

# Performance limits of real delay tolerant networks

Alessandro Di Nicolò  
Dipartimento di Elettronica  
Politecnico di Torino, Italy  
Email: dinicolo@mail.tlc.polito.it

Paolo Giaccone  
Dipartimento di Elettronica  
Politecnico di Torino, Italy  
Email: paolo.giaccone@polito.it

**Abstract**—We consider a generic Delay/Disruption Tolerant Network (DTN) and we focus our investigation on assessing its performance limits. Our contribution is twofold. First, we propose a framework to evaluate the optimal performance in terms of minimum delay, maximum throughput and minimum number of hops, in the case of multi-hop transmissions. We consider both single and multi traffic flows scenarios. Second, we apply our framework to the case of a real DTN, that exploit the mobility of a public transportation system. We think that our contributions, even if based on some simplified assumptions, allows the network designer to evaluate the feasibility of supporting some applications, given their QoS requirements in terms of delay and throughput.

## I. INTRODUCTION

Delay (or Disruption) Tolerant Network (DTN) is an emerging packet switching paradigm, whose relevance has been highlighted in many recent papers [1], [2], [3], [4]. The interest for such paradigm is due to the fact that no fixed infrastructure is required to communicate between any nodes, even if the network is at any time partitioned. Data routing is based essentially on exploiting: (i) the cooperation among the nodes, which can transfer data through intermediate relays in multi-hop fashion, (ii) their physical mobility, which enables them to carry physically data from one location to another and meet the destination or some useful relay. Hence, the DTN paradigm can be summarized as “store, carry and forward”, in opposition to the traditional “store and forward”. In summary, DTN-ing can be considered an extension of classical mobile ad-hoc networks, but in which the network connectivity is sparse and thus nodes’ mobility is exploited for routing. This novel paradigm is technically very challenging due to nodes’ mobility and hence has attracted the interest of a large community of researchers.

Note that, due to the multi-hop approach, the forwarding delays depend mainly on the mobility process. In traditional packet networks delays are in the order of, at most, milliseconds or seconds, whereas in DTNs they can be minutes or hours, depending on the mobility speed ([5] shows that half day is a time constant typical in many examples of real mobility). We think that the term “delay tolerant” for DTNs is in some sense misleading, since there does not exist any application which can tolerate a very large delay. Furthermore,

there exist many contexts in which there are some guarantees on the mobility of the nodes, for example in the case of periodic mobility in public transportation systems. Hence, it is important to assess the performance, in terms of delays but also throughput, achievable in a DTN.

### A. Our contributions

In our paper we propose an optimization approach, described in Sec. III, to evaluate a limit on the absolute performance of a DTN in terms of maximum throughput and minimum delays. Thanks to these results, starting from the QoS requirements (bandwidth and delay) of an application, we can understand whether the considered DTN can potentially support it. If the QoS constraints can be supported by a DTN according to our limits, this does not imply that the practical sub-optimal algorithms proposed for routing (see the discussion on [4] for a recent overview of such algorithms) are able to satisfy such constraints; but the “distance” between our performance limits and the QoS requirements can help the designer to understand which is the best tradeoff between the performance provision and the implementation complexity of a practical routing algorithm.

Our approach focuses not only on the minimum delay and maximum throughput solution of the optimal routing problem, but also on the solution with the minimum number of hops. Minimizing the number of hops is useful not only to maximize the overall throughput in an environment when many sources are active, but also because it allows to save space in internal buffers of each node and to limit the transmissions. The latter implies to save battery energy and to reduce the radio interferences.

Note that in our work we do not consider the possible beneficial effects of network coding which has been recently considered an additional approach to improve the performance of distributed routing schemes in wireless networks [6], [7].

We assume that the nodes’ mobility is perfectly known, i.e. we know all the times during which any two nodes meet and the amount of data they are able to exchange; this is usually described by a contacts trace. This assumption is very strong, but in many practical cases a trace can be approximated by an estimation, for example when the node mobility is periodic (or almost periodic), as in the case of public transportation systems. In the case of human mobility,

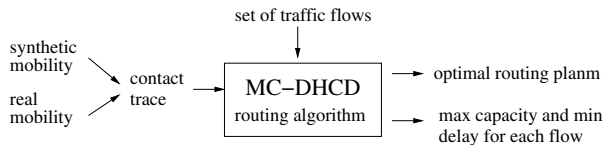


Fig. 1. Performance evaluation framework, based on optimization engine MC-DHCD

this assumption is unrealistic, even if some knowledge of periodic movements of people during the day (like: home, drive, office, drive, home) can be considered realistic (even if not always reliable). Note that the mobility trace can be obtained not only by measurements “on-the-field”, but also as output of any synthetic mobility model.

