Content Replication in Mobile Networks

Chi-Anh La, *Student Member, IEEE,* Pietro Michiardi, *Member, IEEE,* Claudio Casetti, *Member, IEEE,* Carla-Fabiana Chiasserini, *Senior Member, IEEE,* and Marco Fiore, *Member, IEEE*

***Abstract*—Performance and reliability of content access in mobile networks is conditioned by the number and location of content replicas deployed at the network nodes. In this work, we design a practical, distributed solution to content replication that is suitable for dynamic environments and achieves load balancing. Simulation results show that our mechanism, which uses *local measurements only*, approximates well an optimal solution while being robust against network and demand dynamics. Also, our scheme outperforms alternative approaches in terms of both content access delay and access congestion.**

***Index Terms*—Content replication, mobile networks, node cooperation, distributed algorithms.**

I. Introduction

ACADEMIC and industrial research in the networking field is pursuing the idea that networks should provide access to contents, rather than to hosts. Recently, this goal has been extended to wireless networks as well, as witnessed by the tremendous growth of services and applications offered to users equipped with advanced mobile terminals.

The inexorable consequence of a steady increase in data traffic exerted by mobile devices fetching content from the Internet is a drainage of network resources of mobile operators [1]. A promising approach to solve this problem is *content replication*, i.e., to create copies of information content at user devices so as to exploit device-to-device communication for content delivery. This approach has been shown to be effective especially in wireless networks with medium-high node density, where *access congestion* is the main limiting factor to the performance of content delivery (see, e.g., [2] for a survey on the topic).

In this paper, we consider a mobile network and explore the concept of content replication in a *cooperative* environment: nodes can fetch content from the Internet using a cellular network, store it, and possibly serve other users through device-to-device communication (e.g., IEEE 802.11) [3]. Our scenario accommodates the possibility for content to exhibit variegate popularity patterns, as well as to be updated upon expiration of a validity-time tag, so as to maintain consistency with copies stored by servers in the Internet.

The scenario we target introduces several problems related to content replication. Our endeavor is to build upon the theoretic works that have flourished in the Location Theory literature and address the *joint problem* of content replication and placement, with the goal of designing a lightweight, distributed mechanism. Our main contributions are as follows:

(i) we revisit traditional Location Theory and propose a distributed mechanism inspired by local search approximation algorithms (Sec. II). Our solution exploits a formulation of a multi-commodity capacitated facility location problem to compute a solution based on local measurements only (Sec. III);

(ii) through an extensive simulation study, we show that our scheme well approximates an optimal solution when both network and content dynamics are considered (Secs. IV and V). Our mechanism achieves load balancing across the network and scales well with the network size, making it suitable for scenarios in which access congestion may appear;

(iii) we compare our content replication scheme with existing mechanisms, and show under which conditions our approach yields better performance (Sec. V).

We review prior work in Sec. VI, and draw our conclusions in Sec. VII.

II. Network scenario and problem statement

We first detail the system model we refer to. Then, we inherit the problem of replication typical of the wired Internet and we discuss the new challenges introduced by the dynamic nature of wireless networks.

System model: We investigate a scenario including mobile users (i.e., nodes), equipped with devices offering 3G/4G Internet connectivity as well as device-to-device communication capabilities (e.g., IEEE 802.11). Although we do not concern ourselves with the provision of Internet access in ad hoc wireless networks, we remark that broadband connectivity allows new content to be fetched and, possibly, updated.

We denote the set of mobile nodes by $\mathcal{V}$, with $V = |\mathcal{V}|$, and we consider that they may be interested in a set of information items, $\mathcal{I}$ ($|\mathcal{I}| = I$). Each item $i \in \mathcal{I}$, of size $s(i)$, is tagged with a validity time and originally hosted on a server in the Internet, which can be accessed through the broadband access we hinted at. We define the content popularity level of the generic item i, $\pi(i)$, as the fraction of nodes interested in such an item. Thus, we have $0 \leq \pi(i) \leq 1$, with $\pi(i) = 1$ when all nodes in the system are interested in content i.

We focus on a *cooperative environment* where a node $j \in \mathcal{V}$ wishing to access the content first tries to retrieve it from other devices. If its search fails, the node downloads a fresh content replica from the Internet server and temporarily stores it for a period of time τ_j, termed *storage time*. For simplicity of presentation, we assume $\tau_j = \tau$, $\forall j \in \mathcal{V}$. During the storage period, j serves the content to other nodes upon receiving a request for it and, possibly, downloads from the Internet server a fresh copy of the content if its validity time has

C.-A. La and P. Michiardi are with EURECOM Institut, Sophia Antipolis, France e-mail: firstname.lastname@eurecom.fr.

C. Casetti and C.-F. Chiasserini are with Dipartimento di Elettronica, Politecnico di Torino, Torino, Italy e-mail: firstname.lastname@polito.it.

M. Fiore is with Université de Lyon, INRIA, INSA-Lyon, CITI Lab, France e-mail: marco.fiore@insa-lyon.fr.

Manuscript received XXX X, 2011; revised July 20, 2011.

expired. We refer to the nodes hosting an information copy at a given time instant as *replica nodes*. We denote the set of nodes storing a copy of item i at time t by $\mathcal{R}_i(t)$, and define $\mathcal{R}(t) = \cup_{i \in \mathcal{I}} \mathcal{R}_i(t)$, with $R = |\mathcal{R}|$. Also, we associate to each replica node j a capacity value c_j, which, as we shall see later, relates to the capability of the node to serve content requests.

A node, which is interested in a generic information item i and does not store any copy of it, issues queries for such an item at a rate λ. Replica nodes, receiving a query for an information item they currently store, will reply with a message including the requested content.

