Cluster-based Replication: a Forwarding Strategy for Mobile Opportunistic Networks

Evangelos Papapetrou and Aristidis Likas
Department of Computer Science and Engineering, University of Ioannina, Greece, e-mail: epap,arly@cse.uoi.gr

***Abstract*—Mutli-copy routing is an efficient and wide-spread approach for coping with the intermittent connectivity of mobile opportunistic networks. One popular multi-copy approach is known as "dynamic" replication; each node creates replicas on a contact basis using a utility that determines a node's fitness for delivering a message to its destination. This scheme is highly flexible, configurable with different utility functions and able to operate efficiently in networks with diverse characteristics. Nonetheless, its drawback is the tendency to produce a high number of replicas that consume limited resources such as energy and storage. Our approach to tackle this problem relies on the observation that, based on their utility values, the network nodes can be grouped into clusters that portray different delivery capabilities. Then, to avoid unnecessary replication, we exploit this finding to replicate a packet to nodes that belong to clusters with increasing delivery capability instead of replicating it to nodes with increasing utility. The new method works in synergy with the basic dynamic replication algorithms and is fully configurable, in the sense that it can be used with virtually any utility function. By conducting experiments in a diverse set of real-life networks, we empirically show that the method effectively reduces the overall number of replicas without hindering delivery efficiency.**

I. Introduction

The key challenge that a routing mechanism faces in the context of mobile opportunistic networks, especially in those with nodes exhibiting human mobility, is the intermittent and random connectivity experienced by nodes. To cope with this, several protocols endorse packet replication [1], [2], [3], [4], [5], [6], [7], [8]. The idea is simple; more replicas increase the probability that a packet carrier will encounter the destination and thus deliver the packet. Yet, replication comes at the cost of more transmissions and increased storage requirements. Therefore, it is imperative for a node to make "smart" replication decisions to improve the trade-off between delivery efficiency and cost (both energy and storage related), i.e., reduce replication without sacrificing delivery efficiency.

So far, the proposed multi-copy routing algorithms work towards this direction but follow two different replication approaches; the "constrained" (or "spray-based") [4], [3], [6], [7] and the "dynamic" one [1], [2], [9]. In the first approach, the source node determines the maximum number of replicas (L) to be sprayed into the network. In "dynamic" replication, the number of replicas is not predefined. Instead, each packet carrier dynamically creates replicas on a contact basis, i.e., according to the network connectivity. This aspect provides algorithms with the capacity to accommodate networks with diverse characteristics in contrast to Spray-based schemes where the optimal L depends on the network. In this work, we focus on dynamic replication due to its flexibility. Unfortunately, algorithms in this category tend to over-replicate, i.e., create an unnecessary amount of replicas [1]. The problem is more severe in those dynamic schemes that endorse a simple "Compare & Replicate" (CnR) approach [8], [1], [9]. There, a node v replicates a packet to an encountered node u if it has a higher *utility* value. The latter captures the fitness (or quality) of a node for delivering a message. Several methods try to improve this strategy by implementing more elaborate replication criteria. Probably the most efficient of those are Delegation Forwarding (DF) [1] and COORD [2].

The motivation of this work is the belief that, despite the success of the above-mentioned efforts to reduce replication, there is still room for significant improvement. We aim at improving the delivery efficiency-cost trade-off, i.e., produce less replicas without significantly impacting delivery efficiency. It is well known that opportunistic networks with human mobility exhibit certain social characteristics [10], [3], [11]. As a result, there exist nodes with diverse capabilities of delivering a message to its destination. Our intuition is that an analysis of the observed utilities will bring to light *clusters of utility values that correspond to groups of nodes with different delivery capabilities*. We also anticipate that such clusters could be identified regardless of the method used for constructing the utility, provided that the latter effectively captures a node's ability to deliver a message. After experimentally confirming our intuition in real-life networks, we follow the strategy to *replicate a packet to nodes belonging to clusters of increasing delivery capability*. This is in contrast to the current approach which is common in all schemes, i.e., to replicate a packet to nodes with progressively increasing utility. Our *Cluster-based Replication (CbR)* strategy can be used on top of the most well-known dynamic replication schemes, such as CnR, DF and COORD, and regardless of the chosen utility function. The experimental evaluation of CbR in a diverse set of contact traces from real-life opportunistic networks demonstrates that it is capable of significantly reducing replication and the related costs with negligible impact on the delivery efficiency.

