance of P_2 is the smallest, while P_3's variance is the largest. P_1, P_2 and P_3 are respectively the optimal paths computed by ON-TIME for deadlines between 34 and 43 minutes, larger than 43 minutes, and shorter than 34 minutes. The curve labeled $P_1 + P_2 + P_3$ corresponds to the success probability obtained by a multi-copy approach exploiting all the three paths.

edge $e = ((s, \ell), (s', \ell'))$ is associated with the following random variable w_e.

$$w_e = tt(\ell, s, s') + wt(\ell', s')$$

where $wt(\ell', s')$ is the waiting time at stop s' before the arrival of the next bus of line ℓ'. We assume that all the random variables defining w_e are known (they will be characterized in Sec. IV-A); moreover, by Assumptions 1 and 2, they are all independent[2].

A. Single-Copy Routing Algorithm and Implementation

We now turn to define our routing algorithm, called ON-TIME, which aims at solving the on-time delivery problem. ON-TIME finds, in general, different paths for different values of the (relative) deadline $t_{stop} - t_{start}$. For example, Fig. 2 compares the Cumulative Distribution Functions (CDF) for the delivery times of 3 different paths. In this case, ON-TIME chooses one of the three paths depending on the given deadline. Nevertheless, the larger the deadline, the larger the resulting on-time delivery probability is.

ON-TIME works by first determining a potentially good path between the source and the destination (for example, that with the minimum expected traversal time), and evaluating its on-time delivery probability. This can be done by performing a (numerical) *convolution* of the different random variables distributions along the path, yielding the end-to-end traversal time distribution. By this distribution, it is then easy to calculate (using the corresponding CDF) the delivery probability by the deadline.

Then, the algorithm proceeds by exploring the graph through a breadth-first search, looking for paths with a higher on-time delivery probability. A *pruning* mechanism avoids the need to determine and evaluate all the paths. Being that the traversal time is obtained by adding non-negative link weigths, for any path $\mathcal{P}$ and any prefix $\mathcal{P}'$ of $\mathcal{P}$, $\Pr\{tr(\mathcal{P}) \leq t\} \leq \Pr\{tr(\mathcal{P}') \leq t\}$. Thus, we can perform hop-by-hop convolution and compute, for each resulting distribution, the probability that the weight (that is, the traversal time) of this path's prefix is less than $t_{stop} - t_{start}$; if the probability is smaller than that of the current best path, there is no need to consider the rest of the path. From a practical point of view, working with a real transportation network, this simple pruning mechanism significantly reduces the number of paths to be considered, even if theoretically we may have a factorial number of paths to explore.

[2]In the technical report [1], we consider the case where transmissions might not be successful. In this case, edge weights depend also on transmission failure probabilities.

In our implementation, we have introduced some other simplifications, which reduce the computation time, but, at the same time, may lead to suboptimal paths. First, we have introduced a limit h for the exploration depth during the search. Given h as a constant, the algorithm is then guaranteed to run in polynomial time. We observe that upon termination, we are able to say if the algorithm has selected the optimal path or there may be a better one. In fact, when we stop, if there is still some path prefix of length not larger than h such that the pruning mechanism cannot discard it, then there could be a longer path with higher on-time delivery probability. But if this is not the case, then the current best candidate is actually the optimal path. In our experiments on Turin transportation network, $h = 8$ was enough to find all the best paths. Although this value may change for other networks, we think that it will remain a relatively small constant. Note that a suitable h for each network can be found by conducting experiments similar to ours.

A second simplification is that we restrict the set of eligible paths such that each line can be used only in consecutive edges. This prevents the algorithm to explore paths using line ℓ_1 then line ℓ_2, and then again line ℓ_1. We expect that these paths have normally worse performance than those where a data message just remains on line ℓ_1.