In the following we model the network topology at a given time instant t through a graph $G(t) = (\mathcal{V}, \mathcal{E}(t))$, whose set of vertices coincides with the set of nodes $\mathcal{V}$ and the set of edges $\mathcal{E}(t)$ represents the set of links existing between the network nodes at time t.

Problem statement: Both content replication and caching have received significant attention in the literature, however they differ since replication is an independent process aimed at creating copies of a content at the network nodes, regardless of whether they asked for it or not. Caching, instead, is a by-product of the content query mechanism as only nodes that retrieved the content have the possibility to cache it [2], [3].

Our claim, confirmed by simulation results presented in the paper, is that, in the above context,content replication is to be preferred to caching. Indeed, given that the storage capacity at the nodes can be considered as unlimited and the content request rate is known, replication can effectively address the scarcity of radio resources and the need for an even traffic load distribution. Caching instead may lead to the creation of a large number of copies in the network, especially for popular content. In medium-high dense networks, this may raise the problems of: (i) large overhead due to multiple replies to a single query, (ii) energy depletion of a large fraction of nodes acting as content providers, (iii) congestion in accessing the cellular network.

We therefore deal with content replication and design a mechanism to determine how many replicas should be created in the network and where, under dynamic, realistic conditions. Traditionally, a similar problem, although in a simpler scenario, has been studied through the lenses of Location Theory [4], by considering replicas to be created in the network as facilities to open. Then, as the first step to understand the problem under study, we restrict our attention to a simplified network setting and revisit a centralized approach for facility location problems. We assume static nodes and constant demand, hence we drop the time dependency from our notation. Furthermore, we drop the load balancing requirement we previously outlined, and assume that content queries are directed to the closest replica node. Finally, for simplicity, we let all users be interested in every content i ($i = 1, \ldots, I$).

Given such a simplified scenario, we formulate content replication as a *capacitated* facility location problem where the set of replica nodes $\mathcal{R} = \cup_i \mathcal{R}_i$ corresponds to the set of facilities that are required to be opened, nodes requesting a content are referred to as clients and items correspond to the commodities that are available at each facility. We model the capacity of a replica node as the number of clients that a facility can serve. The goal is to identify the *subset* of facilities that, at a given time instant, can serve the clients so as to minimize some global cost function while satisfying the facility capacity constraints. Note that, in our scenario, both clients and facilities lay on the same network graph $G = (\mathcal{V}, \mathcal{E})$. The problem can be defined as follows:

Definition 1: Given the set $\mathcal{V}$ of nodes with pair-wise distance function d and the cost f_j of opening a facility at $j \in \mathcal{V}$, select a subset of nodes as facilities, $\mathcal{R} \subseteq \mathcal{V}$, so as to minimize the joint cost $C(\mathcal{V}, f)$ of opening the facilities and serving the demand while ensuring that each facility j can only serve at most c_j clients. Let $C(\mathcal{V}, f)$ be:

$$C(\mathcal{V}, f) = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{R}_i} f_j(i) + \sum_{i \in \mathcal{I}} \sum_{h \in \mathcal{V}} d(h, m_h(i)) \qquad (1)$$

where $f_j(i)$ is the cost to open a facility for commodity i, $\mathcal{R}_i \subseteq \mathcal{V}$ is the subset of nodes acting as facilities for commodity i, $m_h(i) \in \mathcal{R}_i$ is the facility holding item i that is the closest[1] to h, and the number $u_j(i)$ of clients requesting any content i attached to facility $j \in \mathcal{R}_i$, i.e., $u_j(i) = |\{h \in \mathcal{V} \text{ s.t. } m_h(i) = j\}|$, is such that $\sum_{i \in \mathcal{I}} u_j(i) \leq c_j$.

Note that our problem formulation is more complex than the traditional one, where the intersection between the sets of facilities and clients is null. Indeed, since in our settings any vertex of the graph G can host a facility (i.e., be a replica node for an item) or be a client (i.e., request an item that does not currently own), a vertex can assume both roles. Moreover, in the location theory literature, two copies of the same facility can be opened at the same location, in order to increase the capacity of a site. Instead, in our work a vertex of the graph can host only one copy of the same facility, as it is reasonable that a node stores only one copy of the same item.

Finding approximate solutions to the problem of multi-commodity capacitated facility locations, even in its (simpler) traditional formulation, is an open issue and little is known concerning heuristics that can be effectively implemented in practice. Thus, we take a simple approach that has been also discussed in [5]: a solution to the multi-commodity problem is built from the union of the solutions to individual single-commodity facility location problems. We transform the formulation from multi-commodity to single-commodity by solving the above problem for each item i ($i = 1, \ldots, I$) separately[2]. Then, we denote the subset of commodities hosted at node j by $\mathcal{I}_j$ and its cardinality by I_j, and we adopt two different techniques to verify the capacity constraints:

1) each opened facility (replica node) has a capacity that is allocated to each commodity individually: this translates into having a separate budget allocated to each commodity (item). The capacity constraints can be written as $u_j(i) \leq c_j / I_j, \forall i \in \mathcal{I}_j$, where we equally split the budget c_j available to facility j over all the commodities it hosts. In the following, we name such a technique *split capacity budget*;

2) we consider that the capacity of a facility is shared among the commodities it currently hosts, i.e., each replica

[1] As distance function, we take the Euclidean distance between the nodes.

[2] A single-commodity facility location problems reduces to the k-median problem when the number of facilities to be opened, k, is given [4].

node allocates a preset budget that is used to serve the requests by other nodes. We write the capacity constraints for this case as: $\sum_{i \in \mathcal{I}_j} u_j(i) \leq c_j$, and we refer to such a technique as *shared capacity budget*.

