

ContentPlace: Social-aware Data Dissemination in Opportunistic Networks *

Chiara Boldrini
IIT-CNR
Via G. Moruzzi 1
56124, Pisa, Italy
chiara.boldrini@iit.cnr.it

Marco Conti
IIT-CNR
Via G. Moruzzi 1
56124, Pisa, Italy
marco.conti@iit.cnr.it

Andrea Passarella
IIT-CNR
Via G. Moruzzi 1
56124, Pisa, Italy
andrea.passarella@iit.cnr.it

ABSTRACT

This paper deals with data dissemination in resource-constrained opportunistic networks, i.e., multi-hop ad hoc networks in which simultaneous paths between endpoints are not available, in general, for end-to-end communication. One of the main challenges is to make content available in those regions of the network where interested users are present, without overusing available resources (e.g., by avoiding flooding). These regions should be identified dynamically, only by exploiting *local* information exchanged by nodes upon encountering other peers. To this end, exploiting information about *social users' behaviour* turns out to be very efficient. In this paper we propose and evaluate ContentPlace, a system that exploits dynamically learnt information about users' social relationships to decide where to place data objects in order to optimise content availability. We define a number of social-oriented policies in the general ContentPlace framework, and compare them also with other reference policies proposed in the literature.

Categories and Subject Descriptors

C.2.1 [Computer-Communication Networks]: Network Architecture and Design—*Wireless Communication*; C.2.2 [Computer-Communication Networks]: Network Protocols

General Terms

Algorithms, Design, Performance

Keywords

opportunistic networks, data dissemination, social-oriented networking, performance evaluation

1. INTRODUCTION

Opportunistic networks [14] are a very promising networking scenario originated from the legacy MANET paradigm.

*This work was partially funded by the European Commission under the HAGGLE (027918) and SOCIALNETS (217141) FET Projects.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MSWiM'08, October 27–31, 2008, Vancouver, BC, Canada.
Copyright 2008 ACM 978-1-60558-235-1/08/10 ...\$5.00.

As in MANETs, in opportunistic networks nodes build a self-organising ad hoc network without requiring any pre-existing infrastructure. However, unlike MANETs, opportunistic networks do not assume that the topology of the network is stable enough to provide an Internet-like routing substrate and enables communications even in presence of prolonged partitions and disconnections. To this end, mobility is seen as a resource to bridge disconnections, rather than a problem to deal with. According to the store-carry-and-forward paradigm, nodes opportunistically exploit any contact ¹ with other peers to exchange messages, if the peer is deemed a good candidate to bring the message closer to the appropriate destination(s).

In opportunistic networks *data-centric* approaches are very promising. Users interested in data objects (files, clips, advertisements, news, ...) might be completely unaware of other users that generate the content they wish to get, and, vice versa, users generating content might ignore who is interested in their data. The network might be so unstable that content producers and consumers might never be connected together. Therefore, it is important to disseminate content in the network so as to reach possibly interested users. This should be achieved by network protocols themselves, as applications might be unable to identify such users as in the case of the conventional Internet. At the same time, as opportunistic networks work on resource constrained mobile devices, the content dissemination process should be aware of resource consumption. These requirements call for distributed, resource-aware content dissemination systems.

In the area of opportunistic networks, knowledge about the social behaviour of users is proving to be an extremely valuable piece of information to design efficient and resource-saving networking systems (see, e.g., HiBOP [5] for a routing solution, and the work in [20] for a social-oriented pub/sub system). As users are mobile, and carry devices with them, users' social behaviour is a valid indicator of users' mobility patterns and is thus an important context information to predict future contacts between users. In this paper, we apply this general idea to the content dissemination problem, by designing and evaluating social-oriented dissemination policies within the ContentPlace framework.

In ContentPlace, each user advertises the data objects they are interested in upon making contact with other nodes. Also, they exchange short summaries of the data objects they are currently carrying with. This *local* information is exploited to implement a totally decentralized decision pro-

¹In opportunistic networks, contact means direct one-hop communication between a couple of nodes.

cess about which data objects should be cached (see Section 3 for more details). Basically, a node assigns to each data object a utility value, computed according to the needs of the social communities the node is in touch with (different solutions are proposed in Section 4). If the utility is high enough, the data object is fetched. In Section 5 we compare the proposed social-oriented policies against each others, and against well-known non-social policies. Our main results can be summarised as follows. In general, social-oriented policies do provide significant benefits over non-social policies. In addition, the social-oriented policies that perform best are the ones that learn and exploit the nodes’ *social roles* in the network.

2. RELATED WORK

Content dissemination systems have been proposed with regard to legacy Internet networks [1], and also with respect to conventional MANETs [18, 6]. In general, these systems assume that network paths are rather stable, and in some cases generate a significant amount of traffic to maintain knowledge of other nodes’ caches. Therefore, they are not suitable to opportunistic networks.

With regard to opportunistic networks, content dissemination has been the subject of a few recent papers. To the best of our knowledge, the closest approaches to ContentPlace have been designed within the PodNet [13] and Huggle [20] projects. As described in detail in Section 5, we use an application and evaluation scenario similar to that defined for PodNet: users advertise the data objects they are interested into, and, when two nodes meet, they decide which data object to exchange based on the information gathered about users’ interests. While the work in [13] proposes and compares simple heuristics for data object exchanges, ContentPlace elaborates a more complete social-oriented framework. We actually compare ContentPlace with the best heuristics identified in [13], showing the advantage of the social-aware dimension. The work in [20] identifies social communities, and “hubs” within communities (i.e., nodes with the highest number of social links inside the community). An overlay network is then built between hubs, that act as the broker overlay of a standard pub/sub topic-based system. ContentPlace assumes the same community-detection mechanisms of [20] but does not rely on any overlay infrastructure, which, in opportunistic networks, might be costly to maintain and rather unstable. Note that similar remarks hold also with respect to solutions for delay-tolerant networks based on strong predictability of the future topological structure, such as the multicasting approach proposed in [21].