Finally, we have avoided the computation burden of performing numerical convolution by assuming that the end-to-end traversal time, which is a sum of independent random variables, can be approximated by a normal distribution. In this case, it is sufficient to take into account the mean and the variance of each edge weight (respectively, $\mu_e = \mathrm{E}[w_e]$ and $\sigma_e^2 = \mathrm{Var}[w_e]$). Then, the CDF of the traversal time of path $\mathcal{P}$ is equal to the CDF of a normal distribution with mean $\sum_{e \in \mathcal{P}} \mu_e$ and variance $\sum_{e \in \mathcal{P}} \sigma_e^2$. In the case of travel edges, average and variance of $tt(l, s, s')$ can be estimated directly on the traces. In the case of travel-switch edges, we have to also to evaluate the average and variance of $wt(\ell, s)$ using the first three moments of the interarrival times of the line ℓ buses to stop s (which can be also measured on the traces) and some basic Palm calculus.

In the rest of the paper, we evaluate the performance of ON-TIME for different source-destination pairs under similar kind of deadlines. If we had fixed a given deadline for all the pairs, then this deadline could be unfeasible for some of them (in the sense that there is no way to deliver the message by this deadline, e.g. if the deadline is smaller than the time a vehicle would take to move from the source to the destination), and trivially satisfiable for other pairs (many different paths would deliver with probability almost one). For this reason, given a source s_s, a destination s_d and a real value $x \in [0, 100]$, let $\phi(x, s_s, s_d)$ be the deadline t_{stop} for which the on-time delivery probability of the path from s_s to s_d with minimum expected traversal time is $x\%$. We denote by ON-TIME(x) the on-time routing algorithm where the deadline is set equal to $\phi(x, s_s, s_d)$ for every source-destination pair (s_s, s_d).

B. Other Routing Approaches

Although the algorithm we described is optimal under our model assumptions, we also consider sub-optimal but simpler heuristics.

The most intuitive approach (denoted as MIN-DELAY) is to route in G_{sl} along the path whose expected traversal time is minimal. Note that MIN-DELAY is equivalent to ON-TIME(50) under the Gaussian assumption on the distribution of the traversal time. Fig. 2 shows that path P_1, found by MIN-DELAY, does not always correspond to path with the highest on-time delivery probability. On the other hand, MIN-DELAY is computationally attractive, because the path with the least expected traversal time can be easily computed with Dijkstra's algorithm (by linearity of expectation).

A second algorithm, MIN-HOPS, selects the path that minimizes the number of times a packet is forwarded until the destination[3].

Another approach, denoted MIN-HEADWAY, tries to minimize the sum of all lines headways along a path [6], thus preferring frequent lines over infrequent ones; it was proposed originally for bus-to-bus communications. In Sec. V, we show that it has the worst performance in our settings among all the different algorithms.

C. Extension to Multi-Copy Routing

We consider multi-copy algorithms, such that at most k copies of the packets are made throughout the execution. Without this constraint a flooding scheme that can copy the data whenever there is a contact, namely in an *epidemic manner*, would achieve the best possible delivery probability.

We propose a greedy multi-copy algorithm for on-time routing, denoted simply as MC-ONTIME. It computes the on-time delivery probability of all paths in isolation and choose the k best paths (without considering the interaction between them). This can be easily implemented by saving the best k paths while enumerating all possible paths as in ON-TIME. Moreover, our pruning mechanism is changed accordingly to consider the k-th best value discovered so far (rather the maximum value as in the single-copy settings)[4].

However, since our algorithm works in a greedy manner, it does not consider the interaction between the paths, and more specifically the gain in probability over previously-selected paths (which can be very small in case the paths overlap). This leads to a theoretical performance degradation with respect to an optimal, infeasible algorithm that considers the joint-probability over all sets of paths. The following theorem, whose proof is in [1], provides tight bounds on this performance degradation:

Theorem 1: The MC-ONTIME algorithm always achieves at least $1/k$ of the on-time delivery probability of an optimal k-copy algorithm. In addition, there is a valid transportation graph for which MC-ONTIME achieves at most $\frac{1}{(1-\varepsilon)k}$ of the on-time delivery probability of an optimal k multi-copy algorithm, for arbitrarily small $\varepsilon > 0$.

