Context and resource awareness in opportunistic network data dissemination [*]

Chiara Boldrini, Marco Conti, and Andrea Passarella
CNR-IIT, Via G. Moruzzi, 1 - 56124 Pisa, Italy
{c.boldrini,m.conti,a.passarella}@iit.cnr.it

Abstract

Opportunistic networks are challenging mobile ad hoc networks characterised by frequent disconnections and partitioning. In this paper we focus on data-dissemination services, i.e. cases in which data should be disseminated in the network without a priori knowledge about the set of intended destinations. We propose a general autonomic data-dissemination framework that exploits information about the users' context and social behaviour, to decide how to replicate and replace data on nodes' buffers. Furthermore, our data-dissemination scheme explicitly takes into account resource constraints, by jointly considering the expected utility of data replication and the associated costs. The results we present show that our solution is able to improve data availability, provide fairness among nodes, and reduce the network load with respect to reference proposals available in the literature.

1. Introduction

Opportunistic networks are wireless mobile self-organising networks in which the network topology is assumed to be extremely dynamic and unstable. Disconnections of nodes, high churn rates, dynamic creation and merging of partitions are considered as normal features of the network instead of exceptions, as in the conventional MANET paradigm. Specifically, nodes' movements are exploited to bridge disconnected partitions, allowing end-to-end communication despite prolonged disconnection periods. Research on opportunistic-network protocols has mainly focused on routing issues (see [11, 16] for surveys on this topic). In this paper we focus instead on *data dissemination* protocols, to support applications in which the set of users interested in receiving a given data is not known in advance. Data dissemination protocols are required to support user-generated content applications in pervasive networks, in which users dynamically produce content on their mobile devices, and share it with a dynamic and possibly unknown set of peers interested in the same type of data. This data generation and access model is one of the key features of the Web 2.0 model.

If nodes' and network's resources were unlimited, data dissemination would be trivially achieved by flooding-based schemes such as epidemic routing [14]. However, such solutions become unsuitable as soon as resource consumption is considered. Therefore, the first contribution of this paper is proposing a general framework for *resource-aware* data dissemination schemes. Specifically, as described in detail in Section 3, in our framework data dissemination is driven by the trade-off between the expected *utility* of data for users, and the associated *cost* of in terms of resource consumption. Besides being completely distributed, the proposed framework is general enough to be customisable to any definition of utility, and to consider any set of resources. Providing such a general framework differentiates our work from the PodNet project [9], which just proposes heuristics for data dissemination applications. A utility-based system is also proposed in [1]. However, this paper considers routing protocols instead of data-dissemination services, and assumes knowledge about the overall network status (e.g., the set of nodes where a given data is currently replicated, the statistics of meeting times between all nodes in the network), which might be impractical to obtain. Energy-delay and storage-delay tradeoffs are investigated in [13], but this work focuses again on routing issues, and does not extend the framework to the general resource-consumption problem.

The second contribution of this paper is to customise the framework by proposing an autonomic,*context-based* data-dissemination scheme (see Sections 3 and 4). Exploiting context information about the users has proved to result in particularly efficient forwarding schemes for opportunistic networks [2, 3, 5]. In this paper we exploit context information about the users, their habits, the social communities to which they belong to define the utility of data. Under this approach nodes autonomically learn, without requiring centralised information, the network state and the users' behaviour. Furthermore, it achieves a cooperative data-dissemination scheme, by which users not only look for data they are personally interested to, but also help dis-

[*]This work was partially funded by the IST program of the European Commission under the HAGGLE (027918) FET-SAC project, and under the SOCIALNETS (217141) FET-PERADA project.

seminating data of interest to users they have acquaintance with. To the best of our knowledge, including the social dimension to data-dissemination protocols is an original contribution of this paper.

In Sections 5 and 6 we compare our solution with the best heuristics proposed in [9], showing that our utility-based approach provides a more flexible solution, able to optimise given target performance figures, such as, for example, the hit rate or the fairness among users.