In the following, we first review the related literature in Section II. Then, in Section III, we discuss the key concepts of our approach and experimentally validate the existence of utility clusters in various real-life opportunistic networks. We delineate CbR in Section IV and present an evaluation of its performance in Section V. Finally, we summarize our findings and discuss future research directions in Section VI.

II. Background and Related Work

The routing protocols proposed for opportunistic networks with human mobility can be broadly categorized in *single-copy* and *multi-copy* ones. Multi-copy schemes are superior to single-copy ones in terms of delivery efficiency because the probability of finding the destination is higher when multiple nodes carry the message. The trade-off is energy depletion and memory starvation at nodes. Therefore, research efforts have focused on reducing replicas without sacrificing the delivery efficiency. One approach is to use a probabilistic scheme [5], i.e., allow a node to probabilistically create replicas. Besides the difficulty in setting up the suitable replication probability, this approach is also prone to poor delivery efficiency. In the deterministic side, there are two prominent approaches; *"Spray-based"* or *"Constrained replication"* and *"Dynamic replication"*. Both use the concept of *utility* $U_v(d)$, a value that captures the *fitness* of node v for delivering the message to its destination d. There are various utility functions constructed based on some feature of a node's connectivity profile such as the contact rate [4], the time elapsed between successive contacts [7], the probability of node meetings [9], as well as metrics based on the social characteristics of nodes [3], [12]. Note that typically a utility is *destination dependent*, however there are also *destination independent* ones. Those capture a node's ability to act as a forwarder regardless of the actual destination, i.e., $U_v(d) = U_v, \forall d$. In the *"Spray-based"* class of algorithms, the source determines the maximum number of replicas L. Then, each node v with replicas selects which of its contacts will receive some of them. The selection process is either blind [6], i.e., every node is eligible for receiving replicas, or examines the utility of each encountered node u [7]. The selected node u will receive either half of v's replicas [6], [7] or a fraction that depends on $U_v(d)$ and $U_u(d)$ [4], [3]. Spray-based schemes can control the trade-off between delivery efficiency and the degree of replication through L. Yet, there is an important downside; choosing the optimal L is not trivial since this depends on the network.

The *"Dynamic replication"* strategy is more flexible since there is no need to predetermine the number of replicas to be created. Instead, every packet carrier v follows a utility-based approach and dynamically creates a replica based on the utility of the encountered node u. More specifically, v implements a *"Compare & Replicate"* (CnR) approach [8], [1], [9], i.e. forwards a copy of packet p to u when:

$$U_u(d) > U_v(d) + U_{th} \tag{1}$$

where U_{th} is a protocol parameter used to secure that the new carrier will contribute a minimum utility improvement. There are also other, less popular, approaches that relax or enforce (1) by co-evaluating how many replicas have been created so far or whether $U_u(d)$ exceeds a fixed threshold [9]. A point of criticism for this approach is that it frequently favors over-replication [1]. And this is true regardless of the utility choice although the latter impacts performance. To tackle the problem, Delegation Forwarding (DF) [1] introduces a replication strategy that exploits the *history of a node's observations*. In the case of a contact between v, that carries a packet p destined to d, and u then p is replicated to u iff:

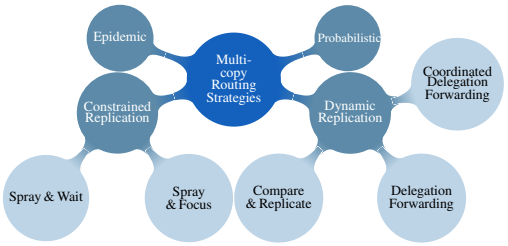


Fig. 1. Classification of multi-copy routing strategies for opportunistic networks with human mobility