ContentPlace takes inspiration from the vast literature on utility-based Web caching [1], which has been also applied to both MANETs and WLANs (e.g., [19, 15]). Utility-based routing for opportunistic networks has been proposed in [2]. ContentPlace inherits the general framework of this body of work, as the ContentPlace utility function is an instance of the general form proposed in the literature. The closest work in this area is [2], which however is focused on unicast routing, and thus addresses a different problem. Another novelty of ContentPlace is the fact that its utility function is defined based on the social behaviour of the users. Utility-based content replication systems have been recently proposed for wired networks [7, 11]. However, these solutions relies on Internet overlays, and are not directly applicable to opportunistic networks.

This paper follows up a preliminary design and evaluation of ContentPlace that we have presented in [4]. While [4] focuses on the definition of the formal ContentPlace utility-based framework, and provides an initial evaluation in a simplified scenario, in this paper we explore much more in detail the *social-oriented* aspects of ContentPlace (Section 4). Furthermore, the performance evaluation part of this paper is totally devoted to analyse these fundamental aspects of ContentPlace, by taking into account more complex users social behaviours. This further differentiates this paper from [4].

3. CONTENTPLACE GENERAL DESIGN

This section provides necessary background information required to present the main contribution of this paper by recalling the target application scenario and the main design features of the ContentPlace system.

3.1 Application scenario

The application scenario we target is similar to the one used in PodNet [13], named “podcasting for ad hoc networks”. As in the typical opportunistic networking paradigm, we consider a number of mobile users whose devices cannot be encompassed by a conventional MANET. Instead, communication is achieved by opportunistically exploiting pair-wise contacts between users to exchange messages, and bringing them towards eventual destinations. Sporadic contacts of users with point of access to the Internet (e.g., WiFi hotspots) are possible although not necessary. In podcasting applications, data objects (e.g., MP3 files, advertisements, software updates, ...) are organised in different *channels* to which users can subscribe. We assume that the channel(s) of a data object is decided by the source of the object at the generation time. Data objects might be generated from within the Internet, and “enter” the opportunistic network upon sporadic contacts of users with Internet Access Points. Or, data objects may be generated dynamically by the users of the opportunistic network according to the Web 2.0 model (e.g., users may wish to share pictures taken with their mobile phones). ContentPlace is responsible for the two main tasks of content dissemination, i.e., i) managing subscriptions to channels, and ii) bringing data objects to subscribed users (content distribution).

3.2 ContentPlace framework

At the high level, the rationale of ContentPlace is as follows. Since stable network structures cannot be assumed, ContentPlace only exploits direct interactions between nodes (contacts) to gather information about the users’ subscriptions and current data objects availability. Each node uses this knowledge to decide which data objects “seen” on other nodes should be locally replicated, according to a *replication* policy. The main challenge of ContentPlace is defining a *local* replication policy (i.e., a policy that does not require precise information about the global state of the network) that achieves a *global* performance target (such as, for example, maximising the hit rate, the per-user fairness, the network efficiency).

More in detail, ContentPlace subscription management works as follows. Nodes just advertise the set of channels the local user is subscribed to upon encountering another node. As will be clear in the following, no per-node state is necessary, and thus unsubscription messages are not required. As far as content distribution is concerned, the core of ContentPlace is the definition of the replication policy,

which can be summarised as follows. ContentPlace defines a utility function by means of which each node can associate a utility value to any data object. When a node encounters a peer, it computes the utility values of all the data objects stored in the local and in the peer’s cache². Then, it selects the set of data objects that maximises the *local* utility of its cache, without violating the considered resource constraints (e.g., max cache size, available bandwidth, available energy, ...). The node fetches the selected objects that are in the peer’s cache, and discards the locally stored objects that are not in the selected set anymore. Finally, a user receives a data object it is subscribed to when it is found in an encountered node’s cache. As discussed in more detail in [4], the set of objects to store in the local cache upon each contact can be found by solving the following multi-constrained 0-1 knapsack problem:

$$\begin{cases} \max & \sum_k U_k x_k \\ \text{s.t.} & \sum_k c_{jk} x_k \leq 1 \quad j = 1, \dots, m, \\ & x_k \in \{0, 1\} \quad \forall k \end{cases}, \quad (1)$$

where k denotes the k -th object that the node can select, U_k its utility, c_{jk} the percentage consumption of resource j related to fetching and storing object k , m the number of considered resources, and x_k the problem’s variables. When the number of managed resource (m) is not big (which is quite reasonable), solving such problems is very fast from a computational standpoint [12]. Such a solution is therefore suitable to be implemented in resource constrained mobile devices.

It is clear that the core of the content distribution mechanism is the definition of the utility function. In the following sections, we discuss how information about the social behaviour of users can be leveraged to this end.

