Buffer Management Policies for DTN Applications with Different QoS Requirements

Panagiotis Matzakos, Thrasyvoulos Spyropoulos, and Christian Bonnet

Mobile Communications Department, EURECOM, 06410, Biot, France

{matzakos, spyropou, bonnet}@eurecom.fr

***Abstract*—Delay and Disruption Tolerant Networks (DTNs) have been proposed for challenging environments where the instability or lack of end-to-end paths is the rule rather than the exception. In this context, the principle of store-carry-and forward is used to sustain data sessions under intermittent connectivity, and data replication to increase the probability of on-time delivery. However, these techniques create the need for efficient scheduling and buffer management techniques, as the data load is generally larger than the amount of the available resources (i.e. bandwidth per communication opportunity, and buffer storage). A number of recent schemes have been proposed to make forwarding decisions that improve or even optimize the usage of these resources, in one way or another. Nevertheless, the majority of these schemes consider application sessions (and thus data messages) of equal importance. Furthermore, the few proposals that consider different traffic classes, do so in a somewhat "ad-hoc" manner, failing to provide real QoS guarantees. To this end, in this paper we formulate the problem of maximizing the network performance, in a limited resource network, subject to constraints corresponding to distinct QoS requirements (e.g., delivery probability) for each application class. Based on this formulation, we propose a distributed algorithm which: (i) guarantees that the individual QoS constraints are satisfied, when this is feasible given the amount of available resources, and (ii) allocates any remaining resources optimally, so as to maximize the desired performance metric. Simulation results, based on synthetic and realistic mobility scenarios, support our theoretical claims and further show that our policy outperforms other existing QoS prioritization schemes.**

Keywords*—*Delay Tolerant Networks, Scheduling policy, Buffer Management policy, QoS requirements.

I. 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 [1] 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. [2], [3]) 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 destination). However, the lack of constraints is rather unrealistic for most DTN settings (e.g. Energy/Buffer limited wireless sensor networks [4], vehicular networks with limited contact durations and/or buffer limited [5], military DTNs [6]). 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 scheduling and buffer management policies, in order to determine which data should be replicated during a meeting with another node and which data should be dropped during a buffer congestion event [7], [8], [9]. Such policies aim to optimize scheduling and dropping decisions, by keeping track of important message parameters (e.g., number of replicas, remaining 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. Although the application framework that we focus on is tolerant to delays, ensuring successful data delivery 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 requirements, and resource allocation decisions should take these into account.

Based on the bundle protocol [10], there is provision for three different QoS classes: Expedited (high priority), Normal (medium priority) and Bulk (low priority) by the DTN community [1]. More recently there has been an extension to support more priority levels within the Expedited class [11]. 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 [12], spray-and-wait is used for routing [3], 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. [13] propose different queue sizes, proportional to each QoS class's priority. Dropping decisions of each queue are then independent of the others. Hence, if a bundle of class i arrives and the queue for class i is full, a bundle from that queue will be dropped, even if other queues are not full. 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 while satisfying individual class QoS constraints* and (ii) to *derive a distributed algorithm for this problem that adapts to the available resources*. 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 [14]. In this aspect, the work closest to ours is [15], that also extends the optimal utilities derived in [14], 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. We support our analytical findings using extensive simulation on both synthetic and trace-based scenarios.

The rest of the document is organized as follows. In Section II we present our QoS policy. We start from the description of our system model (Section II-A), and then formulate the QoS prioritization problem as a constrained optimization problem (Section II-B). Finally, we propose a distributed resource allocation policy that is proven to be equivalent to a distributed gradient-ascent implementation of the solution (Section II-C). In section III, we evaluate the performance of our scheme by comparing it with other prioritization schemes. In section IV, we conclude the paper and discuss some future steps.

II. Buffer Management for Guaranteed QoS

In the following, we first describe our network, mobility and data traffic models, as well as the basic forwarding scheme used for data exchange during node meetings.

A. Model description

Mobility Model We assume there are N mobile nodes in the network. Each node encounters other nodes according to a random "contact" process. We assume that the pair-wise inter-contact times are independent, and exponentially distributed with rate λ. It has been shown that a number of mobility models and real traces correspond to contact models with approximately exponential tails [16], [17], [18].