2. Reference scenario

As in [9], we assume a podcasting-like application, in which data are organised in different *feeds* (or *channels*). Data dissemination is based on pair-wise contacts between nodes. During a contact, nodes advertise the channel they are interested into, and exchange an index of the data objects they store in their buffer. The data dissemination policy decides which object each node should fetch from the other peer. In general, data objects might be associated with a TTL value, such that they can be discarded when the TTL expires. We assume that users are organised in communities. Each community represent a social group (e.g., a working environment, a sport team, etc.). We assume that, at any point in time, each user is associated with one "home" community (e.g., the office mates during working hours). Users can also have social relationships outside their home communities, and therefore occasionally "visit" other communities. Accordingly, to represent users behaviour, we consider the home-cell community-based mobility model (HCMM [4]), which is a group-based socially-inspired mobility model. In HCMM user's movements are driven by the attraction exerted by i) other peers on the user (e.g., users move to meet their friends), and ii) the attraction exerted by physical locations on the user (e.g., users move to their office building). Finally, since communities represent homogeneous social group, we assume that the probability that a user is interested in a given channel is the same for all users of the same community.

3. A framework for resource-aware data dissemination

According to the scenario laid down in Section 2, the goal of the data-dissemination protocol is selecting, upon nodes' encounters, which data objects should be fetched. To this end, we exploit a utility-based approach, as follows. Upon a pair-wise contact, nodes exchange an index of the data they are carrying[1]. Each node computes the *utility* of the data objects stored by the other peer (i.e., the *peer's objects*) and of the data objects it currently stores (i.e., the *local objects*). The node then selects those data objects (either locally stored or available at the peer) that maximise its overall utility, subject to the constraints imposed by the resource limitations. Therefore, it selects the data to fetch according to the solution of the following problem:

$$\begin{cases} \max & \sum_k U_k^{(p)} x_k^{(p)} + \sum_k U_k^{(b)} x_k^{(b)} \\ \text{s.t.} & \text{resource constraints are met} \end{cases} \tag{1}$$

where $U_k^{(p)}$ and $U_k^{(b)}$ are the utility of the k-th peer's and local object, respectively, and x_k are the variables of the problem.

By assuming that m resources have to be managed at each node, and by denoting with $c_{jk}, j = 1, \dots, m$ the percentage of consumption of resource j related to the k-th data (i.e., $0 \le c_{jk} \le 1$), the problem statement in Equation 1 can be cast to a Multi-costrained 0-1 Knapsack Problem (MKP) ([8]):

$$\begin{cases} \max & \sum_k U_k x_k & \\ \text{s.t.} & \sum_k c_{jk} x_k \le 1 & j = 1, \dots, m \\ & x_k \in \{0,1\} & \forall k \end{cases} \tag{2}$$

Note that, as the number of constraints is equal to the number of resources, which is not expected to be high, computing the optimal solution or accurate approximations of the optimal solution is fast from a computational standpoint [8]. The formulation in Equation 2 is quite flexible, as it can be used for any set or managed resources, and for any definition of the utility function. In the following of this section, we provide our definition of the utility function, which, by exploiting context information, captures the utility of data objects based on the users' social behaviour.

3.1. Context-aware utility

According to our reference scenario, the utility function should reflect not only the utility for the local users, but also the utility for the other members of the user's communities. Therefore, we define the utility value computed locally by a node for a data object k as:

$$U_k = u_k^{(l)} + \sum_i \omega_i u_{k,i}^{(c)} \tag{3}$$

where $u^{(l)}$ is the utility for the local user, $u_{k,i}^{(c)}$ is the utility for the i-th community the user belongs to, and ω_i is a cooperation index (a real number between 0 and 1), which defines the willingness of the user to cooperate with the i-th community. It is clear that this utility definition permits a social, non-greedy behaviour.

