Joint Scheduling and Buffer Management Policies for DTN Applications of Different Traffic Classes

Panagiotis Matzakos, Thrasyvoulos Spyropoulos, and Christian Bonnet

Abstract—Delay/Disruption Tolerant Networks target environments suffering from the instability or lack of end-to-end paths. Store-carry-and forward principle aims to sustain data sessions, and data replication to increase the probability of on-time delivery. However, these techniques require efficient scheduling and buffer management, to comply with limited resources availability (i.e., communication duration, storage). Multiple existing schemes aim to improve, or even optimize the resources usage. Nevertheless, their majority considers equally important application sessions. The few proposals considering different traffic classes, fail to provide real QoS guarantees. In this paper, we formulate the problem of maximizing the performance, subject to distinct QoS constraints (requirements) for each application class. We consider requirements related to delivery probability and delay. Then, we propose a distributed algorithm which: (i) guarantees satisfaction of the individual constraints, when this is feasible given the available resources, and (ii) allocates any remaining resources optimally, to maximize the desired performance metric. We first consider homogeneous mobility, and then extend our analysis to heterogeneous contact rates and sparse contact graphs, that better correspond to real life mobility. Simulation results, based on synthetic and real mobility scenarios, support our theoretical claims and show that our policy outperforms other existing schemes (i.e., ORWAR [1] and CoSSD [2]).

Index Terms—Delay/Disruption Tolerant Networks, Scheduling policy, Buffer Management policy, QoS provision.

1 Introduction

Delay and Disruption Tolerant Networking (DTN) aims to provide communication capabilities for a wide range of challenged environments, where there is a difficulty in establishing end-to-end paths. Such networking environments may be subject to connectivity disruptions due to sparse network topologies, terrain obstacles, nodes mobility or resource constraints (bandwidth per contact, storage, energy). In this context, the DTN community [3] has promoted the store-carry-and-forward routing paradigm: instead of dropping an end-to-end session (as TCP for example would do), nodes can store data content (their own, or their neighbors) persistently, and forward it to other nodes or the destination, when a communication opportunity appears.

In DTNs, end-to-end control feedback might be absent due to long delays and highly dynamic network topologies. As a result, discovering valid routes to content destinations is a challenging task which has dominated the research efforts for years. In this context, if we consider resource-unconstrained environments, data replication (e.g. [4], [5]) can increase the performance significantly both in terms of delivery ratio (i.e., data received/data sent) and delivery delay (i.e., time needed to reach the destination). However, the lack of constraints is rather unrealistic for most DTN settings (e.g. Energy/Buffer limited wireless sensor networks [6], vehicular networks with limited contact durations and/or buffer limited [7], military DTNs [8]). As a result, disseminating multiple replicas per message in a DTN network can increase dramatically the overall traffic load comparing to the available resources.

To this end, a lot of multiple copy routing protocols incorporate distributed scheduling and buffer management policies, in order to determine which data should be replicated during a meeting of limited duration with another node and which data should be dropped during a buffer congestion event [9], [10], [11]. Such policies aim to optimize scheduling and dropping decisions, by keeping track of important message parameters (e.g., number of replicas, remaining Time-to-Live (TTL)) and using them to estimate the probability of encountering the destination. Nevertheless, these schemes assume all end-to-end sessions (and, thus, messages) to be of equal importance.

This is generally not true. In many envisioned scenarios, network nodes might be running multiple applications in parallel. In this context, ensuring successful data delivery and/or minimizing the delivery delay may be more important for one DTN application than for another. Consider the example of a military operation where we have two applications launched concurrently at the DTN nodes: one reporting position information of friendly forces periodically and another one generating mission debriefings less frequently. We can consider that the delivery delay requirement for the first one is lower than the second one, since, after some time, a reported position may be stale. On the contrary, ensuring that a single mission debriefing message is delivered successfully may be more important than losing some (out of many) position updates. It is thus reasonable to assume that different messages might have different QoS

- *P. Matzakos, T. Spyropoulos and C. Bonnet are with Eurecom, Sophia-Antipolis, FRANCE*
  E-mail: {matzakos, spyropou, bonnet}@eurecom.fr
- *This work has been funded by the EDA MIDNET project [B 0882 IAP4 GC]; EURECOM acknowledges the support of its industrial members: SFR, Orange, ST Microelectronics, BMW Group, SAP, Monaco Telecom, Symantec, IABG.*

requirements and resource allocation decisions should take these into account.

Based on the bundle protocol [12], there is provision for three different QoS classes: Expedited (high priority), Normal (medium priority) and Bulk (low priority) by the DTN community [3]. More recently there has been an extension to support more priority levels within the Expedited class [13]. While such QoS classes provide a static characterization of different classes of messages, prioritization decisions among bundles belonging to different classes is an open issue. If we simply prioritize messages based on their QoS class, then applications belonging to lower classes would "starve" (i.e., they would always be the last to be scheduled and the first to get dropped), if resources are limited (which is most often the case).

A number of recent proposals attempts to address prioritization in a more elaborate manner. In the ORWAR protocol [1], Spray-and-Wait is used for routing [5], but a higher number of copies is assigned to bundles of higher QoS classes. Additionally, when it comes to dropping decisions, the higher QoS classes are always prioritized (dropped last) over the lower ones (assuming bundles of fixed size). Soares et al. [14] propose different queue sizes, proportional to each QoS class's priority. Dropping decisions of each queue are then independent of the others. In terms of scheduling, they propose that the contact window during a communication opportunity can be shared among different classes, again proportionally to their nominal priority (e.g. 60 % of time for expedited class, 30% for Normal and 10% for Bulk). However, there is no described mechanism on determining the dropping or scheduling sequence among bundles of the same queue.

All the aforementioned policies apply prioritization between QoS classes essentially by distributing the available resources (e.g. number of copies per class, available contact window, available buffer space per class), proportionally to the importance of each QoS class. However, this distribution is based on applying *fixed thresholds*. This raises a number of concerns. First, it is not clear how these thresholds could be tuned based on the environment in hand. Second, fixed thresholds cannot keep up with a dynamically changing DTN environment. Finally, depending on the availability of resources and threshold parameters, the behavior of the policy might be qualitatively different. E.g., if resources are not sufficient to satisfy all constraints, such policies still distribute resources proportionally, and might not satisfy the requirement of any class, not even the highest priority one. On the other hand, if resources are plenty, applying fixed thresholds might keep favoring higher classes unnecessarily (since the marginal utility of extra resources becomes small), and restrain lower classes from achieving a high performance.

To this end, the key contributions of this paper are to: (i) *formally define the problem of optimizing network-wide performance, either in terms of delivery delay minimization or delivery rate maximization, while satisfying individual class QoS constraints* (ii) *derive a distributed algorithm for this problem that adapts to the available resources* (iii) *provide a framework to address the challenges of applying this algorithm in real life mobility conditions*. We show that the optimal algorithm for the prioritization problem ends up adding appropriate penalty functions to the optimal utilities of the unconstrained problem of [15]. In this aspect, the work closest to ours is [2], that also extends the optimal utilities derived in [15], to support QoS based prioritization. However, their approach is a heuristic additive term which can neither ensure that QoS requirements are met nor that the resulting allocation of resources leads to optimal network-wide performance, as we will show.

A prerequisite for the fine operation of our algorithm is to be able to make accurate predictions regarding the delivery performance, on a per message basis. In this context, capturing the inter-contact time statistics between individual pairs of nodes adequately can increase accuracy. In a previous work [16], also focusing on optimal QoS prioritization, we consider that the pair-wise inter-contact times are independent from each other and can be modeled assuming *homogeneous* mobility scenarios: i.e., scenarios where all pairs of nodes meet with approximately the same rate. In the current work, we remove this assumption and make the necessary adaptations in our scheme to account for *heterogeneous* mobility scenarios, which correspond better to real life mobility conditions [17], [18], [19]. As we show through extensive simulations with real traces in section 4.2, the benefits in the performance of our scheme are significant.

We also propose an alternative approach to our basic scheme, accounting for scenarios where the message related information which is required for our policy (e.g. per message number of copies that exist in the network at a given time) cannot be available. This approach is based on Spray-and-Wait (SnW), instead of epidemic routing. The results of the comparison of this approach with our basic scheme are presented in section 4.2.3.

The rest of the document is organized as follows. In section 2 we present our QoS policy. We start from the description of our system model (section 2.1) and then formulate both QoS prioritization problems, as constrained optimization problems (sections 2.2-2.3). We propose a distributed resource allocation policy that is proven to be equivalent to a distributed gradient-ascent implementation of the solution (section 2.4). In sections 3.1-3.3, we derive some closed-form approximations to account for delivery predictions for real heterogeneous mobility scenarios. In section 4, we evaluate the performance of our scheme by comparing the obtained results: (a) based on different approximations (b) with the ones from other prioritization schemes. Finally, in section 5, we conclude the paper.

2 The QoS prioritization policy

2.1 Model description

In the following, we provide some details on our system model regarding our assumptions on mobility, data traffic, application QoS requirements and resource constraints, as well as our general framework with respect to data routing, scheduling and buffer management. In table 1 the notation that is used throughout the paper is summarized.

Mobility Model The impact of mobility on the delivery performance, and consequently the performance of our scheme, is determined by the distribution of the **pairwise**

residual inter-contact times: *time elapsed until the next contact of a pair of nodes starting from a random observation moment*. Indeed, in our framework, residual inter-contact times can model the distribution of time durations between a moment when scheduling or dropping decisions have to be made and the moment of the next meeting with a bundle's destination. In this context, we assume that there are N mobile nodes in the network and each node encounters other nodes according to a random "contact" process. Initially, we model the residual inter-contact times of this process through the exponential distribution with a common rate parameter $\tilde{\lambda}$. It has been shown that a number of mobility models and real traces correspond to contact models with approximately exponential tails [20], [21], [22]. Assuming that the nodes meet each other with approximately the same rate, $\tilde{\lambda}$ can describe the inter-contact patterns between different pairs accurately enough (**homogeneous pairwise contacts**). In section 3, we relax this condition to be more compliant with real life mobility scenarios.

Data traffic Model We consider each DTN node running C distinct applications concurrently. Each application generates autonomous data units that we will call bundles from now on, to be compliant with the DTN Architecture [23] and the associated bundle protocol [12]. The bundle generation is modeled through the Poisson distribution with mean rate λ_g per node per application class. We denote as $L_k(t)$ the number of distinct bundles of class k that exist in the network at time t. All application bundles have the same fixed size, which cannot be fragmented. Each individual bundle has a unique destination (unicast) and each transmission is considered successful, if it reaches its destination before expiry (i.e., within its Time-to-Live (*TTL*) interval).

Application QoS model We associate each distinct level of QoS performance (we will call it priority class from now on) with a specific value of the Bundle Delivery Ratio (BDR) (i.e., *Bundles received on time/Bundles sent*) or Bundle Delivery Delay (i.e., *elapsed time since bundle's creation, until one of its copies is delivered to the destination*). From a bundle's perspective, this requirement can be expressed as its minimum accepted delivery probability $P_{QoS}^{(k)}$, or maximum accepted delivery delay $D_{QoS}^{(k)}$, respectively. We note that each bundle can belong to a single QoS class.

Resource Constraints Our prioritization policy applies to DTN settings with limited buffer availability (b slots per node) and contact windows, comparing to the network traffic load. This load originates from bundle generations at the DTN nodes and bundle replications during node meetings. We model the amount of data which can be replicated within a contact window through the Poisson distribution with a rate parameter r_d.