Data traffic Model We consider each DTN node running C distinct applications concurrently. The traffic generated from each application is in the form of autonomous data units that we will call bundles from now on to be compliant with the DTN Architecture [19] and the associated bundle protocol [10]. We denote as $L_k(t)$ the number of distinct bundles of class k 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). Moreover, all applications generate traffic with the same rate (BGR).

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) metric (i.e. *Bundles received on time/Bundles sent*). From a bundle's perspective, this requirement can be expressed as its minimum accepted delivery probability QoS_k. We note that each bundle can belong to a single QoS class. Similarly, we could define our performance metric in terms of Delivery Delay (average, maximum, or per bundle).

Resource Constraints Our prioritization policy applies to DTN settings with limited buffer availability and scheduling opportunities, comparing to the network traffic load. This load originates from bundle generations at the DTN nodes and bundle replications during node meetings. In the evaluation of our work we consider only buffer constraints so far. However, it can easily be extended for scheduling constraints as well (i.e. maximum transferable data amounts per meeting).

Routing model and Buffer Management framework We consider epidemic routing. When two nodes encounter, they exchange their non-common packets and if this exchange leads to buffer congestion(s), the buffer management policy determines which bundles should be dropped. In [8] and [14],

distributed scheduling and buffer management schemes which consider the probability of on-time bundle delivery, $P_i^k(T_i)$, are suggested. The aim of our work was to extend such policies, by considering the distinct QoS requirements per bundle, as additional prioritization factors.

In table I the notation that we use in the rest of the paper is provided.

Notation	Description
N	Number of nodes in the network
$n_i^{(k)}(T_i)$	Number of copies of bundle i belonging to QoS class k at elapsed time T_i since its creation
$m_i^{(k)}(T_i)$	Number of nodes who have "seen" bundle i belonging to QoS class k at elapsed time T_i since its creation
$R_i^{(k)}$	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
QoS_k	Required probability of delivery for bundles of class k
$P_i^{(k)}(T_i)$	Probability of delivery for bundle i belonging to class k at time T_i since its creation.
λ	Meeting rate between two nodes.

TABLE I: Notation

B. QoS Prioritization Policy

As we have already highlighted, a good buffer management 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. Since we have chosen the Packet Delivery Ratio as our performance metric, we can formulate our prioritization problem as a constrained optimization problem in the following form:

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

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

$$Nb - \sum_{k=1}^{C} \sum_{i=1}^{L_k(t)} n_i^{(k)} \geq 0, \tag{3}$$

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

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 before its TTL expires 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 (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). Finally, 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).

Assuming $n_i^{(k)}$ could take real values, the above problem is a convex optimization problem. It can be solved analytically using the method of Lagrange multipliers and KKT conditions [20, 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 $\mathbf{n}$[1].

However, the above solution requires a centralized implementation of bundle copies and is not feasible, 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 [14]. There, the objective is differentiated to get the "marginal gain of an extra copy for each message" (referred to as "message utility") which is equal to:

$$U_i = \left(1 - \frac{m_i^{(k)}(T_i)}{N-1}\right) \cdot \lambda R_i^{(k)} exp(-\lambda n_i^{(k)}(T_i) R_i^{(k)}), \tag{5}$$

This utility function integrates $m_i^{(k)}$ term, which results from the fact that an enhanced objective function is used. Comparing to eq. (1), this objective function considers also the probability that one of the nodes who have "seen" the bundle is actually its destination. 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 [14].

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 the constrained optimization problems, gradient ascent algorithms can be modified to include appropriate penalty functions for each violated constraint [21, chapter 23.5]. Thus, an enhanced objective function would have the following form:

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

$$\phi(\mathbf{n}^*) = f(\boldsymbol{n^*}) - \sum_{k=1}^{C} \sum_{i=1}^{L_k(t)} \psi_k(QoS_k - P_i^{(k)}(T_i)), \qquad (6)$$

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:

$$U_i^{(k)} = U_i \cdot \left[1 + max\{0, c_k(QoS_k - P_i^{(k)}(T_i))\}\right]. \qquad (7)$$

In other words, the utility of a message is equal to its unconstrained utility U_i of Eq.(5), 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 differentiating the extended objective of Eq.(6) 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[2].

C. Buffer Management Policy

In the previous section, we have described a distributed QoS algorithm for the constrained optimization problem in hand, and have provided theoretical support for its convergence to the desired solution (i.e., optimal, conditionally on satisfying the requirements). Here, we show a simple implementation of this algorithm, and discuss some additional practical issues.