To solve such a problem, we resort to the local search heuristic detailed in [6], which finds a solution to the capacitated, single-commodity location problem that is one of the best known approximations to optimal replication and placement. Hereinafter we term such a heuristic *centralized* facility location (CFL) algorithm because it can only be executed in a centralized, synchronous environment. We consider the CFL algorithm to be a baseline against which we compare the results obtained by our approach.

Also, note that existing distributed approximation algorithms of the optimal solution to facility location problems either require global (or extended) knowledge of the network [6] or are unpractical [7]. Therefore, in the next section we propose a new approach that only requires local knowledge, which is acquired with simple measurements, and adapts to the system dynamics. In addition, our scheme provides load-balancing; it follows that, even in a static scenario, our distributed algorithm would not converge to a static configuration in which a fixed set of nodes is selected to host content replicas. As such, the traditional methods that are used in the literature to study the convergence properties and the locality gap of local search algorithms cannot be directly applied, which is the main reason for us to take an experimental perspective and validate our work through simulation.

III. Content replication

Armed with the insights on the problem formulation discussed in Sec. II, our mechanism mimics a local search procedure, by allowing replica nodes to execute one of the following three operations on the content: (1) handover, (2) replicate or (3) drop. However, unlike the traditional local search procedures, in our mechanism the three operations yield the solution to the content replication problem iteratively, albeit *asynchronously*. Furthermore, in our network system, replicate and handover are constrained operations: only vertexes that are connected by an edge to the current vertex hosting a content replica can be selected as possible replica locations. Thus, our operations are *local* and replicas can only move by one hop at the time in the underlying network graph.

In the following we describe our mechanism in terms of two objectives: content *replication* and *placement*. Indeed, the handover operation amounts to solving the optimal placement of content replicas, whose number is determined through the replicate and drop operations. For simplicity, we consider again that all users are interested in every content i ($i = 1, \ldots, I$) and we fix the time instant, hence we drop the time dependency from our notation.

Content replication: Let us define the workload of the generic replica node j for content i, $w_j(i)$, as the number of requests for content i served by j during its storage time. Also, recall that we introduced the value c_j as the capacity of node j and we provided a definition that suited the simplified, static scenario described in Sec. II. We now adapt the definition of c_j to the dynamic scenario at hand, as the reference volume of data that replica node j is willing to provide during the time it acts as a replica node, i.e., in a storage time τ. Then, with reference to (1), we denote by $f_j = \sum_{i \in \mathcal{I}_j} f_j(i)$ the cost that a node j must bear while acting as a facility for any content.

Given the load balance we wish to achieve across all replica nodes and the capacity constraints, the total workload for replica node j should equal c_j. Thus, we write f_j as:

$$f_j = c_j - \sum_{i \in \mathcal{I}_j} s(i) w_j(i) \tag{2}$$

where we recall that $s(i)$ is the size of content i. In other words, we let the cost associated with replica node j grow with the gap between the workload experienced by j and its capacity c_j.

Then, during storage time τ, the generic replica node $j \in \mathcal{R}$ measures the number of queries it serves, i.e., $w_j(i)\ \forall i \in \mathcal{I}_j$. When its storage time expires, the replica node j computes f_j and takes the following decisions: if $f_j > \epsilon$ the content is *dropped*, if $f_j < -\epsilon$ the content is *replicated*, otherwise the hand-over operation is executed (see below). Here, ϵ is a tolerance value to avoid replication/drop decisions in case of small changes in the node workload.

The rationale of our mechanism is the following. If $f_j < -\epsilon$, replica node j presumes that the current number of content replicas in the area is insufficient to guarantee the desired volume of data, hence the node replicates the content and hands the copies over to two of its neighbors (one each), following the placement mechanism described below. The two selected neighbors will act as replica nodes for the subsequent storage time. Instead, if $f_j > \epsilon$, node j estimates that the workload the current number of replicas can provide is exceeding the total demand, thus it just drops the content copy. Finally, if the experienced workload is (about) the same as the reference value, replica node j selects one of its neighbors to which to hand over the current copy, again according to the mechanism detailed next.

Replica placement: As noted in Sec. II, given the graph representing the network topology at a fixed time instant, the placement of $R = k$ replicas can be cast as a k-median problem. By applying the approximation algorithm in [6], in [8] we observed that the solution of such a problem for different instances of the topology graph yields replica placements that are instances of a random variable uniformly distributed over the graph. As a consequence, in a dynamic environment our target is to design a distributed, lightweight solution that closely approximates a uniform distribution of the replicas over the network nodes while ensuring load balancing among them. To this end, we leverage some properties of random walks and devise a mechanism, called *Random-Walk Diffusion (RWD)*, that drives the "movement" of replicas over the network.

According to RWD, at the end of its storage time τ, a replica node j randomly selects another node l to store the content for the following storage period, with probability $p_{j,l} = \frac{1}{d_j}$ if l is a neighbor of j, and 0 otherwise, where d_j is the current number of neighbors of node j. In this way, each replica performs a random walk over the network, by moving from

one node to another at each time step τ. Thus, we can apply the result stating that in a connected, non-bipartite graph, the probability of being at a particular node j converges with time to $d_j/(2|\mathcal{E}|)$ [9]. In other words, if the network topology can be modeled by a regular graph[3] with the above characteristics, the distribution of replicas moving according to a random walk converges to a stationary distribution, which is uniform over the nodes.

In general, real-world networks yield non-regular graphs. However, when V nodes are uniformly deployed over the network area and have the same radio range, the node degree likely has a binomial distribution with parameters $(V-1)$ and p, with p being the probability that a link exists between any two nodes [10].