In general, we define the utility of an object for a community (or for the local user) as a function of *context* parameters that describe the community (or the local user) itself. This permits to see $u_k^{(l)}$ and $u_{k,i}^{(c)}$ in a homogeneous way. They can be seen as the utilities related to the i-th context the user is in touch with, considering the local user just as one of these contexts. Therefore, Equation 3 can be

[1]Since in opportunistic networks data are carried in form of *large* bundles, the index is typically much shorter than the data itself.

expressed as

$$U_k = \sum_i \omega_i u_{k,i} \qquad (4)$$

The first step to define the utility function is identifying the parameters it should depend on. Specifically, we use the definition provided by Equation 5. Note that, to simplify the notation, in Equation 5 we remove the explicit indication of the context and the data object the utility refers to (i.e., u in Equation 5 is any of the $u^{(i)}$ factors in Equation 4 for any data object).

$$\begin{aligned} u \quad = \quad & f_u(\text{access probability}, \text{value}, \\ & \text{context stability}, \text{resources' consmuption}) \end{aligned} \quad . \qquad (5)$$

In Equation 5 we identify four main parameters:

- *Access probability*. This parameter should capture the likelihood that the data object will be accessed by users in the context. The highest the access probability, the more the data should be useful for the users in the context.
- *Value*. This parameter quantifies the advantage for the users in the context brought by the fact that the local user stores that particular data object. It can also be seen as the marginal utility for the users of the local node storing the data. We will discuss in more depth this parameter in the following of the section.
- *Context stability*. This parameter captures the stability of the context the utility refers to, and is a sort of reliability index for the previous two parameters (access probability, value). The lower the context stability the lower the utility should be.
- *Resource consumption*. As we will describe in the following, the *value* parameter describes the value of the data object for the users in the context, irrespective of any costs related to storing or fetching it. This parameter permits to trade off the value of the data object and its associated costs.

In the rest of this section we provide a possible concrete realisation of the general form in Equation 5. Let us firstly focus on the contribution to f_u of the *value* parameter. We quantify the value of a data object as a function of its *availability* in the context, and its *residual TTL*. The rationale behind considering the availability is that *rare* data in the context should be preferably stored (thus, should more valuable). This idea is borrowed from common p2p systems (e.g., BitTorrent), which try to preferentially replicate data that are less widespread, since data being already spread are easy to get from the network. The rationale behind considering the residual TTL is the fact that data that are going to become invalid quite soon should not be of high value. By assuming that availability and residual TTL are independent parameters, we define the value of the data object as

$$v = f_v(availability) \cdot g_v(TTL) \ . \qquad (6)$$

We also assume to have two system-wide parameters, v_{min} and v_{max}, that are the minimum and maximum values of any data (i.e., $v_{min} \leq v \leq v_{max}$). While the residual TTL can be easily computed, to quantify the data availability we assume that the local node has an estimate of the probability of users in the context to actually receive the data object, denoted as p_{av} (we will discuss how to obtain such estimates in concrete example in Section 4). Based on the above remarks we can define $f_v()$ and $g_v()$ as follows:

$$\left\{ \begin{array}{lcl} f_v() & = & \sqrt{v_{max}} - \left(\sqrt{v_{max}} - \sqrt{v_{min}}\right) e^{\lambda(p_{av}-1)} \\ g_v() & = & \sqrt{v_{max}} - \left(\sqrt{v_{max}} - \sqrt{v_{min}}\right) e^{-\theta TTL} \end{array} \right. \qquad (7)$$

The definition of f_v and g_v is the simplest one satisfying the following requirements:

1. v should be in the interval $[v_{min}, v_{max}]$
2. f_v should be minimum when p_{av} is maximum (i.e., $p_{av} = 1$), and maximum when p_{av} is minimum (i.e., $p_{av} = 0$).
3. g_v should be minimum when TTL is minimum (i.e., $TTL = 0$), and maximum when TTL is maximum (theoretically, $TTL = \infty$).

Let us now focus on the *access probability* parameter. Theoretically, each user could have a different access probability to any given data object. However, we choose to consider a single *aggregate* parameter for all users sharing a similar context. This relies on the fact that users are in the same context because they share interests, and therefore we can reasonably assume a rather homogeneous access probability across users of the same context. The access probability is throughout referred to as p_{ac}, and is computed as described in Section 4.