Routing model, scheduling and buffer management framework We consider epidemic routing. When two nodes encounter, they aim to exchange their non-common bundles. If the contact opportunity is limited comparing to the amount of non-common bundles, the scheduling policy determines which bundles will be replicated. If the amount of replicated bundles leads to buffer congestion(s) at the recipient node, the buffer management policy determines which bundles will be dropped.

Notation	Description
N	Number of nodes in the network
T_i	Elapsed time up to current time since the creation of bundle i
$n_i^{(k)}(T_i)$	Number of copies of bundle i belonging to QoS class k after time T_i since the creation of the bundle
$m_i^{(k)}(T_i)$	Number of nodes who have "seen" bundle i belonging to QoS class k after time T_i since the creation of the bundle
$R_i^{(k)}$	Remaining Time To Live (TTL) for bundle i belonging to QoS class k
b	Number of buffer slots per node
C	Number of distinct QoS classes
$L_k(t)$	Number of distinct bundles of class k at time t
$P_{QoS}^{(k)}$	Minimum required probability of delivery for bundles of class k
$D_{QoS}^{(k)}$	Maximum accepted delivery delay for bundles of class k
$P_i^{(k)}(T_i)$	Probability of delivery for bundle i belonging to class k after time T_i since the creation of the bundle
$E[D_i^{(k)}(T_i)]$	Expected delivery delay for bundle i belonging to class k after time T_i since the creation of the bundle.
$\tilde{\lambda}$	Mean inter-meeting rate parameter (exponential distribution)
λ_{ij}	Meeting rate between node i and node j
σ_{λ}^2	Variance of meeting rates
p_s	Network's density coefficient
r_d	Rate of exchanged data per contact (poisson distribution)
λ_g	Bundle generation rate per node per application class (poisson distribution)

TABLE 1: Notation

2.2 QoS optimization for average delivery rate

As we have already highlighted, a good prioritization policy should: first, make sure that the QoS requirements of different application classes are satisfied; second, it should allocate the remaining resources, if any, in order to maximize the overall performance of the network. In the current section, we first formulate the prioritization problem for average Bundle Delivery Rate (BDR) maximization, given a set of different application QoS requirements that have to be satisfied. Then, we show analytically how we can obtain a distributed solution to this problem. Our formulation is one of a constrained optimization problem in the following form:

$$\max_{n_i^{(k)}} f(\boldsymbol{n}) = \max_{n_i^{(k)}} \sum_{k=1}^{C} \sum_{i^{(k)}=1}^{L_k(t)} (1 - exp(-\tilde{\lambda} n_i^{(k)} \cdot R_i^{(k)})), \quad (1)$$

$$g_k(n_i^{(k)}) = (1 - exp(-\tilde{\lambda} n_i^{(k)} \cdot R_i^{(k)})) \geq P_{QoS}^{(k)} \quad \forall i \in \text{class k}, \quad (2)$$

$$Nb - \sum_{k=1}^{C} \sum_{i=1}^{L_k(t)} n_i^{(k)} \geq 0 \quad \forall i \in \text{class k}, \quad (3)$$

$$N - n_i^{(k)} \geq 0 \quad \forall i \in \text{class k}, \quad (4)$$

$$n_i^{(k)} \geq 1 \quad \forall i \in \text{class k}, \quad (5)$$

Assuming message i of class k has $R_i^{(k)}$ remaining time-to-live, then if we maintain $n_i^{(k)}$ copies of it, the probability of it being delivered is the probability of one of its $n_i^{(k)}$ copies to encounter the destination before its TTL expires and it is given by $(1 - exp(-\lambda n_i^{(k)} R_i^{(k)}))$, from fundamental properties of the exponential distribution. Hence, the objective function (1) is the sum of the delivery probabilities of each individual bundle over all bundles and all classes. The objective function, denoted as $f(\boldsymbol{n})$, is concave on $n_i^{(k)}$.

The first constraint (2) expresses the per bundle delivery probability requirement ($P_{QoS}^{(k)}$), depending on which application class (k) it belongs to. This constraint is concave as well. Constraint (3) is linear and states that the total number of bundle copies should not exceed the total buffer space in the network (Nb)[1]. Constraint (4) ensures that each bundle should not have more copies than the total number of nodes (i.e., no node is allowed to have more than one copy). Finally, constraint (5) is there to make sure that a bundle should have at least one copy throughout its lifetime (i.e., the source of a bundle is not allowed to drop it before it expires).

Given that $n_i^{(k)} \in \mathbb{N}$, the above problem is an integer non-linear optimization problem, hard to solve optimally. However, we relax this condition, assuming $n_i^{(k)} \in \mathbb{R}^+$. The continuous relaxation of the problem leads to a convex optimization problem; it can be solved analytically using the method of Lagrange multipliers and KKT conditions [24, chapter 5], to derive a vector of $\mathbf{n}^*$ values that is *feasible*, i.e., ensures that the delivery probability of each message is at least as high as its class requirement, and *optimal*, i.e., $f(\mathbf{n}^*) \geq f(\mathbf{n})$ for all feasible $\mathbf{n}$[2].

However, the above solution requires a centralized implementation of bundle copies, which is not possible, since there is no central entity in DTNs that could control the state of all messages, instantaneously. Instead, each node only has access to its own buffer content. During a contact between two nodes, dropping a bundle from one buffer or copying a bundle to the node encountered will affect a single variable in the allocation vector $\mathbf{n}$. Hence, two nodes encountering each other can compare the bundles they have in their own buffers and make decisions independently of other nodes. The goal of these decisions should be to modify the allocation vector $\mathbf{n}$ towards increasing the objective $f(\mathbf{n})$. If we ignore the set of QoS constraints (Eq. (2)), such a distributed solution has been derived in [15], based on an enhanced bundle delivery probability expression, $P_i^{(k)}(T_i)$, used within the objective function (as opposed to Eq. (1)):

$$\left(1 - \frac{m_i^{(k)}(T_i)}{N-1}\right) \cdot (1 - exp(-\tilde{\lambda} n_i^{(k)}(T_i) R_i^{(k)})) + \frac{m_i^{(k)}(T_i)}{N-1}, \tag{6}$$

There, the objective is differentiated to get the "marginal gain of an extra copy for each message" (referred to as "message utility"):

$$U_i(DR) = \left(1 - \frac{m_i^{(k)}(T_i)}{N-1}\right) \cdot \tilde{\lambda} R_i^{(k)} exp(-\tilde{\lambda} n_i^{(k)}(T_i) R_i^{(k)}), \tag{7}$$

Note that, $m_i^{(k)}(T_i)$ term in Eq. (6) and (7) stands for the number of nodes who have "seen" bundle i (i.e., they have obtained one copy of it at some point, regardless of whether they still have it or not). This term accounts for the probability of one of these $m_i^{(k)}(T_i)$ nodes being actually the destination of bundle i. If a node ranks all bundles in its buffer according to this utility, and uses it to make drop or scheduling decisions, then during each contact, the improvement in $f(\mathbf{n})$ will be maximal among all feasible directions (a variable $n_i^{(k)}$ cannot change during a contact, if message i is not present in the buffer of any of the two meeting nodes); given the concavity of the objective, this method is shown to correspond to a distributed implementation of a gradient ascent algorithm [15].

Nevertheless, the above solution considers a single priority class only and does not provide any QoS guarantees. Our aim is to modify this distributed algorithm, in order to be able to first satisfy the set of constraints in Eq.(2), i.e., to find a "feasible" solution to the problem, and then to maximize the performance among all feasible allocations. In the context of constrained optimization problems, gradient ascent algorithms can be modified to include appropriate penalty functions for each violated constraint [25, chapter 23.5]. Thus, an enhanced objective function would have the following form:

$$\phi(\mathbf{n}^*) = f(\boldsymbol{n}^*) - \sum_{k=1}^{C} \sum_{i=1}^{L_k(t)} \psi_k(P_{QoS}^{(k)} - P_i^{(k)}(T_i)), \tag{8}$$

where $\psi_k(\cdot)$ is a penalty function related to the constraint for bundles of class k. We would like $\psi_k(x) = 0$, when $x < 0$, i.e, no penalty when the predicted delivery probability $P_i^{(k)}(T_i)$ for message i is large enough. However, we would like $\psi_k(x)$ to take very large values when the constraint is not satisfied ($x \geq 0$), imposing a large negative penalty on $\phi(\mathbf{n})$.

Based on this observation, we can maximize the above objective and solve the constrained version of the problem in a distributed manner, by using the following per message utilities:

$$U_i^{(k)}(DR) = U_i(DR) \cdot \left[1 + \max\{0, c_k(P_{QoS}^{(k)} - P_i^{(k)}(T_i))\}\right] \tag{9}$$

In other words, the utility of a message is equal to its unconstrained utility U_i of Eq.(7), if the predicted delivery probability is above the class requirement. Otherwise, this utility is incremented by a term proportional to the delivery probability deficit. c_k is a very large constant which ensures that the utilities of bundles that do not satisfy their constraint will always be higher than the utilities of bundles that do satisfy them (to ensure convergence to feasible solutions only).

As a result of these utilities, the bundles which are below their desired QoS threshold are always prioritized (i.e., dropped last, scheduled first) over the ones which are above. Furthermore, note that these utilities correspond to

1. We note that this constraint is valid if all the nodes in the network have equal buffer size b and IID mobility, as assumed in the basic framework. In the case of different capacities per node, a different constraint would be needed for each node, in the theoretical formulation. In practice however, due to the distributed nature of our problems, each node by definition will satisfy its own individual capacity even if it's different from other nodes.

2. In practice, one could round these values to the closest integer to get an approximately optimal solution.

differentiating the extended objective of Eq.(8) with $\psi_k(\cdot)$ chosen as an appropriately normalized quadratic penalty function. Hence, ranking and handling (e.g. dropping) bundles according to these utilities at every contact, guarantees: (i) eventual convergence to a feasible solution (i.e, satisfying the constraints), if there is one, and (ii) allocating any "extra" resources optimally, i.e., among all feasible allocations delimited by the constraints[3].

2.3 QoS optimization for average delivery delay

In the following section, we turn our attention to the QoS prioritization problem with respect to minimizing the average delivery delay. Thus, we proceed with the following formulation:

$$\min_{n_i^{(k)}} f(\boldsymbol{n}) = \min_{n_i^{(k)}} \sum_{k=1}^{C} \sum_{i^{(k)}=1}^{L_k(t)} E[D_i^{(k)}(T_i)], \quad (10)$$

$$g_k(n_i^{(k)}) = E[D_i^{(k)}(T_i)] \leq D_{QoS}^{(k)}, \quad \forall i \in \text{class k} : T_i < D_{QoS}^{(k)} \quad (11)$$

Similarly to section 2.2, the objective function (Eq. (10)) is expressed as the sum of the expected delivery delays over all bundles and classes. Based on the model of exponential inter-contact times, the expected time until one of the $n_i^{(k)}$ copies encounters the destination, considering also the $m_i^{(k)}$ nodes who have "seen" bundle i, can be approximated as:

$$E[D_i^{(k)}(T_i)] = \left(1 - \frac{m_i^{(k)}(T_i)}{N-1}\right) \cdot \left(T_i + \frac{1}{n_i^{(k)}(T_i)\tilde{\lambda}}\right), \quad (12)$$

where $D_i^{(k)}(T_i)$ stands for the delivery delay of bundle i belonging to class k and T_i is the already elapsed time since bundle's i creation. Based on Eq. (12), the new objective function is obviously convex on $n_i^{(k)}$. The constraint of Eq. (11) expresses the desired average delivery delay requirement ($D_{QoS}^{(k)} \leq TTL$) for bundles of class k and it is also convex on $n_i^{(k)}$. It is important to stress here that the constraint refers only to bundles whose elapsed time since creation is less than their threshold of delivery delay requirement (i.e., for a bundle i belonging to class k: $T_i < D_{QoS}^{(k)}$). This makes sense, if we consider that it is pointless to give higher priority to bundles which have already missed their delay target. Furthermore, such a policy would lead to wasting resources against other bundles which still have the possibility of being delivered before $D_{QoS}^{(k)}$. The rest of the constraints are the same with the delivery rate optimization problem (i.e., Eq. (3) - (5)).