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 is below their QoS threshold; the second group consists of the bundles which are above their threshold. 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.(7) is monotonically decreasing from left to right in the queue of Fig. 1, and thus dropping messages from the right and scheduling from the left of this queue implements the desired policy.

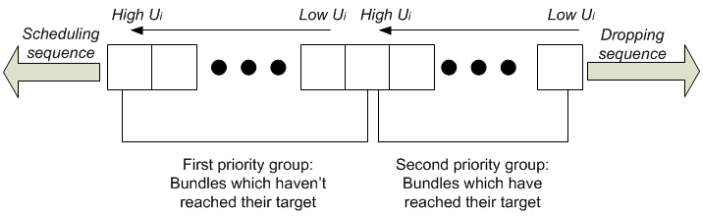

Fig. 1: Bundle scheduling and dropping sequence

The above policy works fine if the network parameters (e.g. packet generation rate, available storage in the network, inter-meeting rates) permit an algorithm to reach the desired delivery probabilities (or expected delivery ratios) 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.(7), for bundles of different QoS classes (k). Specifically, choosing $c_1 \gg c_2 \gg ... \gg c_K \gg 0$ (with 1 corresponding to the class with the highest nominal priority, and K the class with the lowest one) ensures a "smooth" fallback, if no feasible solution exists. 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 $j < K$, 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 resources[3]. It is important to stress here is 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 above 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. In this context, a subgroup attributed to a higher QoS class will always have higher priority than a subgroup of a lower QoS class.

In the following, the QoS prioritization policy is summarized in terms of bundle utilities and buffer classification mechanisms. $Group_i(T_i)$ refers to the priority group where bundle i is classified at elapsed time T_i, since its creation (Fig 1). $Subgroup_i(T_i)$ refers to the subgroup where i is classified

[2]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.

[3]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.

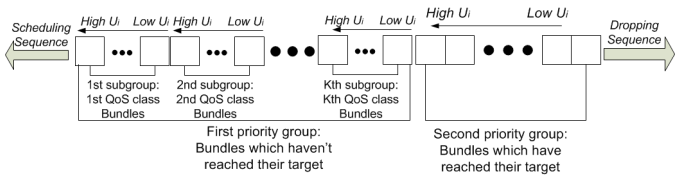

Fig. 2: An approach for bundle scheduling and dropping sequence for infeasible domains

based on each QoS class k, given that it belongs to the first priority group (Fig 2).

Summary of QoS Prioritization Policy:

- If $P_i^k(t) < QoS_k$
  - $U_i^{QoS_k} = [1+max\{0, c^k(QoS_k - P_i^k(t)\}] \cdot U_i$
  - $Group_i(T_i) = First$, $Subgroup_i(T_i) = k$
- If $P_i^k(t) > QoS_k$
  - $U_i^{QoS_k} = U_i$
  - $Group_i(T_i) = Second$

III. Performance Evaluation

A. Evaluation Setup

To evaluate our policy we considered three priority classes, namely Expedited (highest), Normal and Bulk (lowest) (based on the terminology of the bundle protocol specification [10] regarding different QoS classes). The BDR results are presented for various values of total available buffer space in the network. The aim is to test our policy as we vary the amount of buffer congestions. Then, we can see how it responds for both feasible and infeasible scenarios, in terms of satisfying the QoS constraints of different classes, as well as maximizing the overall network Performance. For our simulations we set up a basic environment where the nodes create bundles every $1/r$ seconds, they meet each other with a common rate based on the exponential distribution and they exchange their non-common bundle copies (Epidemic routing), as we described in section II-A. In terms of our mobility model, we also tested the performance with realistic traces. We compare the performance of our policy with two other prioritization policies that we described in section I, namely: ORWAR [12] and CoSSD [15].

In the following, we present the results of our policy compared to the other two policies, as well as with the approximate performance that we expect based on the network parameters that we use. We first show our results when using synthetic traces based on the exponential distribution and then when using traces from the SLAW mobility model [22]. SLAW is a state-of-the art mobility model which can produce realistic synthetic traces of the human walk. This makes it an appropriate tool, as part of a framework, to evaluate the performance of DTN protocols.