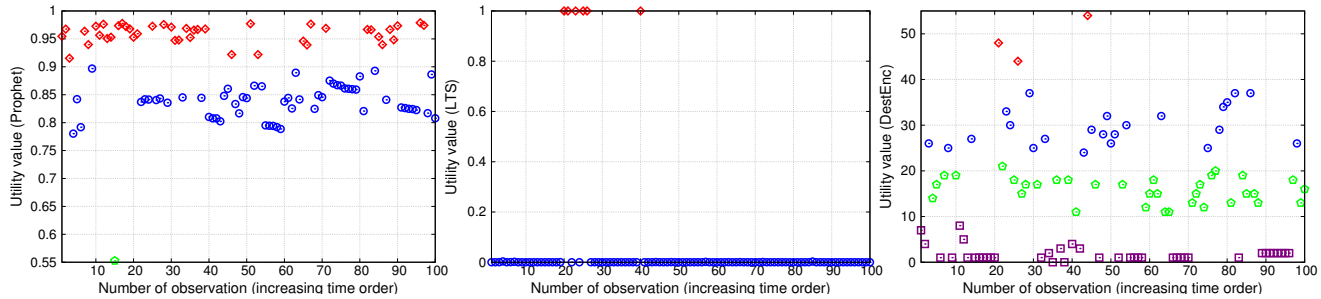
$$U_u(d) > \tau_v^p \ (= \max_{k \in N_v} \{U_k(d)\}) \tag{2}$$

where N_v is the set of all nodes that v has met since the reception of p and τ_v^p is the delegation threshold that v knowns for p, i.e., the highest utility recorded so far among the nodes that received p. The idea here is clear; there is no point in replicating a packet to u if another node with a higher utility already has the packet. COORD [2] builds on the idea of DF. It makes the observation that τ_v^p captures only v's perspective of the highest utility among the packet carriers. Therefore, enables carrier nodes to coordinate their thresholds which results to significant performance improvements.

III. Seeing the Forest not just the Trees

In this work, we focus on "dynamic" replication due to its flexibility in accommodating networks with diverse characteristics. In this class of algorithms, when a node v meets a node u with utility $U_u(d)$ the critical question that arises is:
Q1: Which values of $U_u(d)$ should result in the decision to replicate to u a packet p destined to d?
Clearly, all "dynamic" replication schemes answer this question by using a simple comparative approach which mandates that $U_u(d)$ should be greater than either $U_v(d)$ or τ_v^p, i.e., v's perception of the highest utility among packet carriers. Despite the improvements introduced by DF and COORD, this strategy is still prone to over-replication. The problem emerges when $U_u(d)$ is only *slightly higher* than the value that it is compared to. In this case, u will receive a copy despite the fact that its delivery capability is probably not significantly better. Therefore, in this case, the decision to create a replica is not profoundly the best one. This is a well-known issue and the reason for including U_{th} in (1). Obviously, in this case it is critical to define the optimal U_{th}, which is a daunting challenge since this depends on the utility function. More importantly, U_{th} also depends on the number and quality of v's contacts. For example, if all of v's contacts have similar utility values then perhaps it is better to consider a small U_{th} to avoid under-replication. However, this is not necessary if multiple contacts with diverse utility values exist.

In this work we take a different approach for answering (Q1). We make the observation that v could benefit by "seeing the big picture" and examine how $U_u(d)$ compares to the utility values of other nodes. To this end, our approach is to use the *distribution of utility values* formed by v's past contacts, i.e., decide on the importance of $U_u(d)$ based on the set of



(a) (b) (c)

Fig. 2. Example of utility values observed by node v $(= 23)$ for destination d $(= 50)$, Replication method: "Compare & Replicate", Trace: Reality [13], Utility function: (a) Prophet [9], (b) LTS [7], and (c) DestEnc [1].

values $\{U_k(d)\}_{k \in C_v}$, where C_v is the set of v's past contacts. Naturally, the question that now emerges is:

Q2: How to use such a distribution of utility values to identify important replication opportunities?

The answer highly depends on the characteristics of this distribution which in turn depend on the properties of the contacts between nodes. The analysis of contact traces from real networks with human mobility has clearly demonstrated that the nodes can be classified based on the contact properties into distinct groups [10], [11], each one corresponding to a different level of delivery capability. Recall that a utility metric is constructed based on one or more features of a node's contacts. Thus, it is reasonable to expect that, for any reasonably well-structured utility, *the grouping of nodes will show up as clusters of utility values*. If this is the case then our approach is to *decide* whether a contact u should receive a packet copy *based on the group that u belongs to*, i.e., *we do not decide based on $U_u(d)$ but rather we decide based on the characteristics of the cluster that $U_u(d)$ belongs to*.