Fig. 1 explains all the input parameters and the output results of our approach; the main core is the MC-DHCD algorithm, which is an optimization engine based on a modified version of the classical Dijkstra algorithm and it will be described in Sec. III. Note that the model requires to know the set of traffic flows induced by the applications running on the DTN.

We apply this general framework in the case of a real DTN scenario, obtained by the contacts traces of the bus transportation system in a university campus. In Sec. IV we evaluate the performance achievable on such real wireless scenario. We compare these results with the performance achievable with other routing algorithms. The framework can be easily extended to consider additional QoS constraints (like the minimum/maximum bandwidth, or the maximum delay) and hardware constraints (like the maximum buffer size). As example, we will discuss the effect of limiting the internal buffer size at each node.

### B. Previous work

Many works have addressed the problem of assessing the asymptotic performance, i.e. how throughput and delays scale when the number of nodes grows to infinity. Pioneering results in [8], [9], [10] show basically that DTN networks can grow to large number of nodes and preserve (according to some scaling law) the performances seen by each individual node, provided that some assumptions on the nodes’ mobility are satisfied.

Other works have addressed the routing problem for finite networks and considered many variants regarding the knowledge of nodes’ mobility and the implementation of the routing, which can be centralized or distributed. We recommend the reader to refer to [4] for an updated review of these approaches. All these works focus on a relative performance comparison of routing schemes, in terms mainly of throughput and delays.

Our approach is based on the optimal routing algorithm for DTN proposed by [1], but it is different because our routing algorithm does not only minimize delays but also the number of hops. In addition, we consider a multi-flows scenario, in which we want to find the maximum performance achievable for a large set of traffic flows among nodes pairs. Our previous

papers [11], [12] present the main methodology to compute the maximum throughput, given the perfect knowledge of the contacts, but without taking into account delays and number of hops. We started from it to incorporate the delay and hop minimization.

Very recently, [4] has shown an optimal routing algorithm, based on hierarchical clustering, minimizing delays and the number of hops. Note that such algorithm is not end-to-end optimal like ours.

## II. SYSTEM MODEL

We say that two nodes meet when they are able to communicate each other (roughly, they are closer than their transmission ranges). When node  $a$  meets node  $b$ , a *contact* event from  $a$  to  $b$  occurs. For simplicity, we assume that contacts are atomic and are associated with a *contact time* and a *contact capacity*, which is the maximum amount of data which can be transferred during that contact. Note that, due to radio channel conditions, the capacities between two nodes can be asymmetric, as also observed in [13]. For simplicity, we neglect the radio transmission time of the data packet, even if it could be included at the cost of some new parameters in the model.

We model the set of all possible contacts through a directed multi-graph, as in [1] and denoted as *multi-contacts graph*. Each node is associated to a vertex and each contact event is associated with a direct edge connecting the corresponding nodes. If  $e$  is an edge,  $t(e)$  returns the contact time and  $c(e)$  (measured in bytes) the maximum amount of data that can be exchanged during the corresponding contact event. Note that the *contact graph* proposed in [11], [12] is a simple graph representing just the cumulative amount of data that can be transferred among the nodes and can be computed directly from the multi-contacts graph. Indeed, the weight of the edge from node  $u$  to node  $v$  is simply  $\sum_{e \in E(u,v)} c(e)$ , where  $E(u,v)$  is the set of all edges from  $u$  to  $v$  in the original multi-contacts graph.

A routing path  $P$  of  $H$  hops is described by a sequence of edges  $(e_h)_{h=1}^H$  and by its capacity  $r(P)$ , which is the amount of data sent across that path; note that this latter quantity can differ from the maximum transferable capacity, defined as the minimum capacity among all the edge capacities along  $P$ , because many flows could potentially exploit the same contact edge. The set of all routing paths and the corresponding capacities provides the global routing plan. This routing plan, to be feasible, avoids to overload any edge; formally,

$$\sum_{P \in \Omega(e)} r(P) \leq c(e) \quad (1)$$

being  $\Omega(e)$  the set of all paths across  $e$ . The total throughput achievable between a source node  $s$  and a destination node  $d$  can be computed as the sum of all the capacities of the corresponding routing paths, divided for the duration of the contacts trace. More formally, if  $\Gamma(s,d)$  returns the set of all routing paths from  $s$  to  $d$ , then the throughput  $G(s,d)$  of traffic

flow from  $s$  to  $d$  is simply:

$$G(s, d) = \frac{\sum_{P \in \Gamma(s, d)} r(P)}{T_{trace}}$$

being  $T_{trace}$  the trace duration.