3.2.1 Utility function

To have a suitable representation of the users’ social behaviour, we take inspiration from the caveman model proposed by Watts [17], which is a reference point in the field of social behaviour modelling. We assume that users can be grouped in communities. Users belonging to the same community have strong social relationships with each other. In general, users can belong to more than one community (a working community, a family community, etc.), each of which is a “home” community for that user. Users can also have relationships outside their home communities (“acquainted” communities). We assume that people movements are governed by their social relationships, and by the fact that communities are also bound to particular places (i.e., the community of office colleagues is bound to the office location). Therefore, users will spend their time in the places their home communities are bound to, and will also visit places of acquainted communities.

Different communities will have, in general, different interests. Therefore, the utility of the same data object will be different for different communities. Based on this remark, the utility of a data object computed by a node is made up of one *component* for each community its user has relationships with, be it either a home or an acquainted community. Formally, the utility function is defined as follows³:

²Hereafter, we use the term cache to denote a memory buffer that a node contributes to the ContentPlace system.

³For easy of reading, with respect to Equation 1 we drop here the k index, as this does not impact the clarity of the discussion.

$$U = \omega_l u_l + \sum_{i \neq l} \omega_i u_i = \sum_i \omega_i u_i, \quad (2)$$

where u_i is the i -th component and ω_i measures the social strength of the relationship between the user and the i -th community. Finally, in Equation 2 we stretch a bit the concept of community, and represent the *local* user just as another community the user is in touch with (i.e., the utility for the local user is represented by u_l). Note that the definition of the weights ω_i defines the social-oriented behaviour of ContentPlace. As the main focus of this paper is on this aspect, we now briefly describe the definition of the utility components, and discuss in detail the definition of weights in Section 4.

3.2.2 Utility components

ContentPlace uses the same definition for all utility components. In this paper we consider a simplified version of the general function defined by ContentPlace, and we also assume that i) the cache space is the only considered resource, and ii) data objects never expire. See [4] and the associated report for the discussion of more general cases. Inspired by the Web-caching literature [1], the utility of a data object for a community is the product of the object’s access probability from the community members (p_{ac}) by its cost c , divided by the object’s size. The cost is measured as a monotonically decreasing function of the object’s availability in the community (denoted as p_{av}), as the more the object is spread, the less it is costly to find it, the less the utility of further replicating it. Dividing by the object’s size is also common in the Web-caching literature, as it also allows very simple approximations of the multi-constrained knapsack problem defined by Equation 1. Specifically, we use the following definition:

$$u = \frac{p_{ac} \cdot f_c(p_{av})}{s} = \frac{p_{ac} \cdot e^{-\lambda p_{av}}}{s}, \quad (3)$$

In Equation 3 we use an exponential function as cost function, which achieves a fairer behaviour with respect to a (more intuitive) linear decay, as shown in [4].

4. CONTENTPLACE SOCIAL DESIGN

The use of the social weights in Equation 2 permits full flexibility in the ContentPlace design. Specifically, it allows us to define and compare different social-oriented policies, as well as, in the limiting case, *non-social* policies such as a greedy behaviour. Although clearly not exhaustive, the set of social-oriented policies we compare covers a fairly large spectrum of possible (reasonable) definitions. Overall, the global goal we wish to achieve is to optimise the hit rate for *all* users in *all* communities. Different policies clearly give different importance to the utility components. The comparison we carry out shows which are the components that provide the best behaviour with respect to the desired global target.

4.1 Policies definition

We consider the following policies:

Most Frequently Visited (MFV) Each community is given a weight proportional to the time spent by the user in that community. Specifically, if t_i is the time spent by the user in the i -th community since the system start-up, then w_i is equal to $t_i / \sum_i t_i$. With this policy, dissemination decisions of a user favour the communities the user is more likely to get in touch with.

Most Likely Next (MLN) The weight of the i -th community is equal to the probability of visiting the i -th

community, conditioned to the fact that the user is in the current community. Hereafter, this probability is referred to as $P(c_i|CC)$, where C denotes the current community. The weight of the current community is set to 0. With this strategy, users favour the communities they will be visiting next, but do not contribute to data dissemination within their current community.

Future (F) As in MLN, the weight of the current community is set to 0. However, the weight of all other communities is set as in MFV, i.e., is proportional to the average time spent by the node in that community. With respect to MLN, with F we consider not only the most likely community for the next visit, but all the communities the user is in touch with.

Present (P) The weight of the current community is set to 1, and the weights of all other communities are set to 0. With this strategy, users always behave as members of the community they are currently in, and do not favour any other community.

Uniform Social (US) All the communities the user gets in touch with are given equal weight. Therefore, w_i is equal to 1 for the home and the acquainted communities.

Clearly, it would be possible to consider a number of additional strategies, by modifying the definition of the weights. The strategies we have selected are representative of several *reference* behaviours. In the US policy all communities a user is in touch with are given the same importance. In MFV the importance of a community is proportional to the time spent by the user in the community. P and MLN can be seen as opposite extreme customisations of MFV. In P only the current community is given importance. This is a MFV customisation which looks exclusively at the current “role” of the user, as a member of the current community, without any “look-ahead” behaviour. In MLN the current community is not given any importance, and only future communities are considered. MLN is thus a customisation of MFV with an extreme “look-ahead” behaviour. Finally, F is an intermediate policy between MFV and MLN. It keeps the “look-ahead” behaviour of MLN, because it does not consider the current community. However, as in MLN, it considers *all* the other communities the user is in touch with, and not only the most likely one for the next visit.

Finally, for comparison purposes, we consider two non-social policies as well, and specifically a greedy and a uniform policy. In the greedy policy all weights but the one of the local user are set to 0. In the uniform policy all the weights are equal. As a minor, but important, remark, note that in this case a lot of data objects ends up having the same utility. In this case, a node breaks ties by choosing data objects according to a uniform distribution.