To validate our approach, we conducted a set of experiments using CnR with different utility functions in various real-life contact traces. More specifically, for every node v we recorded the utilities reported by its contacts for a destination d, i.e., the set of values $\{U_k(d)\}_{k \in C_v}$. Then, we used the k-Means clustering algorithm [14] on this set of one-dimensional data in order to identify clusters of values, where the best number of clusters was automatically selected using the Silhouette criterion [15]. Fig. 2 illustrates a series of 100 values for the destination with id 50 recorded by the node with id 23 in the Reality trace [13]. The values are presented in the order they were recorded and different colors (and point types) correspond to different clusters. The three subfigures correspond to three different utility functions, namely Prophet [9], LTS [7] and DestEnc [4] (details about the simulation setup can be found in Section V). Clusters are evident in all figures. We recorded similar findings for utility values from various traces and observer-destination pairs. Since a utility value captures the node's fitness for delivering a packet, we *interpret such clusters of utility values as groups of nodes with different delivery capabilities*. Following this interpretation, *the key idea in our approach is to distribute replicas to nodes that belong to clusters with an increasing delivery capability*. However, efficiently implementing this strategy highly depends on the utility of the observing node. Therefore, it is imperative to polish the key idea to propose a sophisticated forwarding strategy. We discuss this strategy in detail in Section IV-C.

IV. Dynamic Replication with Clusters of Utility

We materialize our cluster-based replication strategy in the form of a method called *Cluster based Replication (CbR)*. It is important to note that CbR should not be seen as a standalone algorithm but rather as a mechanism that can be integrated into existing replication algorithms, such as CnR, DF and COORD. More specifically, we implement CbR on top of these strategies to transform their decision making processes so that, instead of just comparing two values, they take into account the clusters that those values belong to. In the following we will illustrate the synergy of CbR with the three strategies. This will result in three CbR flavors, namely *CbR-CnR*, *CbR-DF* and *CbR-COORD*. In the following we delineate CbR's mechanisms.

A. Identifying Clusters of Utility values

Each node first goes through a training period during which it collects a sufficient sample of the utility values reported by its contacts in order to detect clusters. In other words, a node v stores, for each destination d, a set of values $S_v^d = \{U_k(d)\}_{k \in C_v}$, where C_v is v's history of contact nodes with at least one packet to d. In the case of destination independent metrics, i.e., when the reported utility does not refer to a specific destination, v stores a single set of values $S_v = \{U_k\}_{k \in C_v}$. Recall that in all utility-based algorithms, including CnR, DF and COORD, the two nodes typically exchange their utility values during a contact. Therefore, the training process of CbR does not involve any additional communication cost. We define the training period in terms of the number of recorded values, i.e., the period ends when $|S_v^d| = N_{TR}$, where N_{TR} is a predefined number. Note that, in the most common case of a destination-dependent utility, the node actually goes through a different training period for every d. Observe that the distinct training periods may end at different times. This happens because, during a contact, the two nodes only exchange their utility values for every d for which at least a packet exists in their buffers. Furthermore, nodes usually report the utility values on a per packet rather than on a per destination basis, i.e., the utility $U_v(d)$ is reported for every packet destined to d. Therefore, in order to avoid importing noise to S_v^d, we record $U_u(d)$ only once per contact.

During this training period, the node implements the decision making process of the underlying algorithm, i.e., either CnR, DF or COORD. However, when the training period ends the node implements the *k-Means algorithm* [14] to cluster the recorded values. In fact, any clustering algorithm could be used as part of CbR. The choice of k-Means is based on the rather simple structure of the clusters observed in the recorded data. This allows us to choose a lightweight algorithm such as k-Means since the computational cost is still a point of consideration in a mobile environment. An important issue when using k-Means is how to estimate the number of clusters k. Recall that, when validating our motivation with data from real-life networks, we found out that every node observes a different number of clusters. Therefore, it is not feasible to find a k value that can be used globally. Instead, we follow a more flexible approach where each node determines k based on its own data. More specifically, each node executes k-Means on its own data for several values of k, i.e., $k \in [2, K_{max}]$. Then, the node chooses as the best k the value for which the corresponding clustering result provides the best score according to the *Silhouette criterion*. The latter is well-known for validating the quality of data clustering solutions [15]. Fig. 3 illustrates the pseudocode of the training process with a destination dependent utility. The same code is also used for destination independent utilities, i.e. when $U_u(d) = U_u, \forall\, d$ with the difference that the loop in line 1 is executed only once.