For any single routing path  $P$ , we define also the corresponding delay  $d(P)$  according to the following rules. Data is generated by the source at some time and can be sent by one node to another only during one available contact time. Given the routing path  $P = (e_h)_{h=1}^H$  followed by the data, we define *network delay* as the interval of time spent inside the network (during the *forwarding phase*), starting from the time the source sends the data to the first node of the path and ending when the data reaches the destination; formally, this corresponds to  $t(e_H) - t(e_1)$ . Note that this definition is independent from the generation process of the data at the source, and takes into account only the contact times between the nodes. In addition, in the case of a single-hop route, the network delay is zero by definition. The *access delay* is instead the interval of time spent between the data generation and  $t(e_1)$  (during the *access phase*). The total delay is the sum of network and access delay. Fig. 2 shows a realization of network delay, access delay and total delay, in function of the time during which the data is generated. The network delay is a step function, whose value depends on the routing path starting from first available contact with the first relay node; the access delay is a decreasing function with 45-degree slope which becomes zero when the source meets the first relay node.

We evaluate the average delay  $D(s, d)$  between a source  $s$  and a destination  $d$  as the sum of all the total delays of the corresponding routing paths, weighted with their capacity. Formally,

$$D(s, d) = \frac{\sum_{P \in \Gamma(s, d)} d(P)r(P)}{\sum_{P \in \Gamma(s, d)} r(P)} = \frac{\sum_{P \in \Gamma(s, d)} d(P)r(P)}{G(s, d)T_{trace}}$$

Note that queueing happens both during the access and during the forwarding phases. We neglect the first, but the second is explicitly taken into account by our routing algorithm. Indeed, (1) does not allow to oversubscribe any edge along any path. Hence, when a routing path  $P$  is computed, the corresponding capacity  $r(P)$  is exactly the amount of data that can reach the destination after the delay estimated as described before. Note that it is possible to describe the exact dynamics of the buffer occupancy of all the nodes: this allows to optimize the routing also taking into account some internal buffer limitations.

### III. OPTIMAL ROUTING ALGORITHMS

Given a set of source-destination flows, to find all the optimal routing paths we have to solve a multi-commodity flow problem. As *DHC-optimal* (Delay Hop Capacity-optimal)

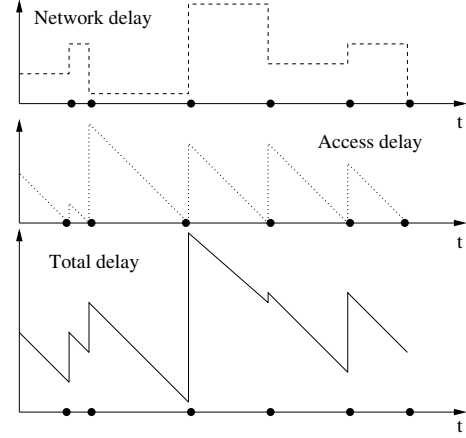


Fig. 2. Example of realizations of network, access and total delay, in the case of multi-hop communication. Black circles on time axis represent the instants of contact between the source and the first relay node.

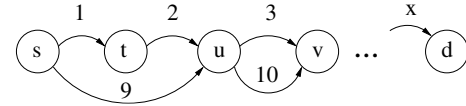


Fig. 3. Example of scenario showing that local decisions could lead to a suboptimal solution; edge labels refer to the corresponding contact times

routing path we mean a path between a source and a destination with minimum delay; when many paths at minimum delay are available, the path is chosen with the minimum number of hops; when many paths at minimum delay and minimum number of hops, the path with the maximum capacity is chosen.

In the following, we describe a family of algorithms, preliminary to the definition of the main algorithm, MC-DHCD. We report the corresponding pseudo-codes in Sec. III-A.

Given the source  $s$  and the destination  $d$  of a flow, finding the minimum delay path is quite immediate with algorithm *DD*, which is a simple modification of classical Dijkstra algorithm and was proposed in [1]. In a nutshell, during the breadth-first-search, *DD* chooses as best neighbor of node  $u$ , already visited and associated with a delay  $x$ , the edge  $e$  from  $u$  with the lowest exploitable contact time, i.e.  $t(e) \geq x$ . Unfortunately, *DD* is not suitable to compute the optimal path, in terms of delay and hops. The explanation for this is that, at each visited node, *DD* takes greedy decisions derived from local information. Because of the particular structure of the multi-contacts graph, the greedy strategy is not optimal, as shown in the following counterexample.