4.2 Social parameter estimation

The ContentPlace social-oriented policies require online, dynamic estimation of the utility components’ parameters, as well as an estimation of the parameters required to compute the components’ weights. In this section we describe how this can be achieved by avoiding any form of (controlled) flooding, such as that implemented by Epidemic Routing [16], which easily saturate networking resources [10]. Clearly, this choice calls for a trade-off between estimation accuracy and network overhead.

A necessary pre-requisite to estimate the parameters is to detect the communities in the network, and to enable nodes

to understand in which community they are currently roaming. Fortunately, there are promising results about autonomous community detection in opportunistic networks ContentPlace can rely on, such as [8, 9]. These mechanisms allow nodes of an opportunistic network to i) be aware of the communities they belong to and have acquaintance with, and ii) be aware of the community in which they are currently in at any given point in time. We assume that one of these mechanisms is in operation.

4.2.1 Estimation of the social weights

The communities’ weights defined in Section 4 can be easily computed thanks to the community detection features and, in some cases, by measuring the time spent by nodes in the different communities, and monitoring transitions between communities. Specifically, the strategies P, US and MLN just require community detection. This trivially holds true for P and US. MLN requires an estimate of the conditional probability of moving to future communities, starting from a given current community, i.e., $P(c_i|CC)$ where CC is the current community. This is equivalent to estimating the transition probabilities of a Markov process whose states are the different communities the user can visit. These probabilities can be estimated on line, by monitoring the transitions between communities during the user’s movements. Finally, MFV and F can be implemented by measuring the time spent by the user in each community. In MFV, the weight w_i is equal to $t_i / \sum_i t_i$, where t_i is the time spent by the user in community i . In F, the weights are defined as in MFV, unless for the weight of the current community, which is set to 0.

4.2.2 Estimation of the utility parameters

Gathering context information to compute utility components requires some more steps. According to Equations 2 and 3 to compute utility components for a given data object a node requires the size of the object (s), and estimates of the access probability (p_{ac}) and the availability of that object (p_{av}) in all the individual communities. Theoretically, to achieve the maximum precision of parameter estimation, nodes should advertise all information for all data objects they become aware of, and for all communities they happen to visit. Clearly, this would result in a very detailed computation of utility values, but in a huge networking overhead. Instead, we chose to exploit nodes’ movements to save on network overhead, which is one of the basic principles of opportunistic networks. Basically, when two nodes meet they exchange a summary of data objects in their caches. From these snapshots, each node is able to compute, for each object, a fresh sample of the local availability $\hat{p}_{av,l}$, as the fraction of time during which the object has been seen on neighbours caches. This newly computed value is then used to update $p_{av,l}$ according to a standard smoothed average $p_{av,l} \leftarrow \alpha p_{av,l} + (1 - \alpha)\hat{p}_{av,l}$. For what concerns $p_{ac,l}$, we assume that $p_{ac,l}$ is equal to 1 for those objects the user is interested into, and equal to 0 for the others.⁴

The estimation of the utility parameters for a generic community i , different from the local one, steams from the fact

⁴A more precise estimation of the access probability of the local user to a data object would require a refined model of the user behaviour as far as data access is concerned. However, this is an orthogonal problem with respect to the ContentPlace algorithms, and therefore we choose this simplified representation of users’ access pattern.

that, in ContentPlace, together with the summary vector describing its cache, each node also advertises the set of data objects *its local user* is interested into, and an estimate of the availability of the object for *its* local user, i.e., $p_{av,l}$. Then, every time period T , each node compute a fresh sample for $p_{ac,i}$ and $p_{av,i}$ as the arithmetic mean of the values advertised by the neighbours during T . Then these values are used to update $p_{ac,i}$ and $p_{av,i}$, using the standard smoothed average as seen above. For a more detailed explanation see [4].

Finally, note that the size s of a data object (for which a utility value is required) is easily derived from the summary vectors advertised by the neighbours.

5. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the social-oriented policies described in Section 4. To this aim, we developed a custom simulator that is extended from the one in [5] and uses the same assumptions for lower communication layers. As we discuss in the following, the simulation scenario we consider has been chosen as it is able to highlight *general* features of the social-oriented policies. Note that in [4] we have already presented results showing the impact of system parameters such as the cache size and the number of nodes, which are not evaluated here. The main focus of this analysis is to understand the impact of the different social-oriented policies. We also include in the comparison the two non-social policies, i.e., the greedy and the uniform policies. Uniform has been identified as the best heuristic (from a number of standpoints) in [13]. Furthermore, greedy and uniform have shown to achieve boundary performance result in [4]. Specifically, uniform is the best possible policy in terms of fairness, while greedy the best in terms of hit rate, *in a scenario with a single homogeneous community*.