B. Refreshing a node's view

In opportunistic networks, especially in those with human mobility, it is reasonable to expect that a node's connectivity profile, i.e., the average rate and duration of its contacts, may evolve over time, e.g., because the node moves in various locations during different hours of a day. Since a utility function hinges on a node's contact properties, one would expect a similar evolution of the utility values observed by a node. Such changes take place during relatively long periods of time. Therefore, they can not be captured by the training period which should be of relatively small duration to timely enable cluster based replication decisions. Thus, there is a need for a process able to capture such changes and update the clustering result. Our experimental results indicate that, in most cases, the clusters of utility values do evolve over time. However, the changes frequently involve the structure and center of the clusters rather than their number. Based on this observation, we decided to employ a simple, yet efficient, low-complexity method for updating the clusters found during the training period. This is the Learning Vector Quantization (LVQ) clustering algorithm [16] which can be considered as an on-line version of the k-Means algorithm. More specifically, each time a node records a new utility value after the end of the trainig period, LVQ decides on which cluster this value belongs to and subsequently moves the center of this cluster towards the new value. This update process runs in parallel with the decision making one and does not interfere with it.

Require: $training(d)$, $S_v^d\ \forall\, d$, N_{TR}
1: **for** every reported $U_u(d)$ **do**
2: **if** $training(d) =$ true **then**
3: $S_v^d = S_v^d \cup U_u(d)$
4: **if** $|S_v^d| = N_{TP}$ **then**
5: $minscore \leftarrow \infty$
6: **for** $i = 2$ to $i = K_{max}$ **do**
7: $clust_i \leftarrow k\text{-}Means(S_v^d, i)$
8: **if** $Sil\_score(clust_i) < minscore$ **then**
9: $minscore \leftarrow Sil\_score(clust_i)$
10: $K_{opt} \leftarrow i$, $clust_{opt} \leftarrow clust_i$
11: **end if**
12: **end for**
13: $training(d) \leftarrow$ false
14: **end if**
15: **else**
16: $Update\_with\_LVQ(U_u(d))$
17: **end if**
18: **end for**

Fig. 3. The pseudocode of the training process (destination dependent utility case) executed when v encounters u.

C. Making Replication Decisions

After completing the training period, a node is able to use the identified clusters to make replication decisions. In a nutshell, the basic idea of CbR dictates that a node v replicates a packet to u provided that the utility of the latter belongs to a cluster of higher utility values. To implement this simple rule, a node should first rank the identified clusters. This can be easily accomplished since the clustered data are one-dimensional. Thus, we rank the clusters in decreasing order based on their center value. Accordingly, each node z is assigned the rank of the cluster on which its utility value belongs to. In the following, we denote the rank of z with R_z. Based on the ranking method, the previous forwarding rule now reads: "*u receives a packet replica if its utility belongs to a cluster of a better rank*". Note that this rule is rather stringent and in certain occasions may result in under-replication and thus poor delivery rates. We have identified two occasions where this may occur. The first is the case that the utility used for decision making by the carrier node v ($U_v(d)$ in CnR, τ_v^p in DF and COORD) belongs to the top level cluster of values. In this case, the rule actually prohibits any replication in the group of most capable nodes even if v is the source node. The second case of potential under-replication occurs when the utility used by v resides in a populous cluster of values and the clusters with a better rank are sparsely populated. In this case, the opportunities for replicating the packet to a better ranked cluster are sparse therefore the most probable scenario is that the packet replication will involve a substantial delay. The best strategy for both the aforementioned cases is to relax the requirement of replicating the packet to a better ranked cluster by *allowing replication to a node u with a utility in the same cluster provided that u's utility is higher than the utility used by v (traditional decision making).* According to this strategy, when CbR is implemented on top of CnR, it transform CnR's replication requirement found in (1) to:

$$R_u < R_v \text{ or } \big(R_u = R_v \text{ and } p.rep = false \text{ and } U_u(d) > U_v(d)\big) \quad (3)$$

where R_v and R_u are the ranks of the packet carrier v and the encountered node u, p the packet and d its destination.