As for the case of the delivery rate metric, the unconstrained message utility function (i.e., without considering the QoS constraints) for the delivery delay is derived in [15], as:

$$U_i(DD) = \left(1 - \frac{m_i^{(k)}(T_i)}{N-1}\right) \frac{1}{\left(n_i^{(k)}(T_i)\right)^2 \cdot \tilde{\lambda}}, \quad (13)$$

3. We note here that the above penalty function is not unique in achieving these goals. Other functions penalizing low predicted delivery probabilities sharply could suffice to implement a distributed ascent algorithm moving to feasible and better solutions. However, the priority given between constraints when the feasible domain is empty depends on the penalty function choice, as we shall see later.

where the number of nodes who have seen the bundle, $m_i^{(k)}(T_i)$, is also considered. Following the same approach with the one described in section 2.2, appropriate penalty functions can be introduced in an enhanced objective function $\phi(\mathbf{n}^*)$, to penalize each violated constraint:

$$\phi(\mathbf{n}^*) = f(\boldsymbol{n}^*) + \sum_{k=1}^{C} \sum_{i=1}^{L_k(t)} c_i(T_i) \psi_k(E[D_i^{(k)}(T_i)] - D_{QoS}^{(k)}), \quad (14)$$

where $c_i(T_i) = 1$ for a bundle i belonging to class k, when $T_i < D_{QoS}^{(k)}$ and zero otherwise, so as to determine whether bundle's i constraint violations are still considered or not. The penalty function $\psi_k(x)$, should take very large values when the expected delivery delay is higher than the class's threshold (i.e., $x > 0$) and zero otherwise (i.e., when $x \leq 0$).

According to the aforementioned rule, the distributed solution of this constrained optimization problem can be achieved by using the following form of per bundle utilities, $U_i^{(k)}(DD)$:

$$U_i(DD) \cdot \left[1 + c_i(T_i) \cdot \max\{0, c_k(E[D_i^{(k)}(T_i)] - D_{QoS}^{(k)})\}\right]. \quad (15)$$

Similarly to Eq. (9), c_k is a constant large enough to ensure prioritization of bundles that do not satisfy their constraint over bundles that do satisfy it.

2.4 Implementation of the scheduling and dropping policies

In the previous section, we have described a distributed QoS algorithm for the two constrained optimization problems in hand, and have provided theoretical support for its convergence to the desired solutions (i.e., optimal either in terms of delivery rate or delivery delay metric, conditionally on satisfying the requirements). Here, we show a simple implementation of this algorithm, and discuss some additional practical issues. Similarly to the previous discussion, the framework of the suggested implementation is the same for both optimization problems and the differentiation between them lies mainly on the distinct objective functions and the respective utilities and QoS thresholds.

We propose that the bundles residing inside a node's buffer (queue) can be separated in two dynamic groups, as shown in Fig. 1: the first group contains all bundles whose predicted delivery probability/delay hasn't reached the desired QoS threshold; the second group consists of bundles which have reached their threshold. In the case of delivery delay optimization, the second group includes also the messages whose elapsed time since creation is higher than the desired threshold (i.e., for $i \in$ class k, $T_i \geq D_{QoS}^{(k)}$). The bundles of the first group are always prioritized over the bundles of the second group. Ranking among bundles of the same group is based on the classic utility U_i. It is easy to see that the desired QoS message utility of Eq.(9) or (15) is monotonically decreasing from left to right in the queue of Fig. 1, and thus dropping bundles from the right and scheduling from the left of this queue implements the desired policy.

The above policy works fine if the network parameters (e.g. packet generation rate, available storage in the

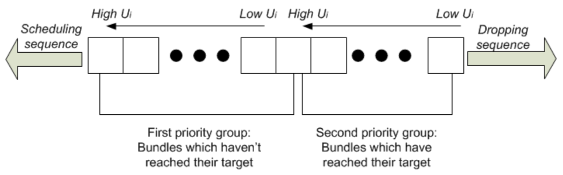

High U_i
Low U_i
High U_i
Low U_i
Scheduling sequence
Dropping sequence
First priority group: Bundles which haven't reached their target
Second priority group: Bundles which have reached their target

Fig. 1: Bundle scheduling and dropping sequence

network, inter-meeting rates) permit an algorithm to reach the desired delivery ratios, or average delivery delays, for all priority classes. However, in some scenarios this might not be possible, i.e., the feasible domain of the defined optimization problem is empty. It is somewhat subjective what a desired policy behavior should be, in that case. While one could apply a heuristic ranking in that case, or accept the (infeasible) solution the above policy converges to, in a number of cases we can modify the policy to provably achieve a desired outcome. We believe that an interesting class of cases is when it is more important to try to satisfy the constraints of the higher QoS classes first.

One could achieve this by choosing a different constant c_k in Eq.(9) or (15), for bundles of different QoS classes (k). Specifically, choosing $c_1 \gg c_2 \gg ... \gg c_C \gg 0$ (with 1 corresponding to the class with the highest nominal priority, and C the class with the lowest one) ensures a "smooth" fallback, if no feasible solution exists [4]. Specifically, if there is a feasible solution, the algorithm converges to it (as explained earlier). But if there is none, it converges to a solution where: for some $j < C$, QoS inequality constraints for all classes from 1 to j are satisfied with equality, class $j+1$ is not satisfied, and all classes larger than $j+1$ (if any) get no more resources than a single copy per bundle (based on Eq. (5))[5]. It is important to stress here that this algorithm *does not need to know in advance whether a feasible solution exists*. By construction, it navigates the infeasible domain so as to either enter the feasible domain eventually, or stop at an infeasible solution that is the most desirable one, according to the previous discussion.

The above algorithm can again be mapped into our simple buffer classification system by defining subgroups inside the first priority group (Fig. 2): each subgroup is composed of bundles of a particular priority class which are below their threshold (and, in case of delivery delay optimization, haven't missed their QoS target, $D_{QoS}^{(k)}$, yet). In this context, a subgroup attributed to a higher QoS class will always have higher priority than a subgroup of a lower QoS class.

At this point we should highlight an important aspect with respect to the implementation of our policy for the ADD optimization problem. So far, we have considered that the bundles can be kept in the node buffers even after their required delivery delay threshold has passed, but in this case they are downgraded to best effort (i.e., second priority group), in order to minimize the storage resources consumption for the sake of newer messages. Such an implementation choice can be meaningful in use case scenarios such as military applications, where the nodes can disseminate operational pictures or position updates. Then, although the faster these messages get delivered the more substantial they are, they might still be beneficial if they get delivered later (e.g., to discover a nodes trajectory). However, our policy can easily be adapted to an alternative implementation, where all messages get dropped after the elapse of the desired delivery delay limit. In this context, we consider both implementation options in our policy's performance evaluation (section 4.1) and examine the respective performance trade-offs.

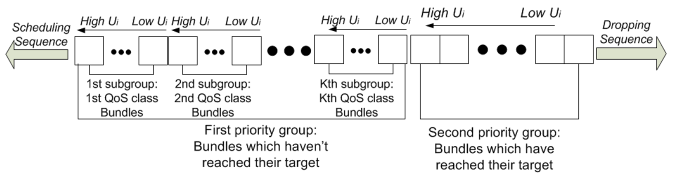

Scheduling Sequence
High U_i
Low U_i
High U_i
Low U_i
High U_i
Low U_i
High U_i
Low U_i
Dropping Sequence
1st subgroup: 1st QoS class Bundles
2nd subgroup: 2nd QoS class Bundles
Kth subgroup: Kth QoS class Bundles
First priority group: Bundles which haven't reached their target
Second priority group: Bundles which have reached their target

Fig. 2: An approach for bundle scheduling and dropping sequence for infeasible domains

As a final remark, the above algorithm requires reliable estimates for the number of copies and seen nodes, (i.e., $n_i^{(k)}(T_i)$, $m_i^{(k)}(T_i)$). This is not a trivial problem in a DTN setting. However, it is a problem that has already been addressed efficiently in [15], in the context of the "HBSD" policy. The authors there propose a distributed protocol to obtain estimates of these quantities, and show, using extensive simulations, that the communication overhead is reasonable and that the policy based on the estimate performs closely to one assuming instant knowledge.

Accordingly, the number of nodes, N, could also be reliably estimated based on either a similar protocol, as suggested in [15], or other distributed algorithms with reasonable overhead (e.g., [26]). In any case, the selected scheme running on top of our policy should ensure that: (i) it converges fast enough to a valid estimation on the current number of nodes, (ii) it ensures that all these nodes share a more or less common estimation. Alternatively, a simpler and faster approach could be used, which would be well aligned to our policy. According to our metric estimation and utility expressions (e.g., Eq (7), (12), (13)), we in fact need to estimate the ratios $\frac{m_i^{(k)}(T_i)}{N-1}$, instead of estimating the absolute value of N. Such estimates could be acquired in a distributed manner, having each node keep track of the percentage of encountered nodes that have "seen" a specific bundle out of all the nodes it has encountered. Assuming homogeneous contact patterns, this percentage would quickly converge to its mean value (due to the law of large numbers). Some additional modifications for the case of heterogeneous mobility could be further considered (e.g. exchanging this percentage estimates between nodes, and averaging appropriately, e.g. methods proposed in [26]).

4. Note that, in practice, it is not necessary to define a QoS threshold for the C^{th} class. Regardless of whether such a constraint is used or not, the corresponding bundles get additional resources, only if they are competing against higher class bundles which are predicted to satisfy their constraints. To this end, it doesn't make any difference if the C^{th} class bundles are classified at the C^{th} subgroup of the first priority group, or at the second priority group

5. While this starvation of low priority classes is undesirable, when enough resources are available to satisfy all classes, it can be argued that it's a desirable feature in emergency cases with very limited resources. Furthermore, other policies could be defined and achieved by manipulating c_k differently.

In the remainder of this text we assume that some sort of reliable estimation algorithm for the aforementioned quantities is implemented, and focus on the problem of reliably estimating the QoS constraints. Finally, in section 4.2.3 we briefly discuss an alternative "one-shot" approach, if no such reliable estimates can be obtained while the algorithm is running.

3 Considering heterogeneous contact networks

As highlighted earlier, a large majority of real mobility scenarios are characterized by heterogeneous pairwise contacts (i.e., some node pairs encounter more or less frequently than others). Furthermore, some pairs of nodes may never encounter each other. If we treat such scenarios as if all the pairwise inter-meeting times follow the same distribution (with rate $\tilde{\lambda}$), we might be lead to significant prediction errors that will degrade the performance of our scheme. Our analysis so far has been based on the assumption of homogeneous pairwise contacts. In the current section, we remove this assumption and make the necessary adaptations in our scheme, in order to extend its suitability for heterogeneous contact networks.