We hereafter describe the default scenario for our simulations. The default scenario is composed of 45 nodes, divided into 3 communities, moving according to the HCMM model [3] in a 4x4 grid (1000m wide). HCMM is a mobility model inspired by the Watt’s caveman model, that has shown to reproduce statistical figures of real user movement patterns, such as inter-contact times and contact duration [3]. In HCMM, each group represent a social community, and nodes within the same group have social relationships among themselves. Also nodes belonging to different communities can have social relations: according to the rewiring probability (p_r), each link towards a friend is rewired to a node belonging to a different community. Social links in HCMM are used to drive movements: each node moves towards a given community with a probability proportional to the number of links he has towards the community. Thus, the rewiring parameter allows us to control the degree of interactions between nodes of different communities. In our simulations, each group is initially assigned to a cell (its home-cell) avoiding that two groups are physically adjacent (no edge contacts between groups) or in the same cell. This allows to eliminate physical shortcuts between groups, which would bias the evaluation of the ability of policies to bring data objects from one community to the others. Therefore, nodes can exchange data only due to social mobility of nodes and not due to random colocation. The rest of HCMM parameters are as in Table 1. We consider as many channels as the number of groups (n_{gr}). Each group is the source for $1/n_{gr}$ of the objects belonging to each channel, i.e., $1/n_{gr}$ -th

<i>Node Speed</i>	uniform in [1,1.86] m/s
<i>Transmission Range</i>	20m
<i>Sampling period</i>	5s
<i>Cost function parameter (λ)</i>	15
<i>Smoothed average parameter (α)</i>	0.9

Table 1: Configuration Parameters

of the objects of each channel are generated (at the beginning of the simulation) in each group. Note that a data object is *always* available from the node that generated it. To not interfere with the data dissemination performance figures, nodes generating data objects make them available through a separate buffer with respect to the cache. Therefore, for any node, the only way to obtain objects not generated in the local group is to get in touch directly (i.e., the node itself moves in a different group) or indirectly (i.e., one of the nodes of the local community goes out and then comes back, with the desired message in its cache) with an external community. To have an integer number of objects generated in each group, we consider 99 data objects per channel. Each node can subscribe just to one channel. When not otherwise stated, nodes’ interests are distributed according to a Zipf’s law (with parameter 1) within each group. Unless otherwise stated, we consider 3 channels. The popularity of channels is rotated in each group, such that each channel is the most popular in one group, the second-most popular in another group, and the least popular in the last group. Cache space on a node can accommodate exactly all messages belonging to an individual channel. An analysis with varying cache sizes has been presented in [4]. The cache size we choose is a good trade off between seeing the impact of dissemination policies, and avoiding data object disappearance due to low utility on too many caches.

Nodes’ requests for messages follow a Poisson process with parameter $\lambda = 200$ (on average 3 requests every 10 minutes for each node). Nodes can request data only for the channel they subscribe. Within the channel, the data object they request is select according to a uniform distribution. Requests are valid until the simulation ends. As we show in the following, the delay distribution highlights that our simulation length is long enough to reasonably approximate an infinite requests validity timeout. As requests are buffered at issuing node only, and do not occupy cache space, having infinite validity does not impact on the system’s performance. Instead, it allows us to derive a complete analysis of the system’s performance for increasing validity timeouts.

All policies are evaluated in terms of the quality of service (QoS) perceived by the users and the resource consumption. The QoS is measured in terms of hit rate, latency, system utility and fairness level. The hit rate is given by the number of successful requests divided by the number of overall requests. Note that, unless otherwise stated, we show hit rate related to infinite timeout values. As not all policies reach 100% hit rate even in this case, this index allows us to show the fraction of requests that *cannot* be served by the policies. We then separately investigate the hit rate index as a function of the validity timeout. The latency in satisfying the requests has been computed as the difference between the time at which the request is satisfied and the time at which the request was generated. The system utility is computed as the sum of the channel hit rate weighted with the access probability of each channel, i.e., $SU = \sum_i p_{ac,i} hr_i$. The fairness of each policy has been computed according to the traditional Jain’s fairness index (using the hit rate as a measure of the service level obtained

by each channel). Resource consumption has been measured in terms of the traffic generated in the network, i.e., the average number of data transmitted by all nodes during the simulations. This includes data exchanged for context creation, buffer state messages, request messages and data objects themselves. Simulations run for 50000s. Exchanges of data objects upon nodes' contacts start after an initial transitory required by parameter estimators to reach the steady state. Results shown in the following section have a 95% confidence interval, obtained through standard independent replication techniques.

5.1 Analysis in the default scenario

Our first experiment is based on a configuration in which the rewiring probability is the same for all nodes, and equal to 5%. This means that the average number of social links across communities is the same for all the three communities. Although the probability of rewiring is not particularly high, it is already sufficient to mix communities enough to let data objects circulate across all communities, independently on the data dissemination policy. Indeed results (not shown here) show that all policies achieve 100% hit rate.

Based on the results of this set of experiments, we consider a less mixed mobility pattern, as follows. Each pair of communities is connected through just one node. Specifically Community 1 (C1) has two nodes ("travellers") with relationships outside C1, one with Community 2 (C2), the other with Community 3 (C3). Hereafter, we show the performance figures related to nodes in C1. The same remarks can be done for nodes in the other communities, as well. Note that, from Equation 2, it is clear that the channel the traveller nodes are subscribed to (which affects the u_i component) impacts on the data dissemination process. For this reason, we replicate the experiments by varying the channel travellers are subscribed to.