We mitigate the risk of under-replication by moving from the basic criterion $R_u < R_v$ to a relaxed one, i.e., $R_u = R_v$, if v has not yet replicated p. We distinguish non-replicated packets from replicate ones using a single bit in the packet's header. As soon as p is forwarded, this bit is set to 1 and the relaxation is canceled. Note that, in contrary to the case that $R_u < R_v$, when $R_u = R_v$ it is possible that $U_u(d) < U_v(d)$. The CnR rule ($U_u(d) > U_v(d)$) eliminates replication in such cases. We follow a similar approach when integrating CbR into DF and COORD. Recall that in both DF and COORD the replication decision is made using (2), where τ_v^p is v's perception of the highest utility among the carriers of p. The two algorithms only differ in the way that τ_v^p is updated. Since the decision making criterion is common in DF and COORD, the implementation of the CbR rule is also common, i.e.,:

$$R_u < R_t \text{ or } \left(R_u = R_t \text{ and } R_t = R_v \text{ and } U_u(d) > \tau_v^p\right) \qquad (4)$$

where R_t is the rank of the cluster that τ_v^p belongs to. Here, we follow a more efficient relaxation approach. We allow replication when $R_u = R_t$ provided that $R_v = R_t$. The latter equality means that $U_v(d)$ and τ_v^p reside in the same cluster, i.e., the packet carrier v and the carrier with the highest utility have similar delivery capacity, thus the packet has not moved to a better cluster. When this happens, τ_v^p will be updated to a new value so that $R_t > R_v$, which will deactivate the relaxation. Again, when $R_u = R_t$ the traditional rule ($U_u(d) > \tau_v^p$) acts as a safeguard. As a final note, all CbR flavors are also compatible with destination independent utility functions.

V. Evaluation

We evaluate the performance of all CbR flavors under various opportunistic environments. To this end, we use the Adyton [17] simulator. For the evaluation we use traces that represent opportunistic networks of different scale. More specifically, we used two conference traces, namely Infocom'05 [18] and Sigcomm'09 [19], two traces from campuses, the MIT Reality [13] and the Milano pmtr datasets [11], and a city-level trace collected in Cambridge, UK, the Cambridge upmc [20]. CbR is able to work with virtually any proposed utility metric. Clearly, the utility choice impacts performance. Thus, we use a collection of five well-known utilities, both destination dependent (DD) and independent (DI) ones, to assess the performance of CbR. More specifically, we use: a) the DestEnc [1] utility that captures the total number of contacts with a specific node and therefore is DD, b) its DI version Enc [4] that captures the number of contacts with all nodes, c) the LTS [7] DD utility which depends on the time elapsed since the last contact with the destination, d) the Prophet [9] utility which is also DD and captures the fitness of a node to deliver a message both directly and indirectly, and e) the SPM [12] DD utility that depends on the frequency, the longevity and the regularity of past contacts with the destination.

Regarding the clustering settings, the analysis of data from real contact traces revealed that using a small value for K_{max} such as 4 is sufficient for capturing reasonable estimates of the number of clusters. Furthermore, we used a training period of 50 samples, i.e., N_{TR}=50. Finally, the LVQ learning rate was set to 0.05, i.e., the center moves towards the newly added value so that the distance is reduced by 5%.

In each experiment we use a traffic load of 1000 packets generated uniformly in the interval during which both the source and the destination are present in the network. To eliminate statistical bias and monitor the network in its steady state, we use a warm-up and a cool-down period during which packets are not generated. The duration of both periods is 20% of the total trace duration. We report results as average values of 20 repetitions. In each repetition we randomly select the source/destination pair and the generation time for each packet.

We test the performance of all three flavors of CbR in all traces, using each time a different utility metric. To eliminate other interfering factors, we assume an infinite buffer in each node. We use the *routing gain (RG)*, i.e., the percentage of transmissions saved when using CbR, to capture the extend at which CbR reduces the replicas and therefore the number of transmissions. More specifically, we monitor the quantity $(1 - T_{CbR}/T)\,\%$, where T is the number of transmissions per delivered packet for the underlying algorithm, i.e., either CnR, DF or COORD, and T_{CbR} is the same number for its CbR flavor. Fig 4 illustrates the routing gain provided by CbR when used on top of CnR (fig. 4(a)), DF (fig. 4(b)) and COORD (fig. 4(c)). In all cases there is a significant gain that, depending on the baseline algorithm and the utility metric, ranges from $\sim 5\%$ to an impressive $\sim 60\%$. Reasonably, the routing gain is smaller when CbR is integrated into DF and COORD since those algorithms already significantly reduce transmissions compared to CnR. Therefore there is a smaller room for improvement. Still, CbR achieves gains that reach up to $\sim 40\%-45\%$. What is of great importance is that CbR's routing gain comes at limited or virtually no delivery cost, i.e., CbR clearly improves the delivery efficiency-cost trade-off. The first value in Table I presents the *normalized delivery rate* for all CbR flavors, i.e., the delivery rate of a CbR flavor normalized to that of the underlying algorithm. We report results for the two traces that posed the most serious challenge to CbR in terms of delivery efficiency, namely Reality and Cambridge. The performance of all CbR flavors is in most cases within $\sim 1\%$ of the performance of the baseline algorithm and in all cases within $\sim 3\%$. Besides being minimal, this performance degradation can be justified if we bear in mind that even random contacts help nodes communicate. However, such random contacts are not predictable and the only way to exploit them is to increase replication. Furthermore, our results