3.1 QoS optimization of delivery rate for heterogeneous contact networks

We still consider that the inter-meeting times between individual pairs are exponentially distributed; however, we now consider that the meeting rate of each individual pair is a random variable λ drawn from a probability distribution $f(\lambda)$, which is a characteristic of a mobility trace. Based on this, the probability of a bundle with $n_i(T_i)$ copies not being delivered, assuming that it has not been delivered yet, can be expressed as follows:

$$\begin{aligned} &P(\textit{bundle i will not be delivered/ has not been delivered yet}) = \\ &E_\lambda\Big[\prod_{j=1}^{n_i(T_i)} exp(-\lambda_j \cdot R_i)\Big] = \\ &\prod_{j=1}^{n_i(T_i)} \int_0^\infty exp(-\lambda_j R_i)\cdot f(\lambda_j) d\lambda_j \end{aligned} \tag{16}$$

Assuming that we do not know the exact distribution $f(\lambda)$, it is not possible to calculate the above probability. However, we can approximate it, based on the knowledge of the first moments of λ.

Our approach is similar to the one described in [27]. Specifically, if we consider Eq. (16) for $n_i(T_i) = 1$, we observe that it is the expectation of a function of a random variable λ (i.e., $g(\lambda) = exp(-\lambda R_i)$), and thus it can be approximated through the Delta method [28], [29]. Based on the Delta method, the expectation of a function of a random variable (i.e., $E[g(\lambda)] = E[exp(-\lambda R_i)]$) can be approximated through the Taylor expansion of the function and the first moments of the variable.

Thus, we first express $g(\lambda)$ as a Taylor series expansion of its first h terms, centered at $\tilde{\lambda}$:

$$g(\lambda) = \sum_{l=0}^{h} \frac{g^{(l)}(\tilde{\lambda})}{l!}\cdot(\lambda-\tilde{\lambda})^l \tag{17}$$

We approximate $g(\lambda)$ by considering the first three terms of the Taylor series (i.e., $h = 2$), corresponding to the first two moments of the random variable λ. We consider that the knowledge of these two moments is a realistic assumption. Then, the approximation on $E[g(\lambda)]$ can be derived after taking the expectation of Eq. (17), as follows:

$$\begin{aligned} E[g(\lambda)] &= \sum_{l=0}^{2} \frac{g^{(l)}(\tilde{\lambda})}{l!}\cdot E[(\lambda-\tilde{\lambda})^l] \\ &= exp(-\tilde{\lambda}\cdot R_i)\cdot\Big(1+\frac{R_i^2 Var(\lambda)}{2}\Big) \end{aligned} \tag{18}$$

where $Var(\lambda)$ is the variance of the meeting rates. The above expression is the probability of one of bundle i's copies not being delivered, given that it hasn't been delivered yet. Notice that, if we consider the first two instead of three terms of the Taylor series (i.e., h=1), the expression becomes the one used in section 2.2. To this end, we will refer from now on to the current approximation as **second order** and to the one of section 2.2 as **first order**, with respect to the utilized moments of variable λ. Note also that, given the convexity of $g(.)$ and, as the first order approximation is a pure function of $\tilde{\lambda}$, it is actually a lower bound on the expected probability of non-delivery, based on Jensen's inequality (i.e., $g(\tilde{\lambda}) \leq E[g(\lambda)]$). Intuitively, this means that the first order approximation is expected to give the most "optimistic" predictions, with respect to the delivery probability.

We can now express the unconditional probability of one of bundle's i, belonging to class k, $n_i^{(k)}(T_i)$ copies to be delivered, based on the second order approximation, as follows:

$$\begin{aligned} &E_\lambda[P_i^{(k)}(T_i)] = 1-\Big(1-\frac{m_i^{(k)}(T_i)}{N-1}\Big)E[g(\lambda)]^{n_i^{(k)}(T_i)} = \\ &1-\Big(1-\frac{m_i^{(k)}(T_i)}{N-1}\Big)exp\Big(-\tilde{\lambda}n_i^{(k)}(T_i)R_i^{(k)}\Big)\cdot \\ &\cdot\Big[1+\frac{\big(R_i^{(k)}\big)^2\cdot Var(\lambda)}{2}\Big]^{n_i^{(k)}(T_i)} \end{aligned} \tag{19}$$

It can be shown that the above function is concave on $n_i^{(k)}$. This allows us to redefine the objective (Eq.(1)) and the constraint (Eq. (2)) functions of the optimization problem in section 2.2 by substituting $P_i^{(k)}(T_i)$ with $E_\lambda[P_i^{(k)}(T_i)]$ of Eq. (19). We can also derive the new unconstrained utilities, as follows:

$$\begin{aligned} &U_i(DR) = \frac{\partial E_\lambda[P_i^{(k)}(T_i)]}{\partial n_i^{(k)}} = \\ &\Big(1-\frac{m_i^{(k)}(T_i)}{N-1}\Big)\cdot exp\Big(-\tilde{\lambda}n_i^{(k)}(T_i)R_i^{(k)}\Big)\cdot \\ &\cdot\Big(\tilde{\lambda}R_i^{(k)}-lnA\Big)\cdot A^{n_i^{(k)}(T_i)} \end{aligned} \tag{20}$$

where $A = 1+\frac{\left(R_i^{(k)}\right)^2\cdot Var(\lambda)}{2}$. Having the new expressions of delivery probability (Eq.(19)) and per bundle utility (Eq. (20)) in hand, we can apply them in our QoS prioritization algorithm, in the same manner we did for the homogeneous case (Eq. (9)).

3.2 QoS optimization of delivery delay for heterogeneous contact networks

Considering the probability distribution of the pairwise meeting rates, $f(\lambda)$, the expected delivery delay, $E_{\lambda}\big[D_i^{(k)}(T_i)\big]$, for heterogeneous contact networks can be expressed as follows:

$$\left(1 - \frac{m_i^{(k)}(T_i)}{N-1}\right) \cdot \left(T_i + \int_0^{\infty} \frac{1}{n_i^{(k)}(T_i)\lambda_j} \cdot f(\lambda_j) d\lambda_j\right) \tag{21}$$

Similarly to the case of delivery probability, the above expression can be approximated through the Taylor series expansion and the first moments of λ, as follows:

$$\left(1 - \frac{m_i^{(k)}(T_i)}{N-1}\right) \cdot \left(T_i + \frac{1}{n_i^{(k)}(T_i)\tilde{\lambda}}\left[1 + \frac{Var(\lambda)}{\tilde{\lambda}^2}\right]\right) \tag{22}$$

Then, the corresponding unconstrained utilities are derived by differentiating with respect to $n_i^{(k)}$:

$$\begin{aligned} U_i(DD) = -\frac{\partial E_{\lambda}\left[D_i^{(k)}(T_i)\right]}{\partial n_i^{(k)}} = \\ \left(1 - \frac{m_i^{(k)}(T_i)}{N-1}\right) \cdot \frac{1}{\left(n_i^{(k)}(T_i)\right)^2 \cdot \tilde{\lambda}}\left[1 + \frac{Var(\lambda)}{\tilde{\lambda}^2}\right] \end{aligned} \tag{23}$$

Thus, we can now substitute expressions 12 and 13 with 22 and 23 respectively, in the optimization problem defined in section 2.3.

3.3 Bounds on the expected performance

Although second order approximations are supposed to predict accurately enough the performance in terms of the metric of interest, it would be useful to know and exploit some bounds with respect to the expected performance. As already described in section 3.1, first order approximations can give us best case estimates (i.e., upper bounds for delivery probability, lower bounds for delivery delay). In the framework of our policy, though, it would be much more useful to derive bounds describing the worst case estimates. Indeed, such estimates would indicate that more resources are required to capture a given QoS threshold, thus ensuring the QoS constraints with higher consistency. Starting from convex functions of the random variable λ for both optimization problems, we can use the upper bounds derived in [30], based on the Edmundson-Madansky inequality [31] and the first h moments of λ:

$$EM_h(\lambda) = \sum_{i=0}^{h} \binom{h}{i} \frac{E[(\lambda - a)^i (d-\lambda)^{h-i}]}{(d-a)^h} f\left(a + \frac{i}{h}(d-a)\right) \tag{24}$$

where α and d correspond to the minimum and maximum values of λ, respectively, and $f(.)$ is our convex function. It generally holds that $E[f(\lambda)] \leq EM_h(\lambda) \leq EM_{h-1}(\lambda)$. Similarly to the case of the second order approximation, we consider the second order bound (i.e., h=2). Thus, by setting either $f = exp(-\lambda R_i)$, or $f = \frac{1}{\lambda}$, we can derive upper bounds on a single copy's expected probability of non-delivery, $EM_2^{DR}(\lambda)$, or delivery delay, $EM_2^{DD}(\lambda)$, respectively. These bounds are based only on the knowledge of the first two moments of λ and its min and max values. Then, the lower and upper bounds for the expected delivery probability and delay respectively, can be expressed as:

$$E_{\lambda}[P_i^{(k)}(T_i)] \geq 1 - \left(1 - \frac{m_i^{(k)}(T_i)}{N-1}\right) \cdot [EM_2^{DR}(\lambda)]^{n_i^{(k)}(T_i)} \tag{25}$$

$$E_{\lambda}\left[D_i^{(k)}(T_i)\right] \leq \left(1 - \frac{m_i^{(k)}(T_i)}{N-1}\right) \cdot \left(T_i + \frac{EM_2^{DD}(\lambda)}{n_i^{(k)}(T_i)}\right) \tag{26}$$

3.4 Considering sparse contact networks

Based on the previous analysis, our policy requires estimates on the first and second moments of the pairwise meeting rates, $\tilde{\lambda}$ and σ_{λ}^2, which characterize each mobility trace. However, for extracting those estimates, only the pairs of nodes $< i, j >$ that encounter at least once during the duration of the trace are considered.

Nonetheless, in real traces there is usually a large portion of node pairs that never encounter (i.e., $\lambda_{i,j} = 0$). In [27] it is shown that, for such sparse contact networks (corresponding to sparse contact graphs), the contact rate moments should be altered as follows:

$$\begin{aligned} \tilde{\lambda}_{p_s} &= p_s \cdot \tilde{\lambda} \\ \sigma_{\lambda_{p_s}}^2 &= p_s \cdot (\sigma_{\lambda}^2 + \tilde{\lambda}^2 \cdot (1 - p_s)) \end{aligned} \tag{27}$$

where p_s is a density characteristic of each trace, which can be approximated as the ratio of the total number of pairs that encounter throughout the trace, over the total number of distinct pairs that exist in the network $\binom{N}{2}$.

3.5 Discussion

Throughout our analysis, we have considered QoS requirements applied with respect to the "expected" values for the metric of interest. However, our framework could be appropriately adopted to support more generalized QoS requirements. Thus, an alternative policy can consider tighter constraints to ensure the requirements satisfaction. For example, one could impose that the delivery delay per bundle should be lower than its class's threshold, with a probability larger than 90 %, as opposed to the two implementation options discussed in section 2.4 which consider the expected delay. Of course there is a tradeoff there. The more conservative a policy is, the fewer the instances when it is actually missing the threshold, but the more the cases when it is wasting too many resources, just to ensure constraints satisfaction. The generalization of QoS requirements is beyond the scope of this paper, but, at the end, the design of a policy boils down to which is the primal goal: meet the constraints at all costs, or optimize the performance, while meeting the constraints, on average, and not wasting resources.