Figure 1(a) show the hit rate for C1, when travellers are subscribed to channel 1. Specifically, the group of boxes related to channel i show the hit rate achieved by nodes whose home community is C1 and are subscribed to channel i . Experiments with travellers subscribed to the other channels provide similar results for all policies but the greedy, that achieves 100% hit rate only for the channel the travellers are subscribed to. Recall that these plots are related to an infinite validity timeout. When shorter timeouts are used, the misbehaviours of Present, MFV and Uniform Social we discuss hereafter become more serious. Also, the performance difference between MLN and Future (which already in this case appear as the best policies) and the other policies increase. We will analyse these aspects in detail in Section 5.2. As a preliminary remark note that, since data objects are always available at generating nodes, a hit rate of 33.33% is always guaranteed. We can identify four different behaviours. With the greedy policy, travellers store only messages to which they are directly interested. This results in a 100% hit rate on the subscribed channel and a 33.33% hit rate on the other channels (there is hit only on the 1/3 of data generated locally). As expected, the greedy policy is able to improve the hit rate of the nodes interested in the same objects the travellers are interested, and is not able to disseminate other objects. With the uniform policy, the hit rate of all channels improves and reaches about 80%. The set of social policies MFV, Present and Uniform Social all show a similar behaviour: they tend to be slightly unfavourable towards the most popular channel. This is due to

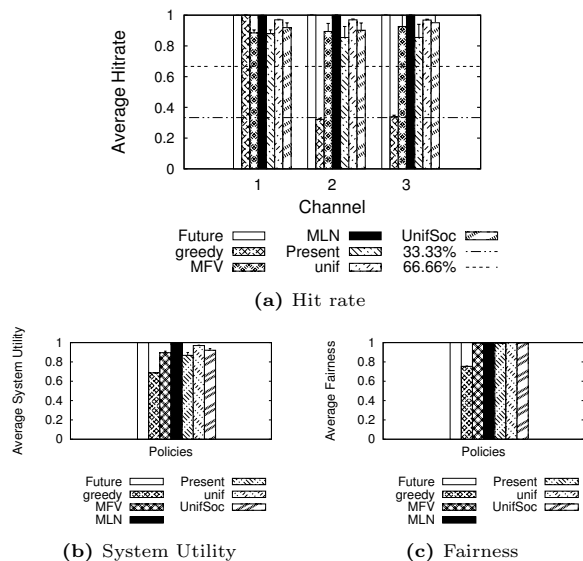


Figure 1: QoS - Travellers subscribed to ch. 1

the fact that nodes within the same group tend to synchronise their behaviour: when nodes realise that certain objects are poorly available, they all fetch them as soon as they become available. This results in the objects becoming highly available, and being dropped simultaneously by all nodes. A detailed analysis of the logs confirms this behaviour. Finally, MLN and Future policies have a very good hit rate in all cases. The key of these policies is that nodes do not consider the community in which they currently roam, but just future communities. It is easy to see that this results in *static* nodes (i.e. nodes without social relationships outside their home community) behaving greedily, and in travellers working to help communities they will visit in the future. This clearly avoids the problems related to the synchronisation effect of the other policies. Results related to the other communities (not shown here) highlight that these policies guarantee the same hit rates shown in Figure 1(a). This is quite important. For example it shows that nodes subscribed to channel 3 in C2 are able to get not only the locally generated objects, and not only the objects carried generated in C1 (directly carried by the traveller of C1 visiting C2), but even those objects generated in C3 that are firstly brought in C1 by the traveller of C1 visiting C3. The identification of these "social" paths is something that is peculiar of the Content-Place MLN and Future policies. We can anticipate that this type of "collective" social behaviour, in which just travellers adopt a social-oriented strategy turns out to be the best solution. The above remarks are confirmed when considering the system utility and fairness indices, as well (Figures 1(b) and 1(c)): Future and MLN are the best policies, although uniform closely approximate their performance. Note that the greedy policy is the only one achieving low fairness. Figures 1(b) and 1(c) are related to the case when the travellers are subscribed to channel 1. Similar results are obtained in the other cases, as well.

Table 2 shows the bandwidth overhead for each protocol. The Uniform policy, although closely approximates MLN and Future in terms of hit rate and fairness, significantly overuses network resources. This is because nodes continuously exchange data objects as a consequence of the tie breaking policy (see, Section 4.1). It is easy to show that this

Bandwidth Overhead [MB]			
<i>Greedy</i>	68.89 ± 0.28	<i>MFV</i>	16497.79 ± 660.57
<i>Future</i>	508.32 ± 21.84	<i>UnifSoc</i>	16749.46 ± 430.16
<i>MLN</i>	509.05 ± 21.61	<i>Unif</i>	71974.40 ± 1316.68
<i>Present</i>	15500.51 ± 881.80		

Table 2: Resource Consumption - Travellers subscribed to ch. 1

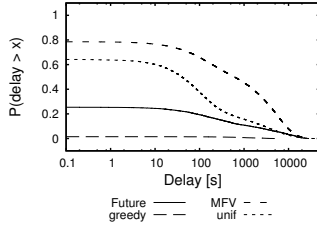


Figure 2: Delay CCDF for requests on channel 1 - Travellers subscribed to ch. 1

is the only way for uniform to make objects circulate. Furthermore, MFV, Present and Uniform Social pay for their synchronisation problem also with regard to the bandwidth overhead. Instead, MLN and Future have a very good performance also with respect to network overhead, while the performance of the Greedy policy is paid in term of hit rate.

Finally, Figure 2 shows the CCDF of the delay for *satisfied* requests on channel 1, when the travellers are subscribed to channel 1. For the sake of readability, we only show the greedy, uniform, Future and MFV policies. The Present and Uniform Social policies are basically equivalent to MFV, while Future and MLN are overlapped. We show this plot only, as plots for the other cases are qualitatively similar. Clearly, greedy achieves the best performance in this case. The other social-oriented policies other than MLN and Future suffer quite high delay. This is a side effect of the unwanted synchronisation issue that we have discussed above. Requests for data objects that have disappeared need long time to be satisfied. MFV and Future clearly outperform uniform also from this standpoint. Note that the maximum delay is in the order of 20000s. This confirms that simulation runs of 50000s are reasonably long to consider the request timeouts as infinite.