TABLE I
NORMALIZED DELIVERY RATE AND DELAY OF CNB FLAVORS

	DestEnc	Enc	LTS	PRoPHET	SPM
MIT Reality trace					
CnB-CnR	0.998-1.016	0.997-1.076	0.996-1.085	0.998-1.052	0.997-1.026
CnB-DF	0.997-1.004	0.994-1.045	0.996-1.007	0.995-1.028	0.998-1.008
CnB-COORD	0.998-1.017	0.999-1.036	0.996-1.015	0.995-1.034	0.998-1.008
Cambridge trace					
CnB-CnR	0.997-1.000	0.967-0.983	0.990-1.003	0.979-0.996	0.982-0.993
CnB-DF	0.992-0.985	0.993-0.995	0.990-0.984	0.984-1.008	0.995-0.995
CnB-COORD	0.993-0.987	1.001-1.006	0.990-0.980	0.984-1.008	0.9950.995

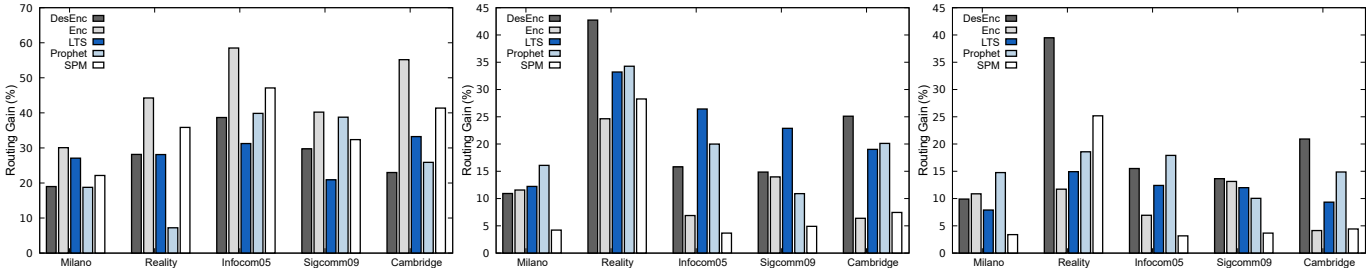


Fig. 4. Routing gain of the CbR approach in various traces and for various utility metrics: a) CbR-CnR, b) CbR-DF, and c) CbR-COORD

in a more realistic setting with limited buffer size (omitted for reasons of brevity) show that when the node's buffer size reduces this minimal degradation of delivery efficiency is almost eliminated and in many cases turns into an improvement.

The reduced level of replication in CbR, as expected, also interferes with the delivery delay. The second value in Table I presents the *normalized delay* of CbR flavors in the Reality and the Cambridge datasets. In the case of the Reality trace there is a limited increase of delay. An easy way to explain this observation is to visualize replication as a process that delivers multiple copies to a destination through different paths. Reducing replication is equivalent to pruning some paths. This delays the packet delivery unless none of the pruned paths is the shortest one, which is rather unlikely. To increase the probability that the shortest path will survive pruning, one should provide a high rank to the contacts comprising this path. However, this is the responsibility of the utility metric and not of CbR. Indeed, note that the delay increase is smaller when the utility takes into account delay-related connectivity aspects such as the frequency and the regularity of contacts (e.g. SPM). In the Cambridge trace, the impact of replication on the delay is limited. An apparent reason is that Cambridge is a more dense trace with a higher contact rate, thus denying replication to a contact results in a smaller delay increase. Note that in some cases the delay of CbR in fact reduces. This decrease is minimal and statistically insignificant. It is attributed to the statistical bias due to the lower delivery rate. This phenomenon is not evident in the Reality trace because contacts are less frequent, thus reduced replication results in more delay which cloaks that statistical bias.