As mentioned before, the *context stability* parameter is meant to be a reliability index for the access probability and value indexes. The value and the access probability are the way in which we embed context information derived from the users' social behaviour in the data dissemination policy. Therefore, their contribution should be reduced in case of unstable contexts (e.g., due to high user churn rates, or significant variability of the access probability, etc.). Based on the value and access probability definitions described above, it is reasonable to quantify the context stability through the coefficients of variation of the access and availability probabilities (cv_{pac} and cv_{pav}, respectively). Specifically, we define the context stability index (cs) as follows:

$$cs(cv_{pac}, cv_{pav}) = e^{-h \max(cv_{pac}, cv_{pav})} \ . \qquad (8)$$

The last contribution to the utility function we have to define is related to the *resource consumption*. To quantify this parameter, we need to aggregate consumption related to each data object over all resource. To this end, we define an aggregate consumption c as the weighted sum of the single-resource consumptions, i.e.,

$$c = \frac{\sum_{j=1}^{m} w_j c_j}{\sum_{j=1}^{m} w_j} \ , \tag{9}$$

where w_j is the weight assigned to resource j. Note that this definition also allows us to rank resources based on their relative importance (computing similar aggregates is common when defining heuristics for solving MKPs). Finally, the contribution to the utility function related to the resources' consumption is defined as:

$$f_c() = \frac{1}{e^{\mu c}} \ . \tag{10}$$

Through this definition we comply with the following requirements: i) the consumption function should be equal to 1 when resources' consumption is negligible (this is necessary for the following definition of the utility function), and ii) the usefulness of the data should decrease *more* than linearly with the consumption increase. As far as the latter point, please note that by tuning the μ parameter, it is possible to control the impact of the consumption on the utility function.

Based on the above discussion we are finally in the position of defining a concrete formula for the utility function of a data, related to a particular context. Specifically, we specialise Equation 5 as follows:

$$u = p_{ac} \cdot cs \cdot \frac{v}{e^{\mu c}} \ , \tag{11}$$

where cs is defined as in Equation 8, v is defined as in Equations 6 and 7, and c is the resources' consumption related to the data, as defined in Equation 9. Note that Equation 11 has the same general form of utility functions widely adopted in the literature about cache replacement systems ([12], [15]). Usually, in this field the utility is defined as the product of the access probability by the value of the data. Furthermore, resource consumption is usually considered just in terms of data size. Our definition of the utility function is more general from this standpoint, as it allows for any set of managed resources, and it still valid in the limit case of negligible resource consumptions (in that case, the consumption contribution to u simply disappears).

In conclusion, the main features of the proposed framework we would like to highlight are as follows:

- we explicitly take into consideration *users' social behavior* through the access probability, the context stability and the value parameters;
- we introduce a *reliability index* (the context stability) to take into consideration possible variability of the information describing the users' social behavior;
- we use a *general resource consumption index* (not limiting to the data size only) which permits to take into consideration any set of resources to be managed.

The framework proposed in this section is quite general, and can be specialised for managing a number of different resources. In the following of the paper, we explain how this framework can be implemented by exploiting HiBOp, a context-aware framework for data forwarding and dissemination we have proposed in [2], and we investigate the performance of the framework when applied to manage the nodes' buffer space.

4. HiBOp-D: a context-aware data dissemination protocol

To implement our framework in a real opportunistic network we exploit the algorithms for managing context information defined by HiBOp [2], which is a context-aware routing protocol for opportunistic networks. The resulting protocol is a HiBOp customisation for data-dissemination, that we call HiBOp-D.

Given the utility function of Section 3.1, designing a data dissemination protocol that works according to that utility means i) to define which are the contexts a node operates in, and ii) how the information about these contexts is collected. Regarding the first problem, we need a way of detecting the communities a node belongs to and matching them to the current environment the node is in. The detection of communities is not a trivial task and researchers are still working on it (see [6] for first approach to the problem). Assuming that each node keeps track of the most frequent events of its life, we suppose that, as each node spends the most of its time with nodes of the communities it belongs to, the most frequent events will correspond to events related to these communities. For tracking the most frequent events of nodes' life, we borrow the History table mechanism from HiBOp protocol. In HiBOp, the History table keeps track of the history of the environment in which each node operates, thus enabling some statistics on the history of the node. We assume the information stored in the History table to be relevant with respect to the communities the node belongs to.