In this section we have presented the policies behaviours with respect to all performance figures. In the following, we concentrate on the hit rate index only to highlight the policies different behaviour in different scenarios. The comparison with respect to the other performance figures is qualitatively similar to the one presented in this section.

5.2 Reduced requests' validity timeouts

In this section we analyse the hit rate index as a function of the requests' timeout. Figures 3(a) and 3(b) show the hit rate for increasing requests' timeout. Again, we only present results related to the Greedy, MFV, Uniform and Future policies. Both plots are related to the case when travellers are subscribed to channel 1. Figure 3(a) shows the hit rate for requests on channel 1, while Figure 3(b) shows the hit rate for requests on channel 3. Thus, we consider requests on the most and least popular channels, respectively. Note that the hit rate in the greedy policy do not change with the validity timeout. Recall that requests start after an initial transient in which the policies reach stability. Thus, the greedy policy has already moved all the data objects it is able to move across communities. Uniform and the social policies increase the hit rate when the validity timeout increases. The dynamic is slower with respect to the greedy

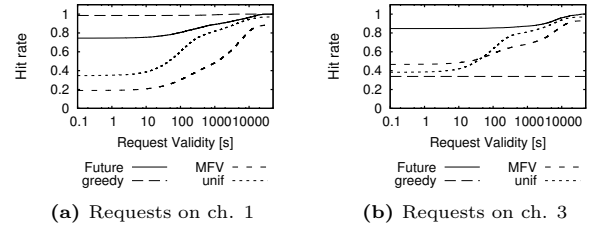


Figure 3: Hit rate with varying timeouts - Travellers subscribed to ch. 1

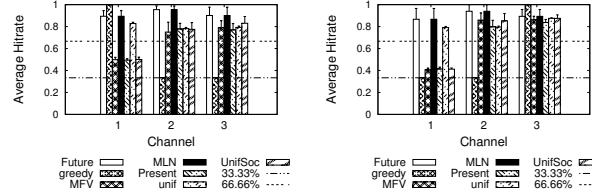


Figure 4: Hit rate with uniform subscriptions

policy, but they are definitely able to provide higher performance to channels that travellers are not subscribed to (see Figure 3(b)). In that case, as expected, the greedy policy is not able to bring any new data objects in addition to those already generated in the community. Finally, again the MLN and Future policies confirm to be the best policies among the one investigated, for all the validity timeouts.

5.3 Uniform Zipf subscriptions

In this section we modify the default scenario by considering a different distribution for subscriptions to channels. Specifically, we continue to use a Zipf distribution with parameter 1, but we do *not* rotate the most popular channels among communities. Channel 1 is always the most popular, channel 2 the second most-popular, channel 3 the least popular. Results have been derived in this case with a limited validity timeout equal to the expected time for a traveller to return in C1, which is about 250s. Figures 4(a) and 4(b) show the average hit rate and the system utility when the travellers are subscribed to channel 1 and 3, respectively, i.e. to the most and least popular channels. The main results we have found in this scenario is that the most popular channel receives the *worst* service for social-oriented policies other than MFN and Future. The channel travellers are subscribed to does not have a significant impact, as is clear by comparing the plots. This is a side effect of the synchronisation effect suffered by these policies. Since data objects of channel 1 are the most requested in *all* communities, they spread aggressively once available, and get easily dropped short afterwards, thus resulting in a high probability of being available just for limited amount of time (unless, of course for data objects generated within each community).

5.4 Large number of travellers

In Section 5.1 we have suggested that policies, like Future and MLN, in which static users are greedy and travellers take care of moving data objects between communities result in the best social-oriented solutions. To further explore this claim, in this section we increase the number of travellers in C1 to 7. All of them are subscribed to channel 1, such that the set of static nodes subscribed to channel 1 in C1 is reduced to one node only. Figure 5 shows the hit rate related to C1. Also in this case, the validity timeout was set to the

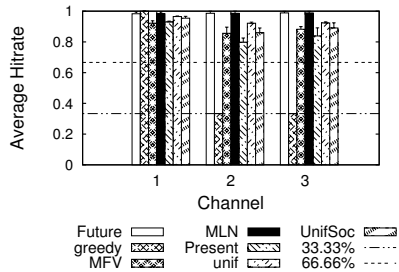


Figure 5: Hit rate - 7 travellers subscribed to ch. 1

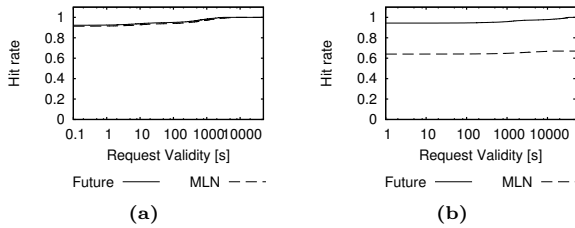


Figure 6: Hit rate for (a) C1 and (b) C3

average time for the travellers to get back in C1. The main result in this scenario is that the hit rate of social-oriented policies other than MFN and Future tends to increase with respect to the case with less travellers. This is not surprising, as more travellers result in a more mixed scenario which (as highlighted at the beginning of Section 5.1), results in higher hit rates, in general. Also note that Future and MLN do not suffer from an increased number of travellers, because in C1 there is anyway enough overall cache space to let data objects of all nodes survive on static nodes.