For practical values of p and V in the scenarios under study, we verified that the node degree distribution is indeed binomial with low variance, i.e., all nodes have similar degree. It follows that a random walk provides an acceptable uniform sampling of the network nodes, hence the replica placement distribution well approximates the uniform distribution.

A similar result can be obtained also for clustered network topologies, where each cluster core results to be an expander graph [11]. In this case, a uniform replica placement over the nodes can be achieved within each of the network clusters, thus ensuring the desired placement in all areas where the user demand is not negligible.

Finally, we stress that the presence of R replicas in the network corresponds to R parallel random walks. As observed in [12], this reduces by almost a factor R the expected time to sample all nodes in the network, which is closely related to the time needed to approximate the stationary distribution by a constant factor [13]. It follows that, given a generic initial distribution of the replicas in the network, the higher the R, the more quickly the replica placement approximates a uniform distribution.

IV. Simulation scenario

We focus on a wireless network with high node density, i.e., $3.2 \cdot 10^{-4}$ nodes/m^2, on a square area of 1 km^2 unless otherwise specified, which results in $V = 320$ and an average node degree of 9.6 neighbors. Nodes move according to the stationary random waypoint model with an average speed of 1 m/s and a mean pause time of 100 s, a setting that is representative of pedestrian mobility. We also derived results using the SLAW mobility model [14], which has been shown to accurately model human mobility in real-world; due to lack of space, the results can be found in [14].

We assume nodes to be equipped with an 802.11 interface, with a 54 Mbps data rate and a radio range of 100 m. We do not simulate cellular access, however we account for the delay associated with the information download from the cellular network by assuming a throughput of 384 kbps, matching that typically provided by 3G technologies to mobile users.

The rate at which a node interested in a content generates queries for that item is set to $\lambda = 0.01$ requests/s. Also, we assume the presence of a content-location service that nodes can access to obtain the identity of the closest content replica (see, e.g., [15] and references therein). A query for the closest replica node is then propagated using sequence numbers to detect and discard duplicate queries, as well as an application-driven broadcast that optimally selects the forwarding nodes by leveraging the PGB technique [16]. Also, a TTL is included into queries, allowing them to travel 5 hops at most so as to prevent network flooding. Once reached by the request, the intended destination serves it, while other replica nodes ignore the query. At each hop, the identity of the last node that relayed the query is included in the message and recorded at the following forwarder. Thus, the path from the target replica node to the query source is backtracked at the application layer without resorting to ad hoc routing protocols, which would induce overhead or delay in the process. If a query fails (i.e., no answer is received after 2 s), a new request is issued, up to a total of 5 times. Finally, concerning the replication/drop parameters, the tolerance value ϵ used in the replication/drop algorithm is set to 5% of the node capacity budget, while the storage time τ is set to 100 s.

For each experiment, the results obtained through ns-2 simulations are averaged over 10 runs, each lasting around 3 hours of simulated time after a warm-up period of 500 s.

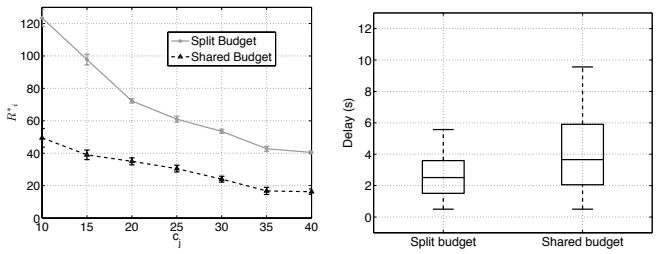
[3]A graph is regular if each of its vertices has the same number of neighbors.

V. Results

We organize the main results of our work in several sections that cover the parameter space we studied. To benchmark our distributed mechanism against the centralized approach discussed in Sec. II, we implement the CFL algorithm as follows. Given the network time evolution, we take a snapshot of the network topology every τ s. For every snapshot, we solve I separate single-commodity problems derived from (1), under both split and shared capacity budgets. To do so, we set $f_j(i) = c_j/I_j - u_j(i)$ and $f_j = c_j - \sum_{i \in \mathcal{I}_j} u_j(i)$ in the case of split and shared capacity budget respectively, with $u_j(i) = s(i)w_j(i)$.

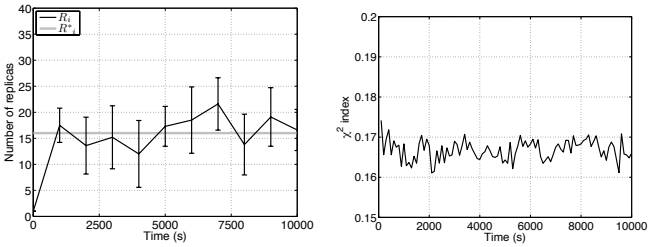
Benchmarking the replication scheme: First, we study the impact of the allocation of the node capacity budget. We take a numerical approach and focus on the CFL algorithm: our objective here is to determine the implications of *split* or *shared* capacity allocations as discussed in Sec. II. Later, we show the performance of our distributed replication scheme.