VI. Conclusion

Dynamic replication schemes provide a flexible forwarding solution for mobile opportunistic networks. Their disadvantage is the tendency to over-replication. We tackle the problem by using a novel approach. We cluster the nodes based on their utility value. Then, instead of replicating a packet to a node with a better utility value, we replicate the packet that belongs to a cluster of better utility values. We first validated that utility values form clusters and that these clusters can be identified by a node using lightweight clustering algorithms. Then, we delineated a forwarding policy that can be used to transform the decision making process of traditional dynamic replication schemes to one that relies on cluster-based decisions. We experimentally demonstrated the significant benefits of cluster-based replication. As a future work, we plan to investigate alternative clustering approaches for updating a node's view about the clusters of utility values. Furthermore, we aim to adapt the proposed method so that it can be integrated into replication schemes that use two or more utility functions, e.g., a destination dependent and an independent one.

References

[1] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot, "Delegation Forwarding," in *Proc. ACM MobiHoc*, 2008, pp. 251–260.

[2] N. Papanikos and E. Papapetrou, "Coordinating Replication Decisions in Multi-copy Routing for Opportunistic Networks," in *Proc. IEEE Int. Conf. Wireless and Mobile Comput., Netw. and Commun. (WiMob)*, 2014, pp. 8–13.

[3] E. M. Daly and M. Haahr, "Social network analysis for information flow in disconnected delay-tolerant MANETs," *IEEE Trans. Mobile Comput.*, vol. 8, no. 5, pp. 606–621, 2009.

[4] S. C. Nelson, M. Bakht, and R. Kravets, "Encounter-based routing in DTNs," in *Proc. IEEE INFOCOM*, 2009, pp. 846–854.

[5] X. Chen, J. Shen, T. Groves, and J. Wu, "Probability delegation forwarding in delay tolerant networks," in *Proc. IEEE ICCCN*, 2009.

[6] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Efficient routing in intermittently connected mobile networks: the multiple-copy case," *IEEE/ACM Trans. Netw.*, vol. 16, no. 1, pp. 77–90, 2008.

[7] T. Spyropoulos, T. Turletti, and K. Obraczka, "Routing in delay-tolerant networks comprising heterogeneous node populations," *IEEE Trans. Mobile Comput.*, vol. 8, no. 8, pp. 1132–1147, 2009.

[8] W. Moreira, P. Mendes, and S. Sargento, "Opportunistic routing based on daily routines," in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2012.

[9] A. Lindgren, A. Doria, E. Davies, and S. Grasic, "Probabilistic Routing Protocol for Intermittently Connected Networks," RFC 6693, Aug. 2012.

[10] E. Yoneki, P. Hui, and J. Crowcroft, "Visualizing community detection in opportunistic networks," in *Proc. of ACM CHANTS*, 2007, pp. 93–96.

[11] S. Gaito, E. Pagani, and G. P. Rossi, "Strangers help friends to communicate in opportunistic networks," *Computer Networks*, vol. 55, no. 2, pp. 374 – 385, 2011.

[12] E. Bulut and B. K. Szymanski, "Exploiting friendship relations for efficient routing in mobile social networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 12, pp. 2254–2265, 2012.

[13] N. Eagle and A. S. Pentland, "CRAWDAD data set mit/reality (v. 2005-07-01)," Downloaded from http://crawdad.org/mit/reality/, Jul. 2005.

[14] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[15] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: an introduction to cluster analysis*. Hoboken, NJ, USA: Wiley, 1990.

[16] T. Kohonen, "Learning vector quantization," in *The Handbook of Brain Theory and Neural Networks*, 1st ed., M. Arbib, Ed. MIT Press, 1995.

[17] N. Papanikos, D.-G. Akestoridis, and E. Papapetrou, "CRAWDAD toolset tools/simulate/uoi/adyton (v. 2016-04-21)," Downloaded from http://crawdad.org/tools/simulate/uoi/adyton/20160421, Apr. 2016.

[18] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD data set cambridge/haggle (v. 2006-01-31)," Downloaded from http://crawdad.org/cambridge/haggle/, Jan. 2006.

[19] A.-K. Pietilainen, "CRAWDAD data set thlab/sigcomm2009 (v. 2012-07-15)," Downloaded from http://crawdad.org/thlab/sigcomm2009/.

[20] J. Leguay, A. Lindgren, and T. Friedman, "CRAWDAD data set upmc/content (v. 2006-11-17)," Downloaded from http://crawdad.org/upmc/content/, Nov. 2006.