Another interesting problem, which we are planning to address in future work, could consider both traffic sensitive to delivery ratio and traffic sensitive to delay, concurrently. This would imply imposing both the respective types of constraints in our system. The solution to such a problem would be of practical use in many scenarios where applications having either type of requirements are launched at the same time.

4 Performance Evaluation

For the evaluation of our policy we examine the results obtained with both *synthetic* and *real mobility traces*. For the synthetic traces, we create an homogeneous contact network. The aim is to show how our policy complies with the intended optimal behavior, discussed analytically in section 2.2. To this end, we compare the performance of our policy with the approximate expected one, based on the resources availability, as well as with two other prioritization policies that we described in section 1, namely: ORWAR [1] and CoSSD [2]. Furthermore, we examine the impact of: generating different loads per traffic class, varying the traffic generation rate during the simulation scenario, as well as the impact of the number of network nodes. In terms of the ADD optimization problem, we evaluate and compare the performance of our policy based on the two different implementation options discussed in section 2.4. For the evaluation with real traces, we compare the results obtained with our policy when its implementation is based on the **first**, **second order** and **upper/lower bound approximations**. Then, we compare our scheme to the two aforementioned policies and an alternative approach that we propose based on using SnW, instead of Epidemic routing. In the following, we split the discussion on synthetic and real mobility traces. The configuration parameters for the three mobility scenarios are summarized in table 2 and will be discussed in the respective sections.

	Synthetic	*Cabspot.*	*Infocom*
Number of Nodes (N)	50, 100, 150	536	98
Total simulation time (sec.)	BDR opt.: 550, ADD opt.: $1.5 \cdot 10^4$	$2.64 \cdot 10^4$	$2.01 \cdot 10^4$
Mean pairwise meeting rate ($\tilde{\lambda}$, (sec^{-1}))	BDR opt.: 10^{-2}, ADD opt.: $2 \cdot 10^{-3}$	$6.9 \cdot 10^{-5}$	$3.8 \cdot 10^{-4}$
Pairwise meeting rates variance ($\sigma^2_{\tilde{\lambda}}$, sec^{-2})	0	$2.5 \cdot 10^{-9}$	$1.3 \cdot 10^{-7}$
Density coefficient (p_s)	100%	47%	68%
Bundle TTL (BDR Optimization problem, (sec.))	30	6000	3000
Bundle TTL (ADD Optimization problem, Exped. class (sec.))	500, 3000	15000	4000
Bundle TTL (ADD Optimization problem, Norm. class (sec.))	3000	15000	4000
Mean data exchange rate of contact window r_d (% of unconstrained rate)	0.5	0.5	0.5
Classes loads ratio (Expedited over other classes)	1, 0.25	1	1
Expedited desired QoS (BDR opt.)	0.77	0.74	0.77
Normal desired QoS (BDR opt.)	0.55	N/A	N/A
Expedited desired QoS (ADD opt, sec.)	500	3900	560

TABLE 2: Simulation Parameters (N/A stands for no applicable QoS requirements in the corresponding 2 class scenarios)

4.1 Homogeneous synthetic traces

4.1.1 Evaluation Setup

Initially, we consider three priority classes, namely Expedited (highest), Normal and Bulk (lowest) (based on the terminology of the bundle protocol specification [12], regarding different QoS classes). The BDR results are presented for various values of total available buffer space in the network, aiming to test our policy, as we vary the amount of buffer congestions. The bundles are created following a Poisson distribution with a fixed, or varying rate parameter per traffic class, throughout the simulations duration. The scheduling constraints are applied in our simulation framework as follows. The number of bundles which can be exchanged in every meeting between two nodes is drawn from a Poisson distribution with rate parameter r_d. In our scenarios, we set r_d equal to 50% of the measured average number of bundle exchanges per meeting, from the corresponding scenarios where we have the same mobility patterns and buffer constraints but where no scheduling restrictions are applied. The actual number of exchanged bundles per contact is then drawn from a Poisson distribution with rate parameter r_d. In this way, we restrict the number of bundles that can be replicated from one node to the other. Finally, as highlighted in section 2.1, the nodes meet each other with a common rate $\tilde{\lambda}$ based on the exponential distribution.

4.1.2 Results

Based on our previous descriptions, the intended behavior of our policy is to prioritize bundles in the order of their QoS class importance, when the available resources do not permit to reach the desired performance for all three classes (infeasible domain). In other words, under these circumstances, the first goal is to satisfy the Expedited class, then the Normal class and then the Bulk class. As a result, we expect from our policy to first reserve enough buffer resources to reach the performance target of the Expedited class and leave the remaining resources to the other two classes. This behavior is shown in Fig. 3 - 5, where we also compare our policy with the other two. To evaluate the performance of the three policies, each set of obtained BDR results, can be compared to the approximate desired performance (Table 3), when the buffer resources are distributed based on this logic. Obtaining such an approximate performance is straightforward, if we consider the expected number of copies per bundle required to reach a desired delivery probability and then map this to the total amount of required buffer space per QoS class. Thus, when the buffer availability is too low (i.e., 200 total spaces), corresponding to the infeasible domain, we expect not to have enough resources to satisfy even the Expedited class (i.e., $BDR = 0.77$). The other two classes should not get more resources, on average, than the ones corresponding to the minimum of a single copy per bundle throughout its lifetime (based on the constraint of Eq. (5)). This can be verified from Fig. 3 where the obtained BDR results per class comply with the expected ones. As the buffer availability increases (i.e., 300 total spaces), the first to converge to the required QoS is the Expedited class. Then, as we move towards the feasible domain (i.e., 300 - 400 total spaces), the other two classes

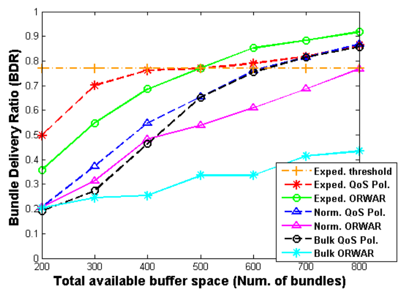


Fig. 3: QoS Policy vs ORWAR

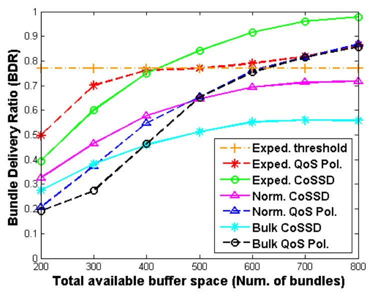


Fig. 4: QoS Policy vs CoSSD

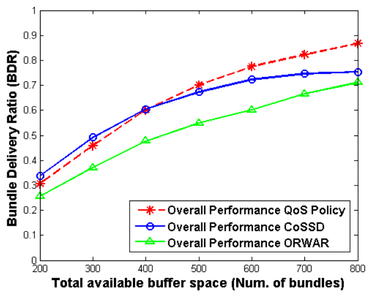


Fig. 5: Overall policies comparison

gradually improve their performance, with the Normal class achieving steadily higher performance than the bulk one.

Buffer spaces	BDR Exped.	BDR Normal	BDR Bulk
200	0.54-0.59	0.25	0.25
300	0.71 - 0.77	0.33 - 0.36	0.25
400	**0.71 - 0.77**	**0.54 - 0.59**	**0.41 - 0.45**
500	0.71 - 0.77	0.54 - 0.59	0.54 - 0.59
600	0.71 - 0.77	0.71 - 0.77	0.71 - 0.77
700	0.74 - 0.80	0.74 - 0.80	0.74 - 0.80
800	0.77 - 0.83	0.77 - 0.83	0.77 - 0.83

TABLE 3: Approximate desired performance when varying the total available buffer space

Inside the feasible domain (i.e., 400 - 800 total spaces) the additional resources are used to improve the performance of the lower classes and, as a result, optimize the overall network performance. This is depicted in the region 400 - 600 spaces of the figure, where the Normal and the Bulk class gradually converge to the performance of the Expedited class. Beyond that point, all of the classes achieve the same performance and exploit the complementary buffer spaces in order to further increase their BDR. We should highlight the fact that, overall, it is not until the point where the two lower classes reach the performance threshold of the Expedited class (600 - 700 total spaces), that the BDR of the latter is increased, which is the intended behavior that leads to optimal resources distribution.

Regarding the comparison with ORWAR, the protocol description [1] does not specify a precise way for selecting the spraying factors per class. Since, for our simulations, all bundles are of the same length, the scheduling/dropping policy is based purely on each bundle's QoS class, by always favoring the higher class bundles over the lower class ones. Thus, to compare ORWAR with our policy, we select the replication factors per class, based on the restrictions imposed by the total buffer availability in the network and by keeping a fixed ratio among replication factors attributed to distinct QoS classes. Particularly, given a total number of available resources, n_{all}, half of the resources $\lceil \frac{n_{all}}{2} \rceil$ are distributed to Expedited bundles and, among the remaining $n_{rem} = \lfloor \frac{n_{all}}{2} \rfloor$, $\lceil \frac{2n_{rem}}{3} \rceil$ are distributed to normal class bundles and $\lfloor \frac{n_{rem}}{3} \rfloor$ to bulk class bundles [6].

Based on Fig. 3, it is clear that our scheme outperforms ORWAR. For low buffer values (i.e., $<$ 500 buffer spaces), all three classes achieve higher BDR with our scheme. ORWAR fails to capture even the required performance of the Expedited class, even when the resources are adequate to do so (table 3). For higher buffer availabilities, ORWAR's expedited class reaches to higher BDR than the required threshold. However, this is not desired based on the previous discussion, as it comes at the cost of the other two classes, whose performance is much lower than it could be. The superiority of our scheme is also captured by the overall network performance (Fig. 5, considering all classes), which is up to 20% higher with our policy, comparing to ORWAR[7].

As described in section 1, the derived utility function in CoSSD is based on a heuristic approach to extend [15] for the support of multiple QoS classes. Particularly, it has the following form:

$$(C - k_i) + \alpha \cdot \left(1 - \frac{m_i^{(k_i)}(T_i)}{N-1}\right) \cdot \lambda R_i exp(-\lambda n_i^{(k_i)}(T_i) R_i), \tag{28}$$

where k_i is the QoS class of bundle i (lower values for higher classes), C is the total number of distinct QoS classes and α is a control parameter. To compare the performance of CoSSD with our policy, we set α equal to the value for which CoSSD achieves the intended per class BDR, on average (based on table 3), for total buffer size equal to 400 (border of the feasible region).

In Fig. 4, the results of the comparison with CoSSD policy are shown. Inside the infeasible region ($<$ 400) the lower classes, as well as the overall performance (Fig. 5), are improved comparing to our policy. However, this comes at the cost of significant performance degradation for the Expedited class, which does not manage to reach its required performance threshold for the first values of Buffer sizes ($<$ 400). This is obviously contrary to the intended behavior, which dictates that our primal goal is to reach the desired performance for the expedited class. The relative behavior between the two compared policies changes inside the feasible region ($>$ 400). The Expedited class's BDR for CoSSD increases beyond its desired QoS threshold, without the lower classes having reached this threshold. As highlighted for the comparison with ORWAR, this is opposite to the optimal behavior. The consequence is

6. We note that in the copies assignment rule, we also impose a minimum of 1 copy per bundle, for all QoS classes.

7. We should note that other static rules for assigning copies per QoS class could have also been used. However, we claim that any such static rule would suffer from similar shortcomings, comparing to our policy and the required performance.