Consider the flow from  $s$  to  $d$  of Fig. 3. When updating the path from node  $u$  to  $v$ , a local decision must be taken: either using the contact at time 10 with a 2-hops path ( $s - u - v$ ) or using the contact at time 3 with a 3-hops path ( $s - t - u - v$ ). Any decision taken, an adversary can choose the contact time of some subsequent edge  $x$  to make this decision wrong for the global optimal solution.

We propose a novel algorithm, whose computational com-

plexity is asymptotically equivalent to *DD*, able to find the DHC-optimal path. The idea is to visit the graph in two directions, forward (from  $s$  to  $d$ ) and backward (from  $d$  to  $s$ ). First, *F-DHCD* (Forward optimal Delay Hops Capacity Dijkstra) finds a minimum delay path from  $s$  to  $d$ , while it tries, greedily, to minimize the number of hops and maximize the capacity. This path is not always DHC-optimal, for the reasons explained above, even if it is always delay optimal. Second, *B-DHCD* (Backward optimal DHCD) exploits the information regarding the delay-optimal path to prune the decision tree very efficiently and to build the DHC-optimal path.

Finally, algorithm *MC-DHCD* (MultiCommodity minDelay minHops maxCapacity Dijkstra) solves the global multi-commodity problem iterating the following two phases: (1) for each flow, it finds the DHC-optimal path between the corresponding sources and destinations, exploiting the sequence of *F-DHCD* and *B-DHCD*; (2) it allocates the capacity using a max-min fair allocation. The algorithm ends when no capacity can be allocated anymore. In summary, *MC-DHCD* is a simple greedy approach to find an approximation of the optimal solution, based on a DHC-optimal search of the paths.

For the sake of space, we do not describe the variants of *MC-DHCD* able to compute the optimal performance when some constraints on the minimum/maximum capacity, or on the minimum delay, or the maximum amount of buffer should be taken into account.

We devised also *MC-DD*, which is a variant of *MC-DHCD* in which the optimal path is computed directly with *DD* instead of the sequence *F-DHCD/B-DHCD*. This routing algorithm was useful to highlight the performance improvements due to DHC-optimal routing paths.

We implemented also *LP*, described in [11], [12], that computes the maximum throughput on the contact graph, by solving the associated maximum multicommodity flow adopting a linear programming (LP) formulation. Note that, by construction of the contact graph, LP does not take into account the contact instants and the delays associated for each path.

All these algorithms were implemented in C and in particular LP exploited GLPK [14] optimization libraries.

#### A. Pseudo-codes

In the following, we adopt the following notation.  $G = (V, E)$  is the multi-contacts graph described by the contact trace;  $Q$  is the set of all visited nodes;  $D$  is the node vector of the distances (in terms of delay) from the source;  $H$  is the node vector of the number of hops from the source;  $C$  is the node vector of the capacity allocated from the source;  $P$  represents the optimal path from the source  $s$ :  $P[v]$  returns the predecessor of vertex  $v$ .