In table II the scenario configuration parameters are summarized for the exponential synthetic traces.

Number of Nodes (N)	50
Total simulation time	550 min
Mean nodes inter-meeting rate (λ)	$10^{-2} min^{-1}$
Message TTL	30 min.
Total Packet generation rate per node (r)	0.08 min^{-1}
Packet generation rate per node per priority class (r/3)	0.027 min^{-1}
Node Buffer space (B)	4 - 15 bundles per Node
Expedited desired QoS	0.75
Normal desired QoS	0.55
Bulk desired QoS	0.45

TABLE II: Simulation Parameters

B. Comparison with Other policies

Based on our previous descriptions, the desired 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 the 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 compare our policy with the other two, as we vary the amount of total buffer space in the network. To evaluate the performance of the three policies we can compare them to the approximate desired performance (Table III), when the buffer resources are distributed based on this logic. Obtaining such an approximate performance is straightforward if we consider the number of required copies per bundle, in order to reach a desired delivery probability and then map this to the total amount of required buffer space per QoS class. Thus, it is shown that for buffer values which correspond to the infeasible domain (i.e. 200 - 350 total spaces), the performance of the Expedited class for our policy is stable around its threshold. The other two classes gradually improve their performance, as we move towards the feasible domain, with the Normal class achieving steadily higher performance and reaching its performance target first.

Inside the feasible domain (i.e. 400 - 750 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 - 550 of the figure, where the Normal and the Bulk class gradually reach 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 (500 - 550 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 [12] does not specify a precise way for selecting the replication factors per class. For our purpose, we considered that a consistent selection is to pick as replication factors the number of copies required to reach the desired performance per class. Since, for our simulations, all bundles are of the same length, the dropping policy is based purely on each bundle's QoS class, by always favoring the higher class bundles