We run the CFL algorithm in presence of 4 items of 1 Mbytes each. We vary the value of budget of each node c_j from 10 Mbytes to 40 Mbytes, which, in the case of optimization with split capacity budget, means that each content is assigned a budget $c_j/4$. The optimal number of replicas per information item,denoted by R_i^*, is obtained by numerically solving the optimization problem in Def. 1, in both its split and shared capacity budget versions, and is shown in Fig. 1(a). Here and in the following, unless stated otherwise, the results refer to one of the four items; similar results were obtained for each of them. The plot clearly shows that, as higher budgets allow replica nodes to satisfy larger amounts of requests, increasing c_j reduces the need for replication thus leading to a lower number of replicas in the network. Using a common budget for all items (i.e., shared capacity budget), forces



(a) Number of replicas (b) Delay

Fig. 1. Numerical solutions of the optimization problems in terms of number of replicas (a) and query solving delay (b). In (b), we show the 5%, 25%, 50%, 75% and 95% percentiles



(a) Number of replicas (b) χ^2 index

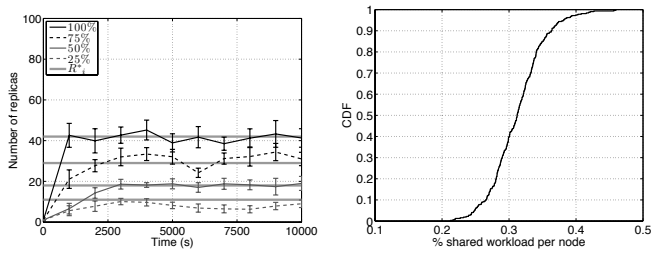
Fig. 2. Numerical solutions of the optimization problems, and comparison against our replication scheme: temporal evolution of the number of replicas (a), and of the χ^2 index (b)

replications only when the total workload for all items exceeds the budget. Conversely, optimization with split capacity budget uses separate budgets for each content and, thus, results in more frequent violations of such constraints.

Now, intuitively, more replicas should imply higher chances for queries to be satisfied through device-to-device communications. In Fig. 1(b) we show the most important percentiles of content access delay, for $c_j = 40$ Mbytes. Contrary to the intuition, our results indicate that the advantage granted by a high number of replicas under the split capacity is quite negligible: indeed, the lower number of replicas deployed by the shared capacity allocation suffices to satisfy most of the requests generated by nodes in the ad hoc network.

In summary, our findings pinpoint that the replication mechanism with shared capacity constraints is a suitable approach. Beside experimental results, there are also practical reasons to opt for shared capacity constraints. Indeed, in the split capacity case, a budget has to be assigned to each item currently stored by a replica node, which is a quantity that may vary over time. As a consequence, content replicas may not be suitably handled if the remaining capacity available to a node is not appropriately re-distributed. Furthermore, it would be unfeasible to ask a user to select a service budget to allocate to every possible item she will ever replicate. In the following we will therefore focus on the *shared* capacity budget only.

Next, we simulate our distributed replication scheme when each node has a budget of $c_j = 40$ Mbytes. As shown in Fig. 2(a), the scheme well approximates the results obtained by solving the optimization problems in a centralized setting: indeed, the number of replicas R_i generated by our scheme is very close to the optimal value R_i^*. We then study the similarity between the replica placement achieved by our technique and that obtained with the CFL algorithm. To do so,



(a) Number of replicas (b) Total workload

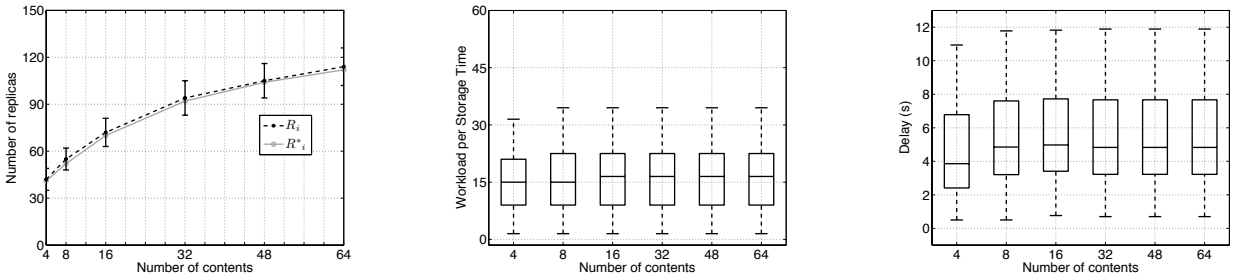
Fig. 3. Impact of content popularity on the replication with shared capacity, in terms of number replicas, workload distribution

we employ the well-known χ^2 goodness-of-fit test on the inter-distance between content replicas As depicted in Fig. 2(b), the χ^2 error we obtain is well below the value (namely, 23.685) needed to accept the null hypothesis that the two distributions are the same at a 95% confidence level.

Impact of the content characteristics: We now study the scenario when not all nodes are interested in a content. In such a situation, a node stores a replica of the content only if it is interested in the item. If a node attempts to hand over the content to an uninterested node, the request will be denied and a different node will have to be selected. Fig. 3(a) shows that the number of replicas for item i, R_i, generated by our scheme oscillates around the optimal value determined by the CFL algorithm for the same item, R_i^*, even when i is characterized by low popularity (the popularity levels are reported in the figure legend). Moreover, the workload remains evenly shared among replica nodes: Fig. 3(b) shows that each node serves at least 0.2% of the total workload and 98% of nodes serve less than 0.4% of the total workload. The load distribution is thus quite dense around 0.3%, i.e., $\frac{1}{V}$ corresponding to a perfectly fair workload distribution among nodes.