<b>DD</b>
Input: $G = (V, E)$ , source $s$ , destination $d$ Output: optimal delay $D$ , optimal path $P$ $Q = V$ , $P[v] = \emptyset$ , $D[v] = +\infty \forall v \in V$ , $D[s] = 0$ // initialize <b>while</b> $Q \neq \{\}$ <b>do</b> // for all unvisited nodes $u = \arg \min_{x \in Q} D[x]$ // let $u$ be the closest node $Q = Q \setminus \{u\}$ // mark $u$ as visited <b>for each</b> $e \in E$ s.t. $e = (u, v)$ <b>do</b> // for each edge neighbor of $u$ <b>if</b> $(D[u] \leq t(e) < D[v])$ <b>then</b> // if the edge is an exploitable contact $D[v] = t(e)$ // update best delay from $s$ to $v$ $P[v] = u$ // update best path from $s$ to $v$
<b>F-DHCD</b>
Input: $G = (V, E)$ , $s$ , $d$ Output: optimal delay $D$ , optimal path $P$ $Q = V$ , $D[s] = H[s] = C[s] = 0$ // initialize $D[v] = H[v] = C[v] = +\infty$ , $P[v] = \emptyset \forall v \in V \setminus \{s\}$ // initialize <b>while</b> $Q \neq \{\}$ <b>do</b> // for all unvisited nodes $u = \arg \min_{x \in Q} D[x]$ , $Q = Q \setminus \{u\}$ // let $u$ be the closest node <b>for each</b> edge $e \in E$ s.t. $e = (u, v)$ <b>do</b> // for each edge neighbor of $u$ // if the edge is an exploitable contact by $u$ and $v$ <b>if</b> $(c(e) > 0$ and $D[u] \leq t(e) \leq D[v])$ <b>then</b> <b>if</b> $(t(e) < D[v])$ <b>then</b> // found better delay path // update best delay and number of hops from $s$ to $v$ $D[v] = t(e)$ , $H[v] = H[u] + 1$ , // update capacity and best path from $s$ to $v$ $C[v] = \min(c(e), C[u])$ , $P[v] = u$ <b>else if</b> $(H[v] > H[u] + 1)$ <b>then</b> // found equal delay, better hops path $H[v] = H[u] + 1$ , $C[v] = \min(c(e), C[u])$ , $P[v] = u$ <b>else if</b> $(H[v] = H[u] + 1$ and $\min(c(e), C[u]) > C[v])$ <b>then</b> // found equal delay, equal hops, better capacity path $C[v] = \min(c(e), C[u])$ , $P[v] = u$
<b>B-DHCD</b>
Input: $G = (V, E)$ , $P_F$ Output: best min-delay-min-hops path $P$ // initialize $Q = V \setminus \{\text{dest}(P_F)\}$ , $w = \text{second-last-node}(P_F)$ , $e = \text{last-hop-edge}(P_F)$ $D[v] = C[v] = 0$ , $H[v] = +\infty$ , $P[v] = \emptyset \forall v \in V$ s.t. $v \neq \text{dest}(P_F)$ $D[w] = t(e)$ , $H[w] = 1$ , $C[w] = c(e)$ <b>while</b> $Q \neq \{\}$ <b>do</b> // for all unvisited nodes let $u = \arg \max_{x \in Q} D[x]$ , $Q = Q \setminus \{u\}$ // let $u$ be the closest node <b>for each</b> edge $e \in E$ s.t. $e = (v, u)$ <b>do</b> // for each edge neighbor of $u$ <b>if</b> $(c(e) > 0$ and $t(e) \leq D[u])$ <b>then</b> // if the edge is an exploitable contact <b>if</b> $(H[v] > H[u] + 1)$ <b>then</b> // found better hops path $D[v] = t(e)$ , $H[v] = H[u] + 1$ , $C[v] = \min(c(e), C[u])$ , $P[v] = u$ <b>else if</b> $(H[v] = H[u] + 1$ and $\min(C[u], c(e)) > C[v])$ <b>then</b> // found equal hops better capacity path $D[v] = t(e)$ , $C[v] = \min(c(e), C[u])$ , $P[v] = u$
<b>MC-DHCD</b>
Input: $G = (V, E)$ , set of flows $N$ Output: set of paths $\{P\}$ and corresponding capacities $\{b_P\}$ 1: <b>for each</b> flow $n \in N$ <b>do</b> find a path $P_F$ with <i>F-DHCD</i> ( $G, \text{src}(n), \text{dest}(n)$ ) find a path $P_B$ with <i>B-DHCD</i> ( $G, P_F$ ) optimal path $P(n) = \text{min-hops-path}(P_F, P_B)$ <b>end for</b> <b>if</b> at least one path has been found for a flow <b>then</b> allocate the capacity according to the <i>max-min fair</i> criterion <b>else STOP</b> <b>goto 1</b>

## IV. TRACE DRIVEN PERFORMANCE

We start by describing the scenario under study and then we discuss the performance obtained adopting the routing algorithms described in the previous section.

### A. Scenario

We have exploited the available traces of the vehicular DTN testbed DieselNet [13], consisting of 40 buses operated by the UMass Amherst campus transportation system. The buses

communicated through IEEE 802.11 interfaces. The testbed was subject to the real schedule of the UMass campus. The publicly available traces refer to the radio contacts among buses, measured in terms of data transferred through TCP/IP connections, and refer to a period of almost 2 months.

We considered only the first two weeks of contact information present in the trace, since after two weeks the daily contacts become more and more sparse and we wanted to consider some stationary behavior. Furthermore, we excluded week-ends and nights. The length of the so modified traces is approximately ten days, with valid contacts among 30 buses. As consequence of such assumptions, if the algorithm calculates a delay of  $x$  hours, then, as a matter of fact, it is equal<sup>1</sup> to  $24\lfloor x/12 \rfloor + (x \bmod 12)$  hours in the real system.

We run two different kind of scenarios, the first is single-flow and the second is multi-flows. Within the single-flow context, we computed the performance of optimal routing for all the possible  $n(n-1)$  source-destination pairs, where  $n$  is the number of nodes. For the available trace,  $n = 30$  and hence we considered 870 flows.