5.5 Multi-hop social paths

The results presented so far show that in the considered scenarios MLN and Future are the best policies, and perform almost the same. However, there are cases in which they behave differently. Specifically, both MLN and Future fetch data objects by estimating social paths of travellers. Within any community, MLN just takes into consideration the *next* hop only, i.e., the next community it will visit. Future takes into consideration *all* the communities it is likely to visit in the near future, weighted by the probability of visiting each. Therefore, MLN might miss to exploit “multi-hop” social paths across multiple communities. To investigate this effect, we consider a scenario with one traveller only, belonging to C1, subscribed to channel 1. It can visit either C2 or C3 with equal probability when in C1, while always gets back to C1 after having been in C2 or C3. Furthermore, all nodes of C1 (C2, C3) are subscribed to channel 1 (2, 3). In this case, it is expected that MLN is not able to completely serve communities C2 and C3. While in C2 or C3, it will consider only the interests of users in C1, and thus it will only bring back data objects of channel 1. It will bring to C2 (C3) only data objects originated in C1 for channel 2 (3). On the other hand, Future is expected to provide 100% hit rate to all communities. This behaviour is totally confirmed by the simulation results in Figures 6(a) and 6(b), that show the hit rate as a function of the requests’ validity timeout. As expected, both policies are fast in achieving the maximum hit rate. However, MLN can only serve 66.66% of the requests for users in C3, as it can only move the data objects of channel 3 generated in C1. Results similar to those in Figure 6(b) hold for community C2, as well.

6. CONCLUSIONS

In this paper we have evaluated social-oriented policies for ContentPlace, a content dissemination system for opportunistic networks. We have also compared them with uniform and greedy policies, that are considered as valid candidates in the literature. We have identified the social-oriented policies that provide best results in the Most Likely Next and Future policies. Both policies fetch data objects by considering their utility *only* for the communities the user will visit *in the near future*. While MLN considers the most likely community only, Future considers all possible future communities. These policies also significantly outperform the uniform and the greedy policy. Finally, we have also shown cases in which MLN overlooks possible long multi-hop “social paths” across communities, while Future does not.

7. REFERENCES

- [1] A. Balamash and M. Krunz. An overview of web caching replacement algorithms. *IEEE Comm. Surv.*, 6(2):44–56, 2004.
- [2] A. Balasubramanian, B. N. Levine, and A. Venkataramani. Dtn routing as a resource allocation problem. In *Proc. of ACM SIGCOMM*, 2007.
- [3] C. Boldrini, M. Conti, and A. Passarella. Users mobility models for opportunistic networks: the role of physical locations. In *Proc. of IEEE WRECOM*, 2007.
- [4] C. Boldrini, M. Conti, and A. Passarella. Context and resource awareness in opportunistic network data dissemination. In *Proc. of IEEE AOC*, 2008.
- [5] C. Boldrini, M. Conti, and A. Passarella. Exploiting users’ social relations to forward data in opportunistic networks: the HiBO solution. *Pervasive and Mobile Computing (PMC)*, 2008.
- [6] C.-Y. Chow, H. V. Leong, and A. T. Chan. Grococa: group-based peer-to-peer cooperative caching in mobile environment. *IEEE JSAC*, 25(1):179–191, 2007.
- [7] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. *ACM Comput. Commun. Rev.*, 32(4):177–190, 2002.
- [8] P. Hui and J. Crowcroft. How small labels create big improvements. In *Proc. of IEEE ICMAN*, 2007.
- [9] P. Hui, E. Yoneki, S.-Y. Chan, and J. Crowcroft. Distributed community detection in delay tolerant networks. In *Proc. of ACM MobiArch*, 2007.
- [10] A. Jindal and K. Psounis. Contention-aware analysis of routing schemes for mobile opportunistic networks. In *Proc. of ACM MobiOpp*, 2007.
- [11] J. Kangasharju, K. W. Ross, and D. A. Turner. Optimizing file availability in peer-to-peer content distribution. *Proc. of IEEE INFOCOM*, 2007.
- [12] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.
- [13] V. Lenders, G. Karlsson, and M. May. Wireless Ad hoc Podcasting. In *Proc. of IEEE SECON*, 2007.
- [14] L. Pelusi, A. Passarella, and M. Conti. Opportunistic Networking: data forwarding in disconnected mobile ad hoc networks. *IEEE Comm. Magazine*, 44(11), Nov. 2006.
- [15] H. Shen, M. Kumar, S. K. Das, and Z. Wang. Energy-efficient data caching and prefetching for mobile devices based on utility. *MONET*, 10(4):475–486, 2005.
- [16] A. Vahdat and D. Becker. Epidemic Routing for Partially Connected Ad Hoc Networks. *Tech. Rep.*, CS-2000-06, 2000.
- [17] D. Watts. *Small Worlds: The Dynamics of Networks Between Order and Randomness*. Princeton University Press, 1999.
- [18] L. Yin and G. Cao. Supporting cooperative caching in ad hoc networks. *IEEE Trans. Mob. Comp.*, 5(1):77–89, 2006.
- [19] L. Yin, G. Cao, and Y. Cai. A generalized target-driven cache replacement policy for mobile environments. *J. Parallel Distrib. Comput.*, 65(5):583–594, 2005.
- [20] E. Yoneki, P. Hui, S. Chan, and J. Crowcroft. A socio-aware overlay for publish/subscribe communication in delay tolerant networks. In *Proc. of ACM MSWiM*, 2007.
- [21] W. Zhao, M. Ammar, and E. Zegura. Multicasting in delay tolerant networks: semantic models and routing algorithms. In *Proc. of ACM WDTN*, 2005.