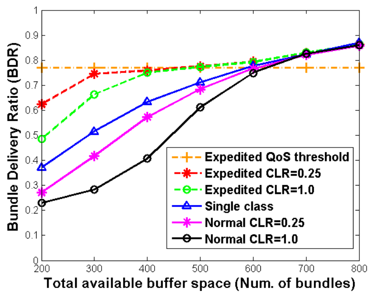


Fig. 6: QoS policy with different CLRs vs Single class policy

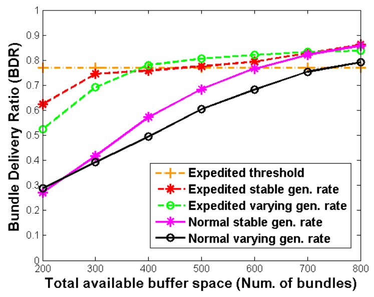


Fig. 7: QoS Policy: Stable vs varying traffic generation rate

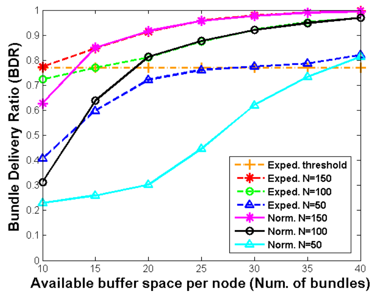


Fig. 8: QoS policy with different number of nodes and same generated data traffic load

that our policy outperforms CoSSD both in terms of lower classes (Fig. 4), as well as overall network performance (Fig. 5) in this buffer availability region.

In Fig. 6 the performance of our policy is evaluated, as we modify the relative class load ratio (CLR) among two traffic classes and compare it also to the optimal QoS unconstrained scheme of Krifa et al. [15] (single class policy). Thus, for our policy we consider the cases where, the poisson traffic generation rate of the Expedited class is one quarter of the Normal class's rate ($CLR = 0.25$) and the scenario where the two traffic rates are equal ($CLR = 1.0$). We note that the total generated traffic is equal among the three different scenarios. Thus, in the $CLR = 0.25$ case we have less amount of Expedited class bundles competing with each other on equal terms for being replicated, or avoid getting dropped, comparing to the $CLR = 1.00$ case. As a result, it should be expected that the Expedited class can satisfy its required QoS performance with fewer resources and leave more remaining resources for the improvement of the Normal class. The performance impact is verified in Fig. 6 where the Expedited class achieves better performance for $200 - 400$ buffer spaces and, accordingly, the Normal class performs better in the region $200 - 500$.

Regarding the comparison with the single class policy, it is evident that our scheme achieves better performance than the single class policy in terms of the Expedited class, but worse in terms of the Normal class in the region $200 - 500$. This indicates how our policy "sacrifises" the Normal class's performance to the required degree, so that the Expedited class can satisfy its QoS requirement. The benefit of using our policy in multi-class scenarios is further justified, by comparing with the $CLR = 0.25$ use case. Thus, our policy significantly outperforms the single class policy with respect to the Expedited class (max. by $\approx 25\%$), at the cost of the Normal class's performance. However, the difference between our policy's Normal class and the single class policy is not large (max. $\approx 10\%$) and it keeps decreasing, as the buffer space availability in the network increases, up to the point where the resources availability allow for the two classes convergence (buffer space = 500).

In Fig. 7 we evaluate the performance of our policy when we vary the total traffic generation rate throughout the simulation and compare this scenario to the one of stable traffic generation rate. Particularly, we divide the simulation time in 4 intervals, among which the total rate changes. In each interval, this rate is equal to either $1.5 \cdot \lambda_g$, or $0.5 \cdot \lambda_g$, where λ_g corresponds to the classic static rate scenario. Thus the average rate throughout the whole simulation remains equal to λ_g. We note that for both cases we set the $CLR = 0.25$ and that the BDR per class performance is extracted at the end of the simulation. It can be observed that, within the infeasible region of the variable rate scenario (buffer space < 400), the performance of the Expedited class is lower than the respective one in the stable rate scenario. This should be expected, if we consider that the BDR performance is averaged over: intervals where the resources are not enough to satisfy the respective QoS requirement given the heavy amount of traffic load (i.e., $1.5 \cdot \lambda_g$); intervals where the resources are enough to satisfy the QoS requirement given the lighter amount of traffic(i.e., $0.5 \cdot \lambda_g$), but our policy restricts the Expedited class performance from going above its threshold when the Normal class performance is lower. Accordingly, the Normal class's performance seems to improve even though the Expedited class has not reached its QoS requirement (on average), but this happens because of the intervals where the traffic load is low and our policy allows for this improvement. However, when we switch our interest to the feasible domain (i.e., buffer availability ≥ 500 allowing for the satisfaction of the Expedited class even with $1.5 \cdot \lambda_g$), it is shown that the Expedited class always reaches its required QoS threshold at the cost of the Normal class's performance (about 10% lower than in the stable rate scenario), highlighting the required policy behavior.

In Fig. 8 the performance of our policy is evaluated as we vary the number of nodes, and consequently the total available buffer space, but generate the same amount of total traffic in the network. Intuitively, we should expect that, as the number of nodes is decreased, the amount of congestions in the network increases significantly and, consequently, the overall performance is decreased as well. This is verified in Fig. 8 where it can be observed that the best per class performance is achieved with N=150 and the worst with N=50. However, despite the quantitative performance differences in the three scenarios, we notice that the intended qualitative behavior, is always captured. Thus the expedited class maintains its performance even with fewer nodes, at least when the total storage capacity leads to a feasible solution of the optimization problem. Of course this comes at the cost of the normal class's performance.

In Fig. 9 - 10 we focus on the performance of our

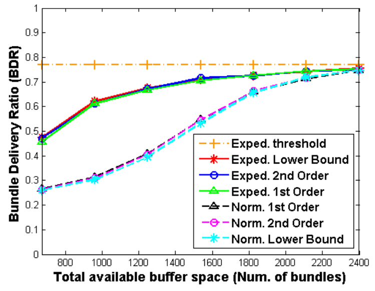


Fig. 11: Infocom: BDR Optimization, Approximations comparison

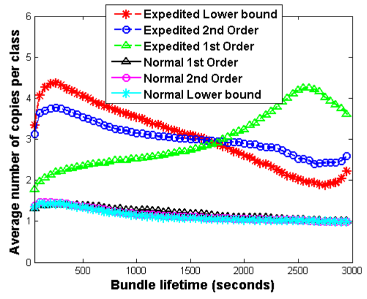


Fig. 12: Infocom: Copies over time, Approximations comparison (Buff= 10)

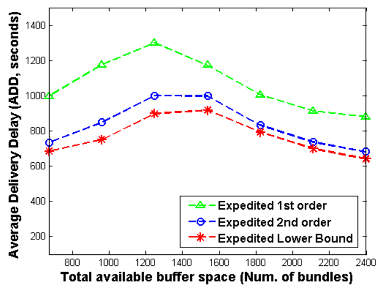


Fig. 13: Infocom: BDR Optimization, ADD metric Approximations comparison

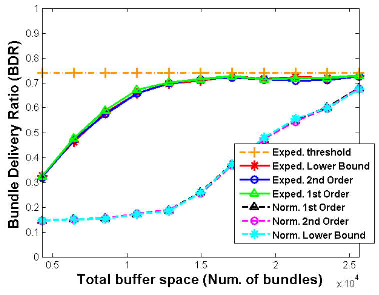


Fig. 14: Cabspotting: BDR Optimization, Approximations comparison

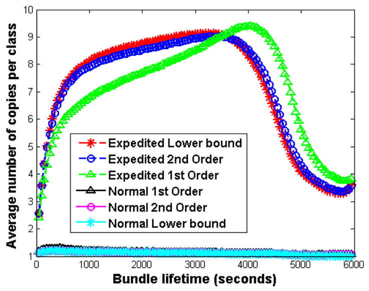


Fig. 15: Cabspotting: Copies over time, Approximations comparison (Buff= 20)

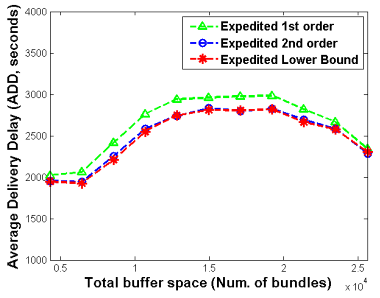


Fig. 16: Cabspotting: BDR Optimization, ADD metric Approximations comparison

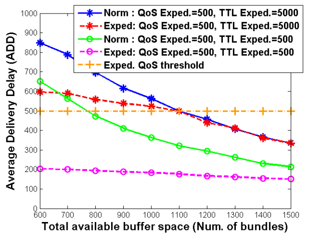


Fig. 9: ADD optimization performance with two different implementations of the QoS policy

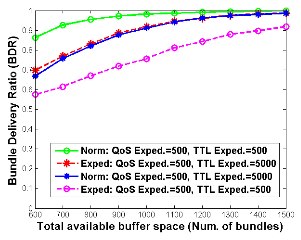


Fig. 10: BDR metric performance with two different implementations of the QoS policy

policy with respect to the ADD optimization problem. We consider two application classes and provide the results out of the two implementation options that were discussed in sections 2.4 and 3.5: *(i) the expedited bundles get dropped when their required QoS delivery requirement is exceeded (i.e., $TTL_{exped} = QoS_{exped} = 500sec.$), (ii) the expedited bundles TTL (3000 sec.) is larger than their QoS threshold*. We note that, for both scenarios, we set the Normal class TTL equal to 3000 sec. For the former case, it is shown in Fig. 9 that the Expedited class stabilizes its performance much below the desired threshold, as the results are extracted from bundles which were delivered before the QoS delivery threshold. For the latter case, the Expedited class gets stabilized close to the desired QoS threshold. For both cases, the Normal class gradually improves its performance, as the buffer availability in the network is increased and eventually converges to the Expedited class's performance, indicating the intended performance of our policy. However, the Normal class's performance is significantly improved in the first implementation option comparing to the second one, both in terms of ADD (Fig. 9) and BDR (Fig. 10). This occurs because, in this case, the Normal class bundles find much more available resources to use for replication, comparing to the second option where they find less ones due to the competition with the "longer living" Expedited bundles. Nevertheless, the first implementation option also comes at the cost of significantly lower BDR for the Expedited class than the second option. This indicates the performance trade-offs that should be considered, based on the application context where our policy is applied.

4.2 Real traces

For the evaluation with real traces we consider: 1. The Cabspotting trace which is based on tracking the movement of 536 taxis in San Fransisco [32], 2. The Infocom trace [33] originating from Bluetooth sightings of 98 nodes during 4 days in the Infocom 2006 conference.

4.2.1 Evaluation Setup

For both traces, we focus on time windows where the total number of meetings per hour does not change significantly. This time, we consider the competition among two distinct QoS classes (i.e., expedited and normal class). The performance is evaluated for both delivery rate and delivery delay optimization problems, as we vary the buffer spaces availability. As observed from the analysis of the Cabspotting and Infocom traces, the respective networks are not fully mixed (i.e., $p_s < 1$, table 2), within the duration of the time windows that we investigate. Based on this observation, the mean and the variance of meeting rates for each trace are extracted based on Eq. (27).