Within the multi-flows context, a criterion is necessary for selecting the couples of nodes forming the set of flows. In our study we have defined two different permutation traffic scenarios: mDC (minimum direct capacity) and MDC (maximum direct capacity). These scenarios have been selected so as to respectively minimize/maximize the sum of capacities associated to the direct communication edges between sources and destinations; this selection has been computed running a minimum/maximum weight matching on the corresponding contact graph. As result, in the MDC scenario the couples of nodes forming the traffic flows will exploit mostly direct contact edges for exchanging data. In the mDC scenario, instead, since couples of nodes are selected so that the the sum of capacities associated to the direct communication edges is minimized, they will exploit mostly other nodes mobility for sending data over the network. Results on both the scenarios are qualitatively similar and, hence, we will discuss only the mDC results.

### B. Simulation results

In Fig. 4 we have calculated the overall throughput achieved by each possible couple of buses in the network and the relative mean delay, in the case of MC-DHCD algorithm and single-flow context. Note that we do not report the results obtained by MC-DD algorithm, which are almost identical to MC-DHCD, since the hop minimization affects the performance only in the multi-flows context. The maximum throughput ranges from 6 Mbytes to 2 Gbytes; recalling that the length of the trace is approximately 10 days, we obtain a throughput ranging from about 25 kbytes/hour for the worst couple to about 8.33 Mbytes/hour for the best couple.

Mean delays range from a minimum of 2 hours to a maximum of 93 hours (approximately four days). It is interesting to note that only 20 traffic flows (among 870 possible) experience

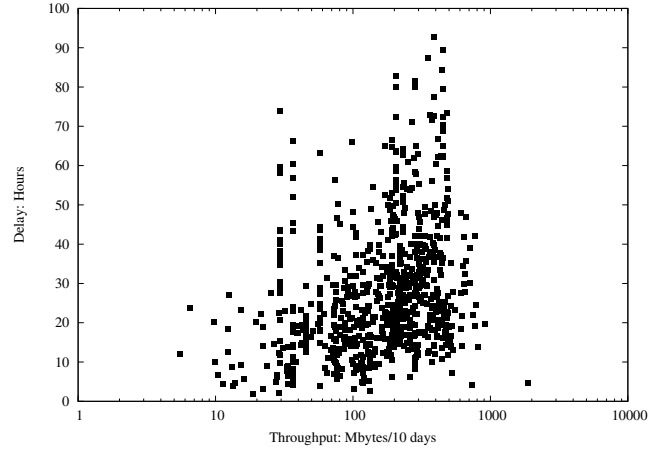


Fig. 4. Throughput and corresponding mean overall delay for all possible traffic flows in the single-flow scenario

a mean overall delay lower than 6 hours and only 70 lower than 10 hours. That means that only a limited set of traffic flows (less than 3-8%) experience average delays compatible with applications like the spreading of campus newspapers and advertisement messages, for which we can assume that the delays less than few hours can be acceptable. Note that these results, not very encouraging, have been obtained in the single-flow case, in which all the network nodes cooperate for the transmission of just one traffic flow.

An interesting aspect is that there are groups of bus couples characterized by identical throughputs but different mean overall delays. These groups of couples are marked out by the same source or destination nodes. Indeed, for all these groups a capacity bottleneck is present either at the source or at the destination, for example when a node meets few times just one other bus. This implies that all maximum throughput paths are affected by such bottleneck, whereas delays depend on the actual paths.

We consider now the multi-flows context, under mDC traffic scenario. Figs. 5 and 6 report the mean delay and the throughput achieved by the MC-DHCD in comparison with LP, in function of all possible traffic flows. Note that in all the following graphs, when showing some results in function of the flow identifier, for the sake of clarity, we sorted the values in increasing order and the  $x$ -axis refers to the relative ranking among the flows. The global throughput achieved by MC-DHCD is almost the 65% of the optimal LP solution. But delays are less than half of the delays for LP. Note that the absolute values of the delays appear quite large and only MC-DHCD shows some delays compatible with applications with daily spreading. These poor results are not surprising, because the DTN under consideration showed already limited performance in the case of the single flow scenario (Fig. 4), where most of the delays are around 20-40 hours and no contention occurs among different traffic flows for the network resources.