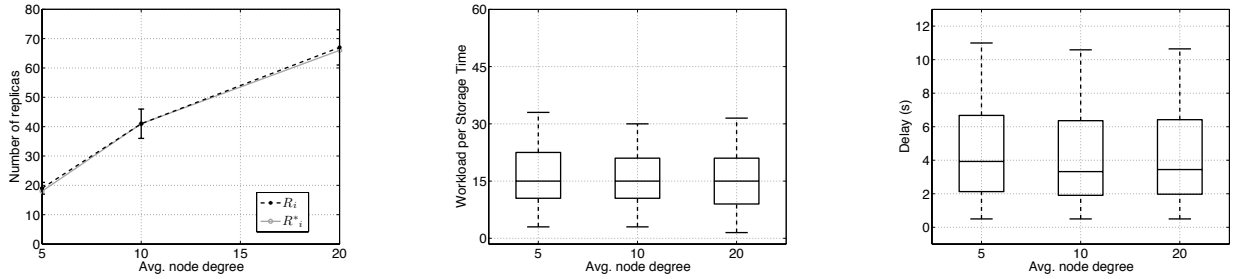
Scalability: We now study the impact of the number of items, network density, and network size on the system performance. We first evaluate the performance when the cardinality of the item set varies between 4 and 64. Fig. 4(a) shows the number of replicas per item generated in the system, which grows as the size of the information set increases. Indeed, a larger content set implies that nodes tend to store more items on average; however, their capacity budget c_j remains constant, and is shared among all items they store. Thus, focusing on one single content, each replica node for that content will be able to serve fewer and fewer queries as the number of available items increases. As a consequence, more replicas for the same content are needed in order to meet the constraint on the capacity budget, hence to keep the workload constant, as depicted in Fig. 4(b). Fig. 4(c) shows the effect that the number of information item has on the service provisioning delay. The increase of the delays is imputable to the heavier traffic on the channel, that results in collisions and retransmissions of the information replies.

We then study the effect of the network density, measured as the average node degree, which is increased up to 20. Fig. 5(a) shows that the number of replicas increases according to the optimal number of facilities computed by the CFL local search algorithm. Indeed, the increased presence of neighbors induces



(a) Number of replicas (b) Workload (c) Delay

Fig. 4. Impact of content set cardinality on the replication in terms of number of replicas, workload distribution, and delay. In (b) and (c), we show the 5%, 25%, 50%, 75% and 95% percentiles



(a) Number of replicas (b) Workload (c) Delay

Fig. 5. Impact of network density on the replication with shared capacity, in terms of number of replicas, workload distribution, and delay. In (b) and (c), we show the 5%, 25%, 50%, 75% and 95% percentiles

a higher query load in the network: in order to satisfy the new demand, and yet fulfill the per-node workload constraint, additional nodes must become providers for each content. The availability of additional replica nodes allows them to experience a practically unchanged workload (Fig. 5(b)), and a similar delay for successful content requests (Fig. 5(c)).

Comparison to other approaches. We now consider information items to be associated to different popularity levels, and compare the performance of our replication scheme with that of the square-root replication strategy [17]. According to such a strategy, the allocation percentage for a content i is proportional to the square root of the total demand per second for that content. In [17], it has been proved that square-root replication is optimal in terms of number of solved queries.

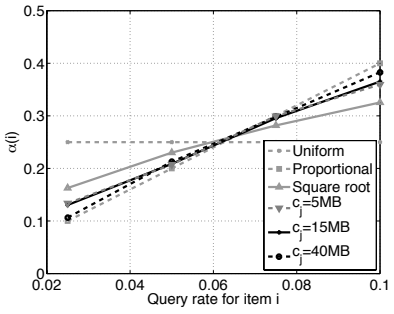
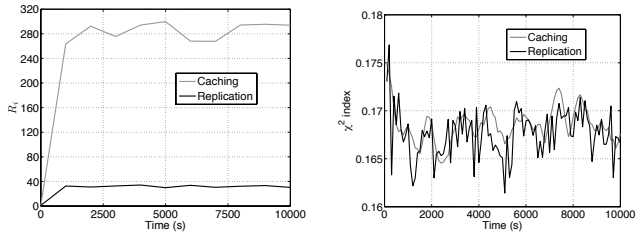


Fig. 6. Fraction of replicas for each of the four items, in comparison with uniform, proportional and square-root allocation

Fig. 6 shows the fraction of the total number of replicas of item i, versus the associated query rate $\pi(i)V\lambda$, for $I = 4$ and c_j=$\{5, 15, 40\}$ Mbytes. The plot compares our scheme with: (i) the square-root strategy, (ii) a uniform strategy, which allocates the same number of replicas per item, and (iii) a proportional strategy, where the number of replicas is proportional to the content popularity. Our solution achieves an allocation in between the square-root and proportional distributions, while it is far from that obtained under the uniform strategy. This suggests that our replication mechanism well approximates the optimal replication strategy. In particular, when c_j is higher, i.e., replica nodes are more generous in reserving resources to serve requests, the allocation tends to follow a proportional distribution. Conversely, in presence of lower values of c_j the allocation better fits the square-root rule.

Since our replication scheme roughly achieves the result obtained by a square-root allocation, it is reasonable to wonder why a different approach to content replication is required. First of all, we have different objectives than that of [17]: load-balancing, for example, requires an additional layer to complement the square root allocation scheme, which instead we achieve as part of our design. Furthermore, the distributed version of the replication algorithms proposed in [17] has some limitations that render them less suitable to be deployed in a mobile, wireless environment. The simple path replication scheme catering to low storage requirements, just like our scheme, substantially over/undershoots the optimal number of replicas. The other approaches discussed in [17] are better at converging to an optimal number of replicas but require the bookkeeping of large amounts of information. Finally, the design and the evaluation of such algorithms in [17] are performed in a static wired environment and do not take into account the dynamics typical of a mobile network, such as that we consider.

As the second step, we benchmark our replication mechanism with a simple caching scheme. We consider a *pull-based* caching mechanism: a node issues a query for an item



(a) Number of replicas (b) χ^2 index

Fig. 7. Performance of caching and replication mechanisms in terms of (a) number of replicas and (b) χ^2 index, for 100% content popularity and 100 s content validity time

of interest. Such a request can travel up to h hops and if it is not satisfied within a timeout, the content is fetched directly from the cellular network. After having successfully obtained the content, nodes store it until the corresponding validity time expires and serve requests through device-to-device communication. Note that, if a node is not interested in an item, it will not participate to the caching process, including content transfer and storage. In summary, with the above mechanism, information spreads from one node to another in a manner that loosely resembles an epidemic diffusion process.