4.2.2 Comparison among approximations

In Fig. 11 - 22, we evaluate the performance of our policy when its implementation is based on the three different approximation approaches. In terms of the **BDR optimization problem** (Fig. 11 and 14), it can be verified that all the three approaches manage to achieve the intended optimal performance, as for the case of the synthetic traces. Particularly, for both traces, the Expedited class stabilizes its BDR around the desired value (i.e., 0.77 for Infocom, 0.74 for Cabspotting) and, as the buffer resources further increase, the normal class steadily improves its performance up to the point where the two curves converge. This behavior is captured ideally in the case of the Cabspotting trace (Fig. 14). In the case of the Infocom trace (Fig. 11), it can be noticed that, for low buffer availability, some of the Normal class bundles start getting delivered without the Expedited class having totally stabilized at its QoS threshold. This can be justified by the source copy restriction that we impose (Eq.(5)) even for Normal class bundles. Given the smaller size of the network comparing to the Cabspotting trace, the relative impact of a single copy on the delivery performance is greater in the Infocom trace. Thus, this restriction in combination with the limited resources availability, do not permit the Expedited class BDR to perfectly converge to the required value, before the Normal class starts increasing its own BDR.

The fact that no difference is observed in the performance of the three policies can be explained by inspecting Fig. 12 and 15. There, the average number of copies per bundle per class is drawn, throughout the bundle's lifetime for some fixed amount of buffer availability. It can be observed that, with the second order and lower bound approximations, more copies are distributed to Expedited class bundles at the beginning of their lifetime, as opposed to the respective copies distribution with the first order approximation. This comes as a result of the second order and lower bound approximations making more "conservative" predictions, with respect to the bundle delivery probabilities. Consequently, they indicate that more copies are required to reach the desired Expedited class performance. However, as described in section 2.2, the predictions are also based on dynamically monitoring the percentage of "seen" nodes $\left(\frac{m_i^{(k)}(T_i)}{(N-1)}\right.$ in Eq. (7) $\left.\right)$. When this percentage is low, with respect to the remaining TTL of the expedited class bundles and the desired QoS threshold, our policy can compensate the possible "over-optimistic" initial predictions, by distributing more copies to them, as they approach at the end of their lifetime. This behavior is more obvious in the case of the first order approximation for the Infocom trace (Fig. 12); It can be explained by the trace's higher heterogeneity with respect to pairwise contact rates, which makes it harder to make accurate predictions based on the first order approximation.

Although the delivery probability mispredictions of the first order approximation can be compensated in the manner we described, the same doesn't occur when we examine the ADD performance for the Expedited class (Fig. 13 and 16, yet still in the context of the BDR optimization problem). There, it is evident that the distribution of more copies at the beginning of the bundles lifetime by the two more conservative approximations permits to decrease the average delivery delay, comparing to the first order approximation. Intuitively, this makes sense if we consider that, by distributing more copies at the beginning of a bundle's lifetime, it is more likely that one of them will encounter the destination sooner. Thus, the usefulness of the better approximations is evident for the BDR optimization problem, as the results in terms of the delay metric are improved, without compromising something else. At this point, we should highlight that the ADD performance is extracted from the delivered messages only. This explains why, in some cases, the increase in buffer space availability is accompanied by an increase, instead of decrease, in the ADD performance. Particularly, when the delivery ratios are lower, it is more likely that the fewer messages that get delivered do so in relatively shorter time, than when they are higher. This also explains analogous behavior in the results of Fig. 17 and 20.

Let's turn our discussion now to the evaluation with respect to the actual **ADD optimization problem** (Fig. 17 - 22). For this set of results, we have selected the implementation option where a bundle's required delivery delay (560 seconds for Infocom, 3900 seconds for Cabspotting trace) is different from its TTL, which is actually much larger (i.e., 4000 sec. for Infocom, 15000 sec. for Cabspotting) to permit for delivery probability close to 100%. In Fig. 19 and 22, the BDR per class performance of the two traces is depicted, for the three different approximation approaches. Once again, the second order and upper bound-based implementations of our algorithm distribute more copies to the Expedited class bundles at the beginning of their lifetime (Fig. 18 and 21)[8]. Contrary to the BDR performance of the respective optimization problem though, this has a crucial impact on the performance of our scheme. The two more conservative approximations manage to capture the QoS requirement of the Expedited class better, for all the range of buffer values (and both traces), as opposed to the first order approximation, whose mispredictions do not allow to do so (Fig. 17 and 20). Of course, the distribution of more copies to the Expedited class, leaves less resources to the Normal class and, as a result, the latter's performance is better with the first order approximation than with the other two, for a range of buffer values. However, once again we verify the intended behavior of the algorithm: *as the buffer availability increases, the delivery delay of the normal class is constantly decreasing with the conservative approximations, while the Expedited class remains stable around its threshold*.

Regarding the comparison between the second order and (upper/lower) bound approximations, for both optimization problems, the following can be observed. For the Infocom trace, it is evident that the ADD of the Expedited class with the upper bound is constantly below the one with the second order approximation for all the evaluated scenarios (Fig. 13 and 17). As explained in section 3.3, this is foreseen, as the predictions with the bound approximations are expected to be more conservative than the ones of the second order approximation (Fig. 12 and 18). For

8. We note that the steep decrease in the number of copies of the expedited class for $T_i > D_{QoS}^{(k)}$ is dictated by our algorithm which imposes that copies of bundles which have exceeded their QoS threshold cannot be classified in the high priority group (section 2.3).

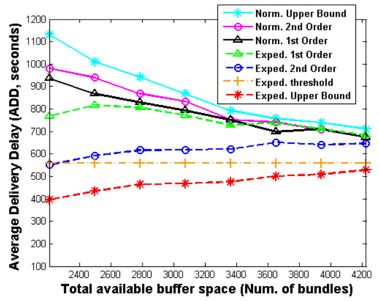


Fig. 17: Infocom: ADD Optimization, Approx. comparison

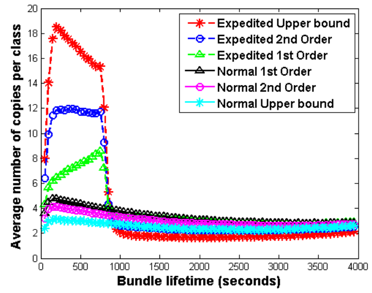


Fig. 18: Infocom: Copies over time, Approx. comparison (Buff= 23)

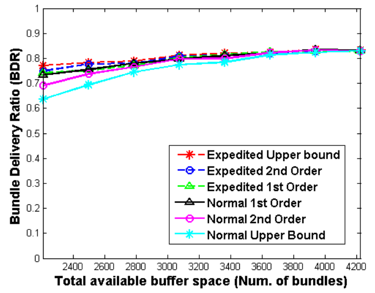


Fig. 19: Infocom: ADD Optimization, BDR metric Approximations comparison

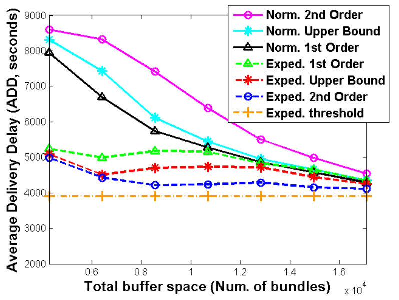


Fig. 20: Cabspotting: ADD Optimization, Approx. comparison

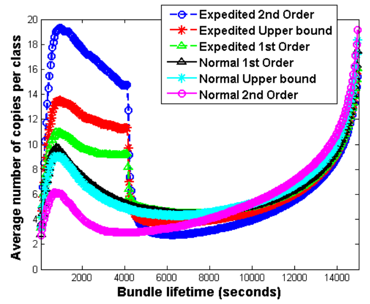


Fig. 21: Cabspotting: Copies over time, Approx. comparison (Buff= 20)

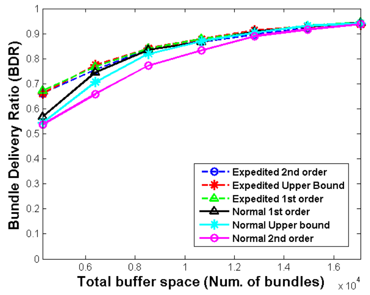


Fig. 22: Cabspotting: ADD Optimization, BDR metric Approximations comparison

the Cabspotting trace, though, the performance differences between the second order and bound approximations are smaller and, for the case of the delivery delay optimization problem, the ADD with the second order approximation is even slightly lower than the respective one with the upper bound (Fig. 20). Although not expected, this can be due to the distribution of meeting rates during the selected time window of the trace, which leads to more optimistic estimations of the delivery delay than the second order approximation (Fig. 21).

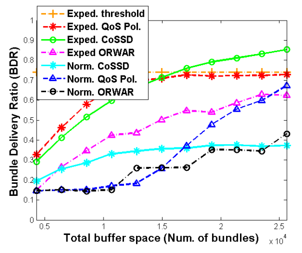


Fig. 23: QoS policy vs ORWAR and CoSSD (Cabspotting)

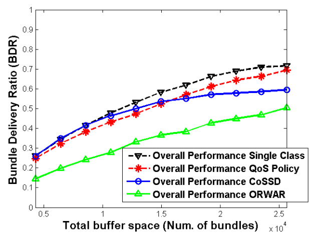


Fig. 24: Overall policies BDR comparison (Cabspotting)

4.2.3 Comparison with other policies

In the following, we compare the performance of our policy with other policies. In Fig. 23, the per class performance of our policy is compared to ORWAR and CoSSD, with respect to BDR optimization. The configuration of the two other policies has been done in the manner described in section 4.1.2. It is obvious that our policy outperforms ORWAR for both classes. This of course has an impact on the overall performance (Fig. 24), where our scheme achieves up to 20% higher results than ORWAR. Regarding the comparison with CoSSD, similarly to the synthetic simulation results (section 4.1.2), it is clear that CoSSD fails to capture the intended per class (Fig. 23) and overall performance (Fig. 24). Our policy steadily outperforms it in terms of Expedited class BDR, inside the infeasible domain (i.e., 4250-12800 buffer spaces), at the cost of the Normal class's performance; inside the feasible domain (i.e., $\geq$ 17000 buffer spaces) the picture changes, with our scheme's Normal class BDR exceeding the respective one with CoSSD, while the Expedited class remains stable around the desired threshold. Notice that, for both domains, the performance difference in terms of Normal class BDR between the two policies is much larger than the respective difference in terms of the Expedited class. This can be explained considering that, given the same amount of additional resources (bundle copies), the performance gain for bundles which are given a low number of initial copies on average (i.e., Normal class) can be much higher than the respective gain for bundles with an already high number of copies (i.e., Expedited class).

In Fig. 24, the overall BDR performance of our scheme is also compared to the optimal QoS unconstrained scheme of Krifa et al. [15], which considers a single priority class. It is evident that the unconstrained scheme achieves higher performance than our policy. This makes sense since the primal aim of our scheme is to satisfy the constraints of the higher classes. Thus, when the resources are limited, this has an impact on the overall network performance degradation comparing to the unconstrained case. However, when the resources are enough to allow the normal class to start converging to the performance of the expedited class (i.e., $\geq$ 15000 spaces), the overall performance of our scheme also starts to converge to the one of the single class scheme. This is another indicator of the optimal intended behavior

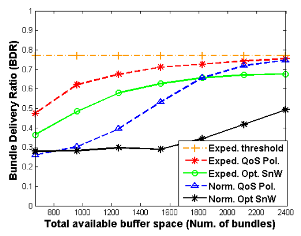


Fig. 25: QoS Policy vs Opt. SnW (BDR, Infocom)

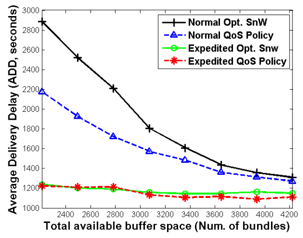


Fig. 26: QoS Policy vs Opt. SnW (Normalized ADD, Infocom)

of our policy.