We compared performances of MC-DHCD and MC-DD

<sup>1</sup> $\lfloor x \rfloor$  is the higher integer  $\leq x$

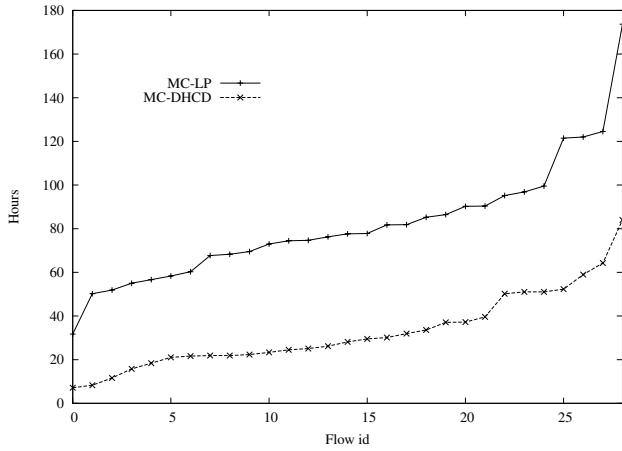


Fig. 5. Mean delay for each traffic flow in the multi-flows scenario

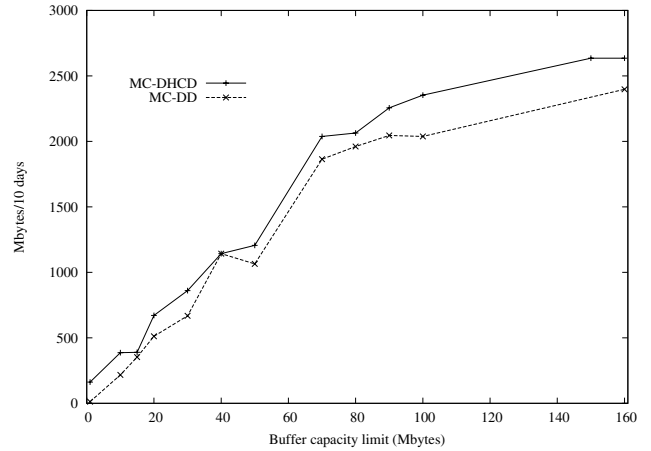


Fig. 7. Global throughput for buffered constrained routing

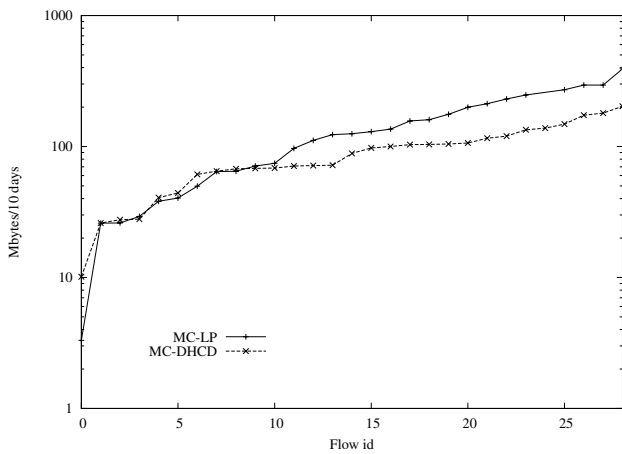


Fig. 6. Throughput for each traffic flow in the multi-hop scenario

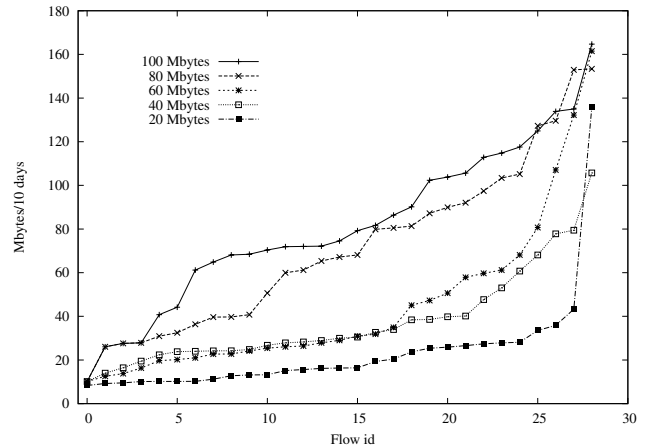


Fig. 8. Throughput for each flow and for different buffer size limit, in the case of MC-DHCD routing

and we do not report the details here for the sake of space. In summary, both obtained the same delays (as expected), but MC-DHCD allowed to increase the throughput of about 10% with respect to MC-DD; this result is corroborated by a decrease of about 15% in the mean number of hops, showing that MC-DHCD performs better by finding paths characterized by a lower number of hops under the same delay conditions.