We also maintain HiBOp's mechanism for collecting context information. In the HiBOp protocol, nodes periodically send a summary of their personal information (which we called Identity Table) to all other nodes in radio range. Then, the information collected during these Neighbour Discovery phases is inserted into the History table for keeping track of the history of the environment in which each node operates.

As we want to detect the interests of the communities a node belongs to, and the current data distribution (needed for computing availability value), we extend the HiBOp solution adding to the personal information (IT table) exchanged during the Neighbour Discovery phase the following data:

- *Interests*: each node specifies the channels he is interested in;
- *Cache composition*: for each channel, a node specifies the fraction of cache space occupied;
- *Local availability*: the ease with which a node can retrieve information of each channels.

The data transmitted by current neighbours are then processed at the end of each neighbour discovery phase. Specifically, the interests of each peer are used to compute the access probability of the CC (Eq. 12), the cache composition of each peer is used to evaluate the current availability for each channel (Eq. 13), and the local availability is used to compute the ease that your peers have to retrieve information for each channel (Eq. 14):

$$p_{ac,ch} = \sum_{i \in CC} \delta_{i,ch} / N_{CC} \tag{12}$$

$$p_{av,ch} = \sum_{i \in CC} S_{i,ch} / \sum_{i \in CC} S_i \tag{13}$$

$$\widehat{p}_{av,ch} = \sum_{i \in CC} \frac{p^{(l)}_{av,ch,i}}{N_{CC}}, \tag{14}$$

where δ_i^{ch} is equal to 1 if node i is interested to channel ch (otherwise it is equal to 0), S_i is total cache space of node i, S_i^{ch} is the amount of cache space of node i occupied by channel ch. CC values are basically samples that are used to compute historical values. They are added to the History table according to a time-window average:

$$\begin{aligned} p^{(h)}_{ac,ch} &\leftarrow \alpha p^{(h)}_{ac,ch} + (1-\alpha) p_{ac,ch} \\ p^{(l)}_{av,ch} &\leftarrow \alpha p^{(l)}_{av,ch} + (1-\alpha) p_{av,ch} \\ p^{(h)}_{av,ch} &\leftarrow \alpha p^{(h)}_{av,ch} + (1-\alpha) \hat{p}_{av,ch} \end{aligned}$$

In addition, for each parameter, we keep last L values, in order to compute the coefficients of variation.

Summarising, our context-aware utility function for a data object i belonging to channel ch is given by $U_i = u_i^{(local)} + u_i^{(history)}$, where:

$$u_i^{(local)} = \frac{p^{(l)}_{ac,ch} \cdot f_v(p^{(l)}_{av,ch}) \cdot g_v(TTL_i) \cdot cs^{(l)}_{ch}}{e^{\mu c_i}} \tag{15}$$

$$u_i^{(history)} = \frac{p^{(h)}_{ac,ch} \cdot f_v(p^{(h)}_{av,ch}) \cdot g_v(TTL_i) \cdot cs^{(h)}_{ch}}{e^{\mu c_i}} \tag{16}$$

where $p^{(local)}_{ac,ch}$ is equal to 1 for all channels in which the local user is interested, otherwise is set to 0.

Node Speed	uniform in [1,1.86] m/s
Buffer size	10 objects
Transmission Range	20m
Sampling period	5s
λ	15

Table 1: Configuration Parameters

5. Simulation Setup

To evaluate our utility-based data-dissemination framework, we consider a very simple scenario. We consider just a single community, and focus on intracommunity dynamics only. Nodes move according to the Random Waypoint (RWP) mobility model, following to the procedure described in [10] for RWP convergence. While, in general, social based mobility models (such as HCMM) fits better our framework, HCMM and RWP are equivalent when just a single community is considered [4]. The evaluation with more than one community is left as a future work. The single-community scenario is a worst-case scenario for our data-dissemination framework, as the advantage of exploiting information about users' social behaviour is likely to be much more evident in the case of several groups of users, each with different interests. Therefore, this setup allows us to evaluate the framework in a rather limiting, unfavourable, case.