In Fig. 25 - 26, the comparison of our policy with a more "optimized" version of the ORWAR protocol is depicted. Similarly to ORWAR, we use SnW and assign different replication factors per QoS class, proportional to their importance. However, this assignment is not done based on a static rule, like the one we used for ORWAR. Instead, it is done based on the logic specified in section 4.1.2. Thus, resources permitting, n_{exp} copies are always given to Expedited class bundles, to ensure its QoS requirement. If there exist remaining resources, $0 < n_{rem} < n_{exp}$, after this assignment, they consist the replication factor for Normal class bundles. Finally, if $n_{rem} > n_{exp}$, both classes are given equal initial number of copies, towards the target of overall performance maximization. For the BDR optimization problem, similarly to ORWAR, the higher class bundles are given absolute priority over the lower class ones. For the ADD optimization problem, though, to prevent starvation of the normal class, expedited class bundles are given absolute priority only while it holds that $T_i < D_{QoS}^{(k)}$. For higher T_i they get the same priority as normal class. Finally, for both problems, bundles of the same class are prioritized in descending order of their remaining TTL [9].

From Fig. 25, it is evident that the aforementioned approach performs worse than our basic policy. Contrary to our scheme, it cannot dynamically adjust the number of copies each bundle is getting in order to cope with possible initial delivery probability mis-predictions. In practice, a lower number of copies might be required on average to reach the desired threshold, than the one indicated by our prediction[10]. However, the subset of bundles which do not get delivered contribute to the worse performance comparing to our basic policy. As a result, the BDR of the Expedited class is constantly below the required threshold and the respective BDR of our basic policy (5% - 15%). Regarding the performance of the Normal class, although it is below our basic policy as well, its "starvation" is prevented for increasing availability of resources (i.e., > 1600 total spaces), in a more efficient manner than with the previous configuration of ORWAR (Fig. 23). This performance is indicative of a more desired behavior with respect to our optimization problem.

9. As it is shown in [15] this simple "drop oldest bundle" (or "schedule youngest") can be a good approximation of the optimal unconstrained utilities, when the congestion regimes in the network are low.

10. To determine the required number of copies, the delivery predictions based on the lower bound approximation were used (most conservative approximation).

We turn our attention to the ADD optimization problem now. Due to the large difference which was observed in terms of BDR per QoS class (up to 23% higher Normal class BDR with the basic policy), a direct comparison in terms of delivered bundles ADD wouldn't be fair. Thus, we compared the two schemes with respect to a normalized ADD metric, which considers the non-delivered bundles as well. Particularly, for a BDR value x and an ADD value y (of delivered bundles), the normalized ADD is computed as: $x \cdot y + (1 - x) \cdot TTL$. Although the performance in terms of the Expedited class is practically the same, it is obvious that the basic QoS policy outperforms the other scheme in terms of the Normal class and, thus, the overall network performance.

Overall, we claim that the optimized SnW scheme can consist a decent alternative to our QoS policy, considering its reduced overhead comparing to the latter. Indeed this "myopic" approach doesn't require any statistics collection for the prediction of n_i and m_i, as the QoS policy does. Instead, the resources distribution is purely based on the initial predictions, with respect to the required number of copies per class. These predictions are dependent on the knowledge of some network conditions information (i.e., mean pairwise meeting rate, meeting rates variance and density coefficient), which consist the only implementation complexity of this scheme. Given this information, the optimized SnW policy can better satisfy the intended overall behavior comparing to ORWAR (static replication factors per class without considering network conditions) and CoSSD (heuristic approach without any specific performance target).

5 Conclusions

In this work, we proposed a dynamic distributed prioritization scheme with the aim of preventing starvation of lower priority applications, while ensuring that the standards of higher priority applications are met. As discussed in sections 2.2-2.3, our policy manages to maximize the overall delivery ratio (thus, the average delivery rate), or minimize the average delivery delay metrics, among feasible solutions (i.e., solutions where the constraints of each QoS class are met). Moreover, we suggested efficient extensions of our policy, in order to be able to guarantee the intended performance in real life mobility conditions, characterized by heterogeneous and sparse contacts. Finally, we suggested an alternative scheme based on SnW. Although performing worse than our basic scheme, it can be appropriate for cases where we need to balance the trade-off between ease of use and high performance guarantees. We verified the optimality of our approach through simulations based on both synthetic and real mobility traces and we compared it with other QoS proritization approaches, to validate its superiority.

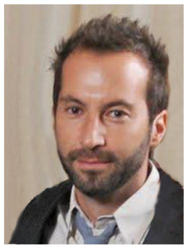
References

[1] G. Sandulescu and S. Nadjm-Tehrani, "Opportunistic DTN routing with window-aware adaptive replication," in *Proc. ACM AINTEC*, 2008.

[2] K. Shin, K. Kim, and S. Kim, "Traffic management strategy for delay-tolerant networks," *Elsevier Journal on Network and Computer Applications*, 2012.

[3] "Delay tolerant networking research group." http://www.dtnrg.org. 2015.
[4] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," tech. rep., Duke University, 2000.
[5] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *Proc. ACM SIGCOMM WDTN*, 2005.
[6] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," in *Proc. ACM ASPLOS*, 2002.
[7] N.Benamar, K.Singh, M.Benamar, D.Ouadghiri, and J.Bonnin, "Routing protocols in vehicular delay tolerant networks: A comprehensive survey," *Elsevier journal on Computer Communications*, 2014.
[8] Z. Lu and J. Fan, "Delay/disruption tolerant network and its application in military communications," in *Proc. IEEE ICCDA*, 2010.
[9] Y.-K. Ip, W.-C. Lau, and O.-C. Yue, "Forwarding and replication strategies for DTN with resource constraints," in *Proc. IEEE VTC*, April 2007.
[10] A. Balasubramanian, B. Levine, and A. Venkataramani, "DTN routing as a resource allocation problem," in *Proc. ACM SIGCOMM*, 2007.
[11] J. Burgess, B. Gallagher, D. Jensen, and B. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *Proc. IEEE INFOCOM*, 2006.
[12] K. Scott and S. Burleigh, "Bundle protocol specification," *IRTF RFC 5050*, 2008.
[13] S. Burleigh, "Bundle protocol extended class of service (ecos)," *draft-irtf-dtnrg-ecos-05*, 2013.
[14] V. Soares, F. Farahmand, and J. Rodrigues, "Traffic differentiation support in vehicular delay-tolerant networks," *Springer US Journal on Telecommunication Systems*, 2011.
[15] A. Krifa, C. Barakat, and T. Spyropoulos, "Message drop and scheduling in DTNs: Theory and practice," *IEEE Transactions on Mobile Computing*, 2012.
[16] P. Matzakos, T. Spyropoulos, and C. Bonnet, "Buffer management policies for DTN applications with different QoS requirements," in *Proc. IEEE GLOBECOM*, 2015.
[17] V. Conan, J. Leguay, and T. Friedman, "Characterizing pairwise inter-contact patterns in delay tolerant networks," in *Proc. ACM Autonomics*, 2007.
[18] W. j. Hsu and A. Helmy, "On nodal encounter patterns in wireless lan traces," *IEEE Transactions on Mobile Computing*, 2010.
[19] A. Passarella and M. Conti, "Analysis of individual pair and aggregate intercontact times in heterogeneous opportunistic networks," *IEEE Transactions on Mobile Computing*, 2013.
[20] A. Picu and T. Spyropoulos, "DTN-meteo: Forecasting the performance of DTN protocols under heterogeneous mobility," *IEEE/ACM Transactions on Networking*, 2014.
[21] T. Karagiannis, J. L. Boudec, and M. Vojnović, "Power law and exponential decay of inter contact times between mobile devices," in *Proc. ACM MobiCom*, 2007.
[22] H. Cai and D. Y. Eun, "Crossing over the bounded domain: From exponential to power-law inter-meeting time in manet," in *Proc. ACM MobiCom*, 2007.
[23] V. Kerf *et al.*, "Delay-tolerant architecture," *IETF RFC 4838*, 2007.
[24] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
[25] E. Chong and S. Zak, *An Introduction to Optimization*. John Wiley & Sons, 2013.
[26] A. Guerrieri, I. Carreras, F. D. Pellegrini, A. Montresor, and D. Miorandi, "Distributed estimation of global parameters in delay-tolerant networks," in *Proc. IEEE WoWMoM*, 2009.
[27] P. Sermpezis and T. Spyropoulos, "Delay analysis of epidemic schemes in sparse and dense heterogeneous contact networks," *IEEE Transactions on Mobile Computing*, 2016.
[28] G. W. Oehlert, "A note on the delta method," *The American Statistician*, 1992.
[29] E. Cooch and G. White, "Program mark: a gentle introduction," *download at http://www. phidot. org/software/mark/docs/book*, 2006.
[30] S. P. Dokov and D. P. Morton, "Higher-order upper bounds on the expectation of a convex function," *Stochastic Programming EPrint Series*, 2002.
[31] A. Madansky, "Bounds on the expectation of a convex function of a multivariate random variable," *Ann. Math. Statist.*, 1959.
[32] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD dataset epfl/mobility (v. 2009-02-24)." Downloaded from http://crawdad.org/epfl/mobility/20090224, 2009.
[33] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD dataset cambridge/haggle (v. 2006-09-15)." Downloaded from http://crawdad.org/cambridge/haggle/20060915/imote, 2006.

Panagiotis Matzakos received his Diploma in Electronic and Computer Engineering from the Technical University of Crete, Greece in 2010. He obtained his Master of Science in Mobile Communications and PhD in Electronics and Communication from Telecom ParisTech, France in January 2013 and October 2016, respectively. He is currently a post-doc researcher in the Department of Communication Systems at EURECOM, France. His research interests are in the areas of resource management for Disruption Tolerant Networks, LTE Device-to-Device communications and Network function virtualization for Cloud-RAN 5G architectures.



Thrasyvoulos Spyropoulos received the Diploma in Electrical and Computer Engineering from the National Technical University of Athens, Greece, and a Ph.D degree in Electrical Engineering from the University of Southern California. He was a post-doctoral researcher at INRIA and then, a senior researcher with the Swiss Federal Institute of Technology (ETH) Zurich. He is currently an Assistant Professor at EURECOM, Sophia- Antipolis.

Prof. Christian Bonnet joined EURECOM in 1992 after more than 12 years in industry. He was at the head of the Mobile Communications Department of EURECOM from 1998 to 2011. He is currently leading the Wireless Systems and Protocols research group. He participated in many (FP5, 6, 7, H2020) European projects dealing with mobility features based on IPv6 in heterogeneous mobile systems, Software defined Radio, Ad Hoc Networks and Mesh architectures for public safety systems. His current field of research is on M2M and Internet of Things. He is the president of the open source OpenAirInterface Software Alliance targeting 5G systems. He is involved in the Secured Communicating Solutions regional cluster for innovation as co-leader of the Mobiles services and M2M strategic axe.