V. PERFORMANCE EVALUATION

We consider a set of 180 source-destination (s_s−s_d) stop pairs. In the first 90 pairs both the source and the destination have been chosen uniformly at random in the entire metropolitan area. In the second 90 pairs, the source s_s is located in a main transportation hub within the city center (close to the main train station), and all the destinations s_d are chosen uniformly at random. We generate a set of 100 traces with the parameters obtained by the statistical analysis, covering all 250 lines for the four hours available from the schedule. In addition, we have developed a simulator that computes the delivery probability of each path by averaging across these 100 traces; note that the one day real-life trace alone would not be enough to compute this probability with any accuracy. Data is assumed to be available at the source stop at 7 AM.

We start to compare the performance of the algorithms defined in Sec. IV—namely, MIN-DELAY, ON-TIME, MIN-DELAY and MIN-HEADWAY—with the EPIDEMIC algorithm that floods the network by taking advantage of all the possible contacts (and therefore making a very large number of copies). We evaluate the actual on-time delivery probability of the best path obtained by each algorithm; for each pair s_s−s_d, we set the deadline to $\phi(x)$ for different values of x, and we compute the 90% confidence interval of the delivery probability considering all the possible 180 pairs. Due to the lack of space, we will report the results only for $x = 10$ ("short deadline") and $x = 50$ ("average deadline"), since these cases are representative.

Fig. 3 compares the delivery probability of the different algorithms for the two deadlines. The gain of EPIDEMIC with respect to all the other single-copy algorithms decreases as the deadline increases: EPIDEMIC achieves a delivery probability 3 times larger than ON-TIME for deadline $\phi(10)$, but only 1.5

[3]Note that this path also maximizes the delivery probability on an infinite time-horizon when packets experience a constant transmission loss probability. In the technical report, this approach is referred to as MAX-PROB.

[4]When comparing to the heuristics of Sec. IV-B, we can similarly get the k paths with minimal expected traversal time, total headway or maximal success probability.

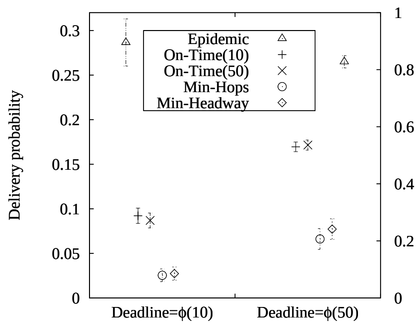

Fig. 3. Delivery probability for two deadlines and different routing algorithms. MIN-DELAY is the same as ON-TIME(50).

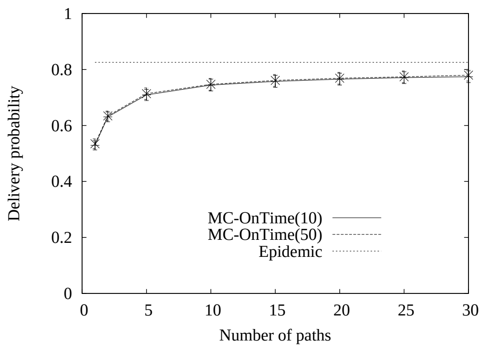

Fig. 4. Delivery probability (average and 90% confidence interval) vs. number of paths for deadline $\phi(50)$ and for multi-copy routing and no transmission failures ($p_f = 0$).

times larger for deadline $\phi(50)$. Indeed, when the deadline is large enough just one copy of the data is enough in order to reach 100% delivery probability. In such a case, EPIDEMIC does not introduce any gain in terms of performance, and the cost in terms of copies and transmissions is much larger than under single-copy algorithms. For example we observed on average more than 600 copies for $\phi(10)$ and more than 900 copies for $\phi(50)$ under EPIDEMIC up to the deadline, while for all single-copy algorithms the number of transmissions for each data is on average 5.0, and always less than 12.