In our setup data objects can belong to three different channels. Each node is interested into one channel. Interests are distributed among nodes according to the Zipf's law. The nodes move, with a typical pedestrian speed, in a 250x250m cell. In the middle of this cell, an Access Point (AP) stores 100 data objects for each channel. These objects are generated before the simulation begins and remain the same for the whole duration of the experiment ($TTL = \infty$). All the data objects have the same size, equal to 50 KB. Nodes' buffer size is set to 10 objects only, so as to test the framework in a resource constrained environment as far as buffer limitations. Furthermore, we consider only the buffer as the constrained resources to be managed according to the scheme in Equation 2. Reference values for other configuration parameters are summarised in Table 1. Note that, since all data objects have the same size, all objects of the same channel have the same utility for a given node, and thus the utility computed by Equation 11 is the utility of the channel.

The performance of the following policies for choosing which data objects to store upon a pair-wise contact is compared:

Context-aware (CA) Objects are selected based on the context-aware utility function explained in Sections 3 and 4.

Access Probability (aP) This policy can be viewed as a special case of our utility function in Equation 11, when only the access probability is considered. Nodes tend to preferentially select data objects of the channel the local user is interested.

Uniform (U) Objects are selected randomly among all available objects in the local and peer's buffers (regardless the channel they belong to). This policy has been proved to be one of the best in [9].

Uniform mono-channel (UM) Nodes select a channel at random, and store all objects of that channel. We consider this version of the uniform policy as we found that nodes under the aP and CA policies tend to store objects of a unique channel only.

The performance of these policies is 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, system utility and fairness level. To measure the hit rate, we simulate a user periodically requesting (with a period equal to 300s) a data objects for each channel (the object of a given channel is randomly selected according to a uniform distribution). We consider a hit if the object is available on at least one node's buffer, a miss otherwise. 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 Jain's fairness index [7] (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 90000s. Exchanges of data objects upon nodes' contacts start after an initial transitory required to build nodes' context. Results shown in the following section have a 95% confidence interval, obtained through the independent replication technique.

6. Simulation Results

6.1 Impact of network size

In this first set of experiments we evaluate the performance of the policies under a variable number of nodes, ranging from 8 to 72.

The Access Probability policy (Figure 1) serves channels in a way proportional to the probability that they are requested by nodes: channel 1 receives the best service,then channel 2, and finally channel 3. On the opposite, the Uniform policy (Figure 3) guarantees to all channels an equal level of service in terms of hit rate. Between these two extremes lies our context-aware caching policy (Figure 2), which maintains the same ordering as the Access Probability policy but reduces the gap between channels with different popularity. Also note that there is basically no difference, in our setup, between the U and UM policies. Therefore, exchanging single data objects belonging to different channels or a bulk of objects of the same channel achieve the same results when the selection is done in a uniform fashion. Figures 1- 4 also highlights that no policy is able to reach a 100% hit rate, not even when the sum of nodes' buffers would be enough for storing all the data objects for all channels (this is the case for nodes more than 30). This is because no policy is able to control the replication level of data objects across the network. This results, for some objects, in more than one copy being present on nodes' buffers, while, for other, not even a copy being available. Improving the policies to achieve higher diversity of stored objects is one of the most required extensions of our framework.

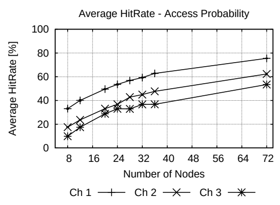

Figure 1: Average hit rate (aP)

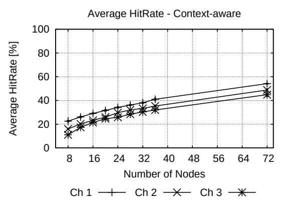

Figure 2: Average hit rate (CA)

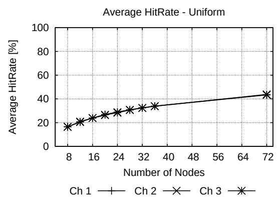

Figure 3: Average hit rate (U)