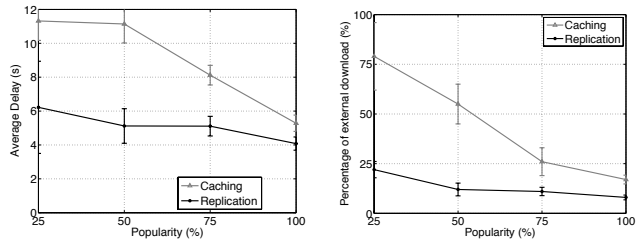
We remark that such a caching scheme eventually achieves full content replication; instead, our goal is to find the optimal number of replicas that minimizes content access costs, while guaranteeing load balancing. Additionally, in the caching scheme, nodes simply discard expired content, while in ours replica nodes are in charge of downloading up-to-date versions of the content. It is well known that pull-based caching approaches are sub-optimal during the bootstrap phase of the content delivery process. The caching scheme we evaluate here partially overcomes this problem by allowing nodes to fetch content through the cellular network. However, it is reasonable to expect a large number of "external" data transfers: as a consequence, access congestion may arise also at the cellular level. Finally, we note that when the content is unpopular, the diffusion process is even slower and the above negative effects are amplified.

We now study the behavior of the replication and caching schemes over time, assuming a content validity time of 100 s and a single replica in the network at the beginning of the simulation. The number of replicas present in the system over time is depicted in Fig. 7(a). As expected, by achieving full replication, the caching strategy is more expensive than our replication scheme, in terms of storage requirements. One may argue that fewer content replicas may lead to a suboptimal placement, while full replication ensures that the content resides where the demand is. The results in Fig. 7(b), however, show that such an additional storage space usage does not lead to any significant advantage in terms of the quality of replica placement. The χ^2 index obtained by comparing the geographical distribution of replicas under the two schemes with that of the CFL solution is essentially equivalent.

We now compare the performance of caching and replication considering the following metrics: (i) query solving delay, intended as the time elapsed from the instant when a node sends the first query until the request is fulfilled, by either a replica node or the cellular network; (ii) percentage of external downloads, i.e., queries that resulted in an external download, with respect to the overall requests generated in the network.

Fig. 8(a) shows the average delay (along with the 95% confidence interval) for the replication and caching scheme as the content popularity varies. The replication scheme outperforms the caching mechanism, and the difference in the relative performance is amplified (in favor of replication), as the content popularity decreases. Indeed, as content popularity decreases, fewer nodes participate in the diffusion process that underlies the caching scheme. As such, nodes have to wait longer for their queries to be satisfied and, in general, they end up downloading the content from the cellular network. Instead, when the content popularity is high, the epidemic-style diffusion process performs better, and the delay decreases. Fig. 8(b) shows that the content diffusion process is hindered by content popularity: hence, nodes resort to the cellular network to compensate for the delays of device-to-device communication. By approximating optimal content replication and placement, our mechanism reduces the content access costs, in terms of congestion. Instead, the caching mechanism does not alleviate access congestion: i) nodes in the vicinity of a content replica "collide" to obtain the content through device-to-device communication, and ii) nodes resorting to the cellular infrastructure also compete for bandwidth. These intertwined aspects are exacerbated when the content becomes stale: with our approach, few replica nodes take care of the update process, while, with the caching scheme we study here, the whole content diffusion process has to start over.

In light of the above results, our content replication scheme clearly emerges as a simple, efficient and performing alternative to traditional mechanisms. By controlling the number and the placement of replicas, our mechanism appears to be suitable especially when content popularity is not 100%, for both performance and cost-related reasons.



(a) Query solving delay (b) External downloads

Fig. 8. Performance of caching and replication mechanisms in terms of query solving delay (a) and percentage of cellular downloads (b), when the content popularity varies between 25% and 100%

VI. Related work

Simple, widely used techniques for replication are gossiping and epidemic dissemination [18]–[20], where the information is forwarded to a randomly selected subset of neighbors. Although our RWD scheme may resemble this approach in that a replica node hands over the content to a randomly chosen neighbor, the mechanism we propose and the goals it achieves (i.e., approximation of optimal number of replicas) are significantly different.

Another viable approach to replication is represented by probabilistic quorum systems for information dissemination and sharing [15], [21]. In particular, in [15] the authors propose a mechanism akin to random walks to build such quorums. However, the problem statement of quorum systems differs substantially from ours (i.e., facility location). We use location theory to model the problem of determining where and, crucially, how many content replicas to place in a dynamic network. Instead, the construction of quorums caters at the following quality metrics: intersection probability between individual quorums, access cost in terms of number of messages (and not distance) and traffic load (this latter being a goal that we also aim at). Node grouping is also exploited in [22], [23], where groups of nodes with stable links are used to cooperatively store and share information items. The schemes in [22], [23], however, require an a-priori knowledge of the query rate, which is assumed to be constant in time. Note that, on the contrary, our lightweight solution can cope with a dynamic demand, whose estimate by the replica nodes is used to trigger replication. We point out that achieving content diversity is the goal of [24] too, where, however, cooperation is exploited among one-hop neighboring nodes only.

Threshold-based mechanisms for content replication are proposed in [25], [26]. In particular, in [25] it is the original server that decides whether to replicate content or not, and where. In [26], nodes have limited storage capabilities: if a node does not have enough free memory, it will replace a previously received content with a new one, only if it is going to access that piece of information more frequently than its neighbors up to h-hops. Our scheme significantly differs from these works, since it is a totally distributed, extremely lightweight mechanism that ensures a replica density that autonomously adapts to the network dynamics.