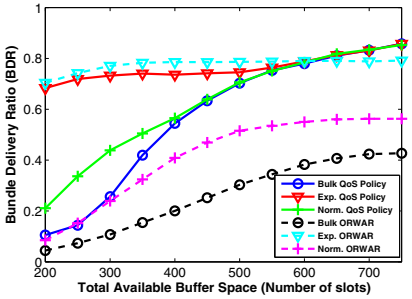

Fig. 3: QoS Policy vs ORWAR

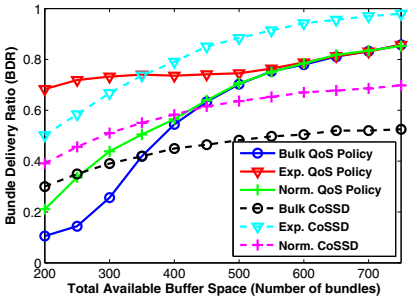

Fig. 4: QoS Policy vs CoSSD

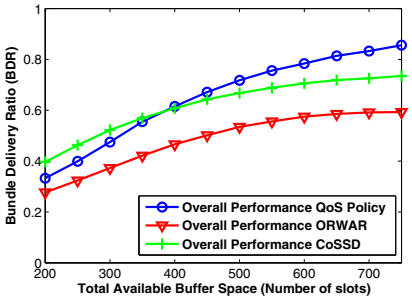

Fig. 5: Overall policies comparison

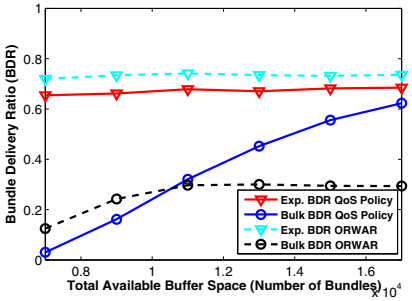

Fig. 6: QoS Policy vs ORWAR with Slaw trace

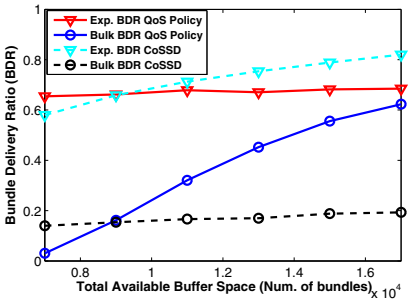

Fig. 7: QoS Policy vs CoSSD with Slaw trace

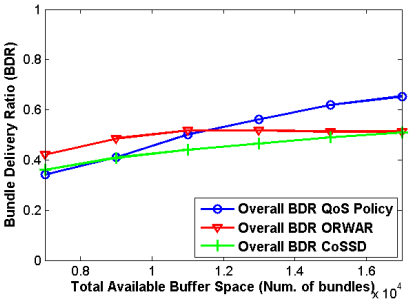

Fig. 8: Overall comparison with Slaw trace

Buffer spaces	BDR Expedited	BDR Normal	BDR Bulk
200	0.75	0	0
300	0.75	0.43 - 0.45	0
400	**0.75**	**0.55 - 0.59**	**0.43 - 0.45**
500	0.75	0.65 - 0.70	0.55 - 0.59
600	0.75	0.75	0.75
700	0.78 - 0.83	0.78 - 0.83	0.78 - 0.83

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

over the lower class ones. We notice that the performance of our policy approaches the expected one (table III) much more than ORWAR does. Particularly, the performance of the lower classes is increased much faster for our policy than for ORWAR. As a result, ORWAR needs much more resources in order to reach the desired threshold per QoS class. Moreover, it fails to exploit the additional resources and further increase the performance after that, as our policy does. This, of course, has an impact on the overall network results where our policy outperforms ORWAR by up to 30 %, as we increase the buffer resources (Fig. 5).

As described in section I, the derived utility function in CoSSD is based on a heuristic approach to extend [14] for the support of multiple QoS classes. Particularly, it has the following form:

$$(K - k_i) + \alpha \cdot \left(1 - \frac{m_i(T_i)}{N-1}\right) \cdot \lambda R_i exp(-\lambda n_i(T_i) R_i), \quad (8)$$

where k_i is the QoS class of bundle i (lower values for higher classes), K 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 (based on table III) 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 (< 350). 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. Once again, this is opposite to the already described optimal behavior. The consequence is that our policy outperforms CoSSD both in terms of lower classes, as well as overall network performance.

In figures 6 - 8 the results of our policy and the comparison with the other policies, when tested with traces from the Slaw mobility model, are shown. In this scenario, we have 200 nodes communicating and we consider two priority classes: Expedited (required $BDR = 0.70$) and Bulk (required $BDR = 0.20$). Based on our theoretical model (section II-A), in order to evaluate the performance of our policy with the mobility traces, we need to extract a common meeting rate λ which characterizes each distinct pair of nodes. However, in reality (mobility trace) there is a large variance between the meeting rates of different pairs. As a result, a large proportion of node pairs has much smaller meeting rates than the mean meeting rate value ($\tilde{\lambda}$) of the trace. If we don't take this into account, our policy might overestimate the delivery probability

at each decision step and often consider wrongly that the desired QoS threshold has been reached, leading finally to poorer BDR performance. In order to avoid this we consider: $\lambda = \tilde{\lambda} - \sigma$, where σ is the standard deviation of the meeting rate. By making such a "conservative" selection for λ, we can ensure that almost every Bundle holder has at least this meeting rate with every other node. Thus, in order to satisfy each QoS constraint, our policy will consider that it needs more copies per bundle. Consequently, when it comes to BDR performance we manage to achieve results which are at least as high as theory predicts for the selected value of λ and the estimated required number of copies. The tradeoff of this choice is that some times the Expedited class might be favored more than it should, in order to make sure that its QoS constraint is satisfied.

The results of this trade-off can be viewed in figure 6, where it is shown that, for very low buffer availability comparing to the data load (infeasible domain), our policy performs slightly worse than ORWAR. However, as the amount of buffer space is increased (feasible domain), our policy approaches the optimal performance, by favoring more and more the Bulk class. As a result, it outperforms ORWAR, which fails to exploit the additional resources efficiently. When compared to CoSSD (Fig. 7), our policy achieves constantly higher overall performance. For the first buffer values, CoSSD is further away from the desired threshold of the Expedited class comparing to our policy. As the buffer capacity is increased, the performance of the Expedited class is constantly increasing while the Bulk class remains practically stable. This clearly has an impact on the lower overall performance (Fig. 8) comparing to our policy.

IV. Conclusions and Future Work

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 we discussed in section II-B, our policy manages to maximize the overall delivery ratio metric among feasible solutions (i.e. solutions where the constraints of each QoS class are met). We verified the optimality of our approach through simulations and we compared it with other QoS proritization approaches, showing its superiority.