In Fig. 7 we trace the global throughput achieved by each traffic flow, when the routing is buffer-constrained. For buffer capacity limit we mean the maximum amount of memory devoted to relay traffic and for not its own traffic. As can be seen, the buffer size of each node plays a fundamental role in the process of data delivery: reducing the buffer implies a decrease in the global throughput achieved by each flow. In this context, reducing the number of hops is crucial since it implies a lower buffer utilization and thus a better exploitation of the limited buffers. Neglecting the global number of hops when finding routes, as in MC-DD, has the undesirable twofold effect of (i) consuming communication resources, (ii) taking up a considerable amount of space in the buffers of relay nodes. Especially when the buffer constraint becomes

smaller, the relative gain in terms of throughput is larger; for example, MC-DHCD allows a gain factor of 12. Another important aspect can be inferred from the graph, related to the betweenness of nodes, i.e. the grade of centrality of the nodes in the routing paths. Indeed, there is unexpectedly a dramatic decrease of the throughput when the buffer becomes smaller than 50 Mbytes. This is because there are some buses that meet frequently all the other buses and thus play a central role in the routing. Thus, their buffer reduction strongly affects the global performance of the system.

Fig. 8 reports the throughput for each single traffic flow, obtained by MC-DHCD. The differences in throughput among the flows is more evident for larger buffer sizes, when the capacity allocation algorithm is more affected by the structure of the graph, whose inhomogeneous nature has been highlighted in [12]. On the contrary, for smaller buffer sizes, the max-min fair allocation in MC-DHCD allows to distribute quite fairly the paths to exploit the available buffers.

## V. CONCLUSIONS

We have proposed a performance evaluation framework to assess the performance limits of a DTN, in terms of minimum delay and maximum throughput achievable. The optimization engine (MC-DHCD) is based on a modified version of the classical Dijkstra algorithm and computes all the routing paths with the best performance and the minimum number of hops. This framework is implemented in a centralized fashion and requires the knowledge of all the contact events in the system; in this sense, it provides optimistic results. We applied it to the contact trace measured in the DTN of the bus transportation system of a university campus; even if evaluated optimistically, the performances were not very encouraging, since only few network nodes were shown to exchange data with reasonable delays and throughput. This is due to the initial size limitations of such pioneering (and technically very sound) testbed, whose preliminary mobility traces were the only available till few months ago. We believe that the new traces, which have been recently become available, will motivate our future investigations.

## REFERENCES

- [1] S. Jain, K. Fall, R. Patra, "Routing in a delay tolerant network, *ACM SIGCOMM*, 2004
- [2] C. Liu, J. Wu, "Scalable routing in delay tolerant networks, *ACM MOBIHOC*, 2007
- [3] X. Zhang, J. K. Kurose, B. N. Levine, D. Towsley, "Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing, *ACM MOBICOM*, 2007
- [4] A. Balasubramanian, B. N. Levine, A. Venkataramani, "DTN Routing as a Resource Allocation Problem, *ACM SIGCOMM*, 2007
- [5] T. Karagiannis, J.Y. Le Boudec, M. Vojnovic, "Power law and exponential decay of inter contact times between mobile devices", *ACM MOBICOM*, 2007
- [6] S. Chachulski, M. Jennings, S. Katti, D. Katabi, "Trading structure for randomness in wireless opportunistic routing, *ACM SIGCOMM*, 2007
- [7] J. Liu, D. Goeckel, D. Towsley, "Bounds of the gain of network coding and broadcasting in wireless networks, *IEEE INFOCOM*, 2007
- [8] M. Grossglauser, D. N. Tse, "Mobility increases the capacity of ad hoc wireless networks, *ACM IEEE Trans. on Networking*, 2002
- [9] S. Diggavi, M. Grossglauser, D. N. C. Tse, "Even one-dimensional mobility increases ad hoc wireless capacity, *IEEE ISIT*, 2002
- [10] A. El Gamal, J. Mammen, B. Prabhakar, D. Shah, "Throughput-delay tradeoff in wireless networks, *IEEE INFOCOM*, 2004
- [11] M. Garetto, P. Giaccone, E. Leonardi "On the Capacity of Ad Hoc Wireless Networks Under General Node Mobility, *IEEE INFOCOM*, 2007
- [12] P. Giaccone, M. Garetto, E. Leonardi, "On the effectiveness of the 2-hop routing strategy in Mobile Ad Hoc Networks, *IEEE ICC*, 2007
- [13] J. Burgess, B. Gallagher, D. Jensen, B. N. Levine, "MaxProp: routing for Vehicle-Based Disruption-Tolerant Networking, *IEEE INFOCOM*, 2006
- [14] GNU Linear Programming Kit, available at <http://www.gnu.org/software/glpk/>