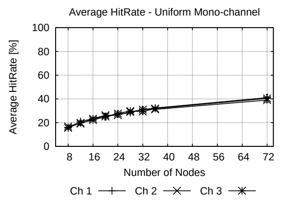

Figure 4: Average hit rate (UM)

Figure 5 shows the system utility index. Being SU a measure of user satisfaction, it is quite expected that the aP policy, by favouring the most popular channel, satisfies the majority of the users. However, Figure 6 shows that this is paid in terms of fairness. On the other hand, uniform policies achieve, by definition, the highest fairness, at the cost of lowest system utility. Therefore, a better trade off between the maximisation of the overall utility and fairness should be met. Figure 5 highlights that this trade off can be satisfactorily reached by the CA policy. Its fairness level approaches that of the uniform policies while still maintaining an higher level of user satisfaction (between 10% and 20% more).

Beside being a good trade-off between user satisfaction and fairness, the utility-based framework shows a resource-saving advantage as well. Figure 7 shows the traffic overhead for all caching policies. The most efficient policy is aP because, once node interests have spread all over the network, values for access probabilities don't change anymore during the simulation. This results in fewer exchanges of

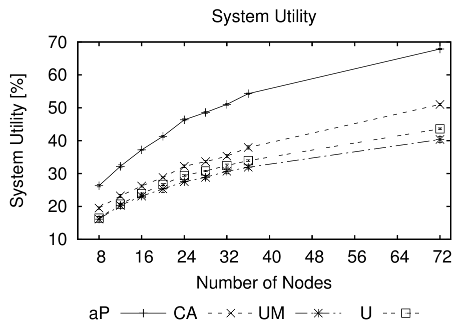

Figure 5: System Utility

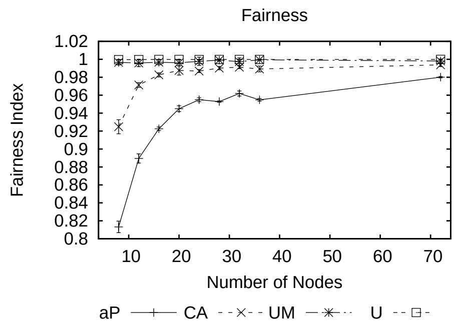

Figure 6: Fairness

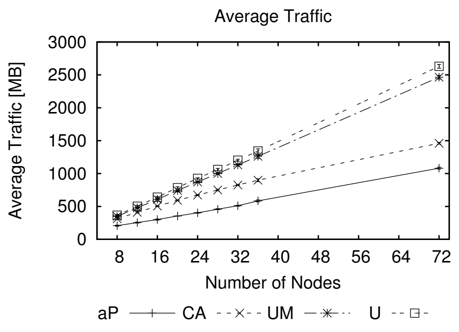

Figure 7: Traffic overhead

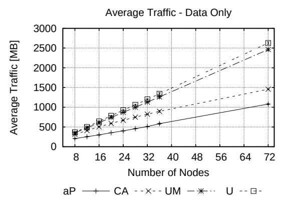

Figure 8: Data overhead

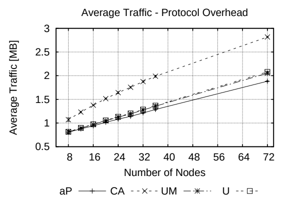

Figure 9: Protocol overhead

data objects (basically, only nodes sharing the same interest exchange objects). In the CA policy, the utility ranking changes dynamically based on the availability measure, thus resulting in a slightly higher traffic overhead. The uniform policies waste more resources because they don't take into account the state of the network, and thus objects are exchanged at every association between nodes. We have also made a separate analysis for traffic due to data object exchanges and traffic due to protocol overhead (context information, message requests, cache state messages). It's interesting to note that total traffic overhead is slightly dependent on the overhead due to protocol operations [8, 9]. Therefore, data exchanged for building node context are negligible. This result is particularly significant to our context-aware caching policy, which was expected to generate more protocol overhead than the others. This expectation has been confirmed, but its relevance has been drastically reduced.