ON-TIME(10) and ON-TIME(50) obtain the maximum delivery probability respectively, for deadline $\phi(10)$ and $\phi(50)$, as expected. But comparing the corresponding confidence intervals, they behave almost the same. A somewhat surprising results is that in many cases (121 out of 180) ON-TIME(10) performs *exactly* as ON-TIME(50) (or, equivalently, as MIN-DELAY). In fact we verified by direct inspection that ON-TIME(10) and ON-TIME(50) select exactly the same optimal path. These results have been confirmed also for other deadline values: The optimal route is not very sensitive to the deadline. In most of the cases the best path computed by ON-TIME(50) is the best for every deadline $\phi(x)$ with $x \in [0, 100]$. Recall the example in Fig. 2, showing that the best path does in general depend on the deadline. Our experiments lead us to conclude that these cases are very rare in a real transportation system. Thus, one can choose the path solely on the basis of the minimum expected travel time (that is, the simple MIN-DELAY algorithm), making it redundant to run the complex optimal algorithm ON-TIME.

We turn now to deal with multi-copy settings. Fig. 4 shows the performance of the MC-ONTIME(x) policy, that takes advantage of the k paths with the highest delivery probability by the deadline $\phi(x)$. The figure shows the results obtained for all the 180 source-destination pairs. For deadline $\phi(50)$, ON-TIME with one copy reaches a delivery probability which is about 66% of that achieved by EPIDEMIC, and a few more copies significantly reduces the performance gap. Yet, after 10 copies we observe only a negligible improvement. This is partially due to the fact that MC-ONTIME exploits a given sequence of paths provided by the algorithms, whose internal "diversity" is limited. Furthermore, EPIDEMIC exploits low-probability paths that are efficient just for the specific trace instance considered in each simulation run; since the number of these low-probability paths can be very large, due to the redundant connectivity of the bus transportation system metropolitan area, there is a high probability that at least one of them will be used to deliver to the destination. Note that the cost in terms of transmissions for EPIDEMIC is two order of magnitude larger than the multicopy approach using a pre-selected subset of 10 paths.

VI. CONCLUSIONS AND ACKNOWLEDGEMENTS

This paper lays the foundations for a framework to analyze bus-based networks, where communication is between the mobile buses and the stops along their trajectories. Through a statistical analysis of real bus traces, we were able to obtain a succinct stochastic graph representation of the system, and to devise optimal routing algorithms on this graph.

An important outcome of this study is that, although different from the optimal but computationally-intensive algorithm, the simple MIN-DELAY algorithm achieves excellent results in term of success probability for all deadlines. In addition, we show that increasing the number of data copies beyond 10 does not provide any meaningful performance improvement.

The research has been supported by INRIA Sophia-Antipolis Méditerranée as a COLOR Action and by Newcom++ NoE funded by the European Commission.

REFERENCES

[1] U. Acer, P. Giaccone, D. Hay, G. Neglia, and S. Tarapiah. Timely data delivery in a realistic bus network. Technical Report RR-7344, INRIA, 2010.

[2] A. Balasubramanian, B. Levine, and A. Venkataramani. Dtn routing as a resource allocation problem. *SIGCOMM Comput. Commun. Rev.*, 37(4):373–384, 2007.

[3] N. Banerjee, M. D. Corner, D. Towsley, and B. N. Levine. Relays, Base Stations, and Meshes: Enhancing Mobile Networks with Infrastructure. In *Proc. ACM Mobicom*, pages 81–91, September 2008.

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

[5] S. Gaito, D. Maggiorini, E. Pagani, and G. P. Rossi. Distance Vector Routing for Public Transportation Vehicular Networks: Performance Evaluation on a Real Topology. In *Proc. IFIP Wireless Days Conference*, 2009.

[6] M. Sede, X. Li, D. Li, M.-Y. Wu, M. Li, and W. Shu. Routing in large-scale buses ad hoc networks. In *Proc. of IEEE WCNC*, pages 2711–2716, 2008.