The performance metric that we used throughout this work is the delivery ratio. We claim that this is more meaningful than delivery delay for DTNs, where by definition it is quite hard to satisfy strict delay constraints. However our policy can easily be extended for the delivery delay metric as well, based on the same methodology. Moreover, throughout this paper we consider that the number of bundle copies is known at each local node who needs to take a scheduling or dropping decision. However, this is not realistic in real DTN scenarios. In this context, we are planning to evaluate the performance of our approach when using some form of packet copies estimation (e.g. HBD in [14]). Finally, it would be interesting to see how our policy can fit for multicast delivery scenarios as well.

V. Acknowledgments

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.

References

[1] "Delay tolerant networking research group." http://www.dtnrg.org. 2015.

[2] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," tech. rep., Duke University, 2000.

[3] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *Proc. of ACM SIGCOMM WDTN '05*, pp. 252–259, 2005.

[4] 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. of ASPLOS '02*, pp. 96–107, ACM, 2002.

[5] N. Benamar, K. D. Singh, M. Benamar, D. E. Ouadghiri, and J.-M. Bonnin, "Routing protocols in vehicular delay tolerant networks: A comprehensive survey," *Computer Communications*, vol. 48, no. 0, pp. 141 – 158, 2014. Opportunistic networks.

[6] Z. Lu and J. Fan, "Delay/disruption tolerant network and its application in military communications," in *Proc. of IEEE ICCDA '10*, vol. 5, pp. V5–231–V5–234, June 2010.

[7] Y.-K. Ip, W.-C. Lau, and O.-C. Yue, "Forwarding and replication strategies for DTN with resource constraints," in *Proc. IEEE VTC '07-Spring.*, pp. 1260–1264, April 2007.

[8] A. Balasubramanian, B. Levine, and A. Venkataramani, "Dtn routing as a resource allocation problem," in *Proc. of SIGCOMM '07*, pp. 373–384, ACM, 2007.

[9] J. Burgess, B. Gallagher, D. Jensen, and B. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *Proc. of IEEE INFOCOM '06*, pp. 1–11, April 2006.

[10] K. Scott and S. Burleigh, "Bundle protocol specification," *IRTF RFC 5050*, August 2008.

[11] S. Burleigh, "Bundle protocol extended class of service (ecos)," *draft-irtf-dtnrg-ecos-05*, July 2013.

[12] G. Sandulescu and S. Nadjm-Tehrani, "Opportunistic dtn routing with window-aware adaptive replication," in *Proc. of AINTEC '08*, pp. 103–112, ACM, 2008.

[13] V. Soares, F. Farahmand, and J. Rodrigues, "Traffic differentiation support in vehicular delay-tolerant networks," *Telecommunication Systems*, vol. 48, no. 1-2, pp. 151–162, 2011.

[14] A. Krifa, C. Barakat, and T. Spyropoulos, "Message drop and scheduling in dtns: Theory and practice," *IEEE Transactions on Mobile Computing*, vol. 11, pp. 1470–1483, Sept 2012.

[15] K. Shin, K. Kim, and S. Kim, "Traffic management strategy for delay-tolerant networks," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 1762 – 1770, 2012.

[16] A. Picu and T. Spyropoulos, "Dtn-meteo: Forecasting the performance of dtn protocols under heterogeneous mobility," *IEEE/ACM Transactions on Networking*, vol. PP, no. 99, pp. 1–1, 2014.

[17] T. Karagiannis, J.-Y. Le Boudec, and M. Vojnović, "Power law and exponential decay of inter contact times between mobile devices," in *Proc. of ACM MobiCom '07*, pp. 183–194, 2007.

[18] H. Cai and D. Y. Eun, "Crossing over the bounded domain: From exponential to power-law inter-meeting time in manet," in *Proc. of ACM MobiCom '07*, pp. 159–170, 2007.

[19] V. Kerf *et al.*, "Delay-tolerant architecture," *IETF RFC 4838*, April 2007.

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[21] C. Edwin K. P. and Z. Stanislaw H., *An Introduction to Optimization*. New Jersey, USA: John Wiley & Sons, 2013.

[22] K. Lee, S. Hong, S. J. Kim, I. Rhee, and S. Chong, "Slaw: A new mobility model for human walks," in *Proc. of IEEE INFOCOM '09*, pp. 855–863, April 2009.