Finally, we point out that the RWD scheme was first proposed in our work [8]. That paper, however, besides being a preliminary study, focused on mechanisms for content handover only.

VII. Conclusion

We addressed the joint problem of establishing the number of content replicas to deploy in a wireless network and finding their most suitable location. We studied the above problems through the lenses of the facility location theory and proposed a distributed, lightweight scheme that builds on local search approximations of the multi-commodity capacitated facility location problem and parallel random walk diffusion in non-regular graphs. We showed that, despite its simplicity and the fact that it only leverages local measurements, our solution approximates with high accuracy the solution attained by optimal centralized algorithms, while also guaranteeing a fair load balancing at the nodes.

References

[1] [Online]. Available: http://www.nytimes.com/2009/09/03/technology/companies/03att.html?_r=1

[2] A. Derhab and N. Badache, "Data replication protocols for mobile ad-hoc networks: A survey and taxonomy," *IEEE Comm. Surveys & Tutorials*, vol. 11, no. 2, pp. 33–51, June 2009.

[3] H. Chen, Y. Xiao, and X. Shen, "Update-based cache access and replacement in wireless data access," *IEEE Trans. on Mob. Comp.*, pp. 1734–1748, 2006.

[4] P. B. Mirchandani and R. L. Francis, *Discrete Location Theory*. New York, NY: John Wiley and Sons, 1990.

[5] R. Ravi and A. Sinha, "Multicommodity facility location," in *ACM/SIAM Symposium on Discrete Algorithms*, New Orleans, LA, Jan. 2004.

[6] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit, "Local search heuristic for k-median and facility location problems," in *ACM STOC*, Heraklion, Crete, Greece, July 2001.

[7] T. Moscibroda and R. Wattenhofer, "Facility location: Distributed approximation," in *ACM PODC*, Las Vegas, NV, July 2005.

[8] C. Casetti, C.-F. Chiasserini, M. Fiore, C.-A. La, and P. Michiardi, "P2P cache-and-forward mechanisms for mobile ad hoc networks," in *IEEE ISCC*, Sousse, Tunisia, July 2009.

[9] L. Lovasz, "Random walks on graphs: A survey," *Combinatorics*, vol. 2, pp. 1–46, 1993.

[10] R. Hekmat and P. V. Mieghem, "Degree distribution and hop count in wireless ad hoc networks," in *IEEE International Conference on Networks (ICON)*, New York, NY, 2003, pp. 603–609.

[11] I. Benjamini, G. Kozma, and N. Wormald, "The mixing time of the giant component of a random graph," Oct. 2006. [Online]. Available: http://arxiv.org/abs/math/0610459

[12] K. Li, "Performance analysis and evaluation of random walk algorithms on wireless networks," in *IEEE International Symposium on Parallel & Distributed Processing (IPDPSW)*, Atlanta, GA, Apr. 2010, pp. 1–8.

[13] J. Kahn, J. Kim, L. Lovász, and V. Vu, "The cover time, the blanket time and the Matthews bound," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, Redondo Beach, CA, Nov. 2000, pp. 467–475.

[14] C.-A. La, P. Michiardi, C. Casetti, C.-F. Chiasserini, and M. Fiore, "Content replication and placement in mobile networks," Eurecom Institut, Tech. Rep., 2011.

[15] R. Friedman, G. Kliot, and C. Avin, "Probabilistic quorum systems in wireless ad hoc networks," *ACM Trans. on Comp. Syst.*, vol. 28, no. 3, Sept. 2010.

[16] V. Naumov, R. Baumann, and T. Gross, "An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces," in *ACM MobiHoc*, Florence, Italy, May 2006.

[17] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," in *ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002.

[18] R. Bakhshi, D. Gavidia, W. Fokkink, and M. van Steen, "An analytical model of information dissemination for a gossip-based protocol," *Comp. Networks*, vol. 53, no. 13, pp. 2288–2303, 2009.

[19] H. Hayashi, T. Hara, and S. Nishio, "On updated data dissemination exploiting an epidemic model in ad hoc networks," in *Lecture Notes in Computer Science*, vol. 3853, Dec. 2006, pp. 306–321.

[20] M. Hauspie, A. Panier, and D. Simplot-Ryl, "Localized probabilistic and dominating set based algorithm for efficient information dissemination in ad hoc networks," in *IEEE MASS*, Fort Lauderdale, FL, Oct. 2004.

[21] J. Luo, P. Eugster, and J.-P. Hubaux, "PILOT: Probabilistic lightweight group communication system for mobile ad hoc networks," *IEEE Trans. on Mob. Comp.*, vol. 3, no. 2, Apr.-June 2004.

[22] T. Hara, "Effective replica allocation in ad hoc networks for improving data accessibility," in *IEEE Infocom*, Anchorage, AK, May 2001.

[23] T. Hara, Y.-H. Loh, and S. Nishio, "Data replication methods based on the stability of radio links in ad hoc networks," in *Database and Expert Systems Applications (DEXA)*, 2003.

[24] L. Yin and G. Cao, "Balancing the tradeoffs between data accessibility and query delay in ad hoc networks," in *IEEE SRDS*, Bilbao, Spain, Sept. 2004.

[25] V. Thanedar, K. C. Almeroth, and E. M. Belding-Royer, "A lightweight content replication scheme for mobile ad hoc environments," *Networking*, May 2004.

[26] M. Shinohara, H. Hayashi, T. Hara, and S. Nishio, "Replica allocation considering power consumption in mobile ad hoc networks," in *IEEE International Conference on Pervasive Computing and Communications Workshop (PERCOMW)*, Pisa, Italy, Mar. 2006.