6.2 Adding off-line prefetching

In the previous experiment, nodes' buffers were empty at the beginning of the simulation. In this second set of experiment we want to evaluate the impact of preloaded buffers on the evolution of the system. We consider to different filling policies: uniform filling, in which nodes' buffers are filled at random at the beginning of the simulation, and maximum diversity filling, in which nodes' buffers are filled as to maximize the diversity of data objects (i.e. avoid replication). We fix the number of nodes to 16. Figures 10- 13 show that, preloading data on nodes at the beginning of the simulation doesn't affect data dissemination performance. However, this is just a preliminary result, which should be confirmed with further analysis.

6.3 Turning the AP off

The AP is fundamental to the regeneration of data objects in the network: in fact, even when all nodes have deleted a given object, the AP guarantees that this object can be injected into the network again. In order to evaluate how the presence of the AP affects the behaviour of the system, we fill nodes' buffers at the beginning of each simulation, and "turn off" the AP. We explore two different policies for initial filling: uniform filling, where data to be copied on nodes' buffers are selected uniformly among all available objects, and maximum diversity filling, where objects are copied on nodes as to minimise replications. We consider the case of 16 nodes, in which not all objects can fit into nodes's buffers to highlight the importance of the AP role in this challenged setting.

Table 2 shows that when the AP is off a certain level of service is maintained, though highly downgraded. This shows that, when the buffer size is very small, the AP is key to provide a reasonable service level. These results also show that, in these settings, the different initial filling policies seem to have no, or very little effect, on system performance for all caching policies. While still preliminary, this set of results suggests interesting properties of the system related to the AP role. We are analysing the system in more depth to achieve better understanding on this point.

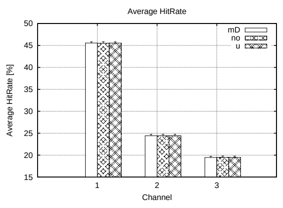

Figure 10: Average hit rate (aP)

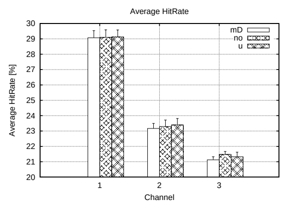

Figure 11: Average hit rate (CA)

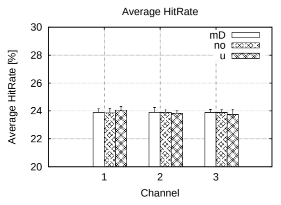

Figure 12: Average hit rate (U)

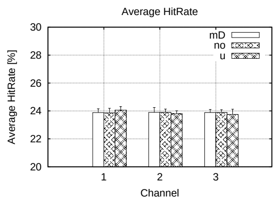

Figure 13: Average hit rate (UM)

	uniform		max div	
	AP on	AP off	AP on	AP off
aP	35.38 ± 0.16	9.33 ± 0.32	35.38 ± 0.14	9.43 ± 0.37
CA	26.23 ± 0.30	6.09 ± 0.39	26.10 ± 0.26	6.06 ± 0.44
U	23.93 ± 0.17	5.45 ± 0.23	23.88 ± 0.20	5.65 ± 0.25
UM	22.92 ± 0.39	4.2 ± 1.06	22.93 ± 0.35	6.34 ± 1.34

Table 2: System utility with and without the AP.

6.4 Dense scenario

The last set of results we show highlight a mis-behaviour of the CA policy that arises in denser scenarios without AP, and a simple modification to fix it. Specifically, we consider a scenario with 16 nodes, but with transmission range equal to 80m (instead of 20m). This results in an increase of the average number of neighbours from 0.3 to about 5. Therefore, in this setting basically each node quickly achieves a complete view of what is available in the buffers of all the other nodes. Qualitatively, since the CA policy takes into consideration availability to compute utility, the least spread channels at the beginning of the simulation become the most useful. All nodes *immediately* discard objects of the other channels, which cannot be later resumed, due to the absence of the AP. In this case, the initial configuration of the simulation environment plays a key role in determining which channel is going to "survive". Table 3 shows the percentage of total buffer space (considering all the nodes' buffers as a single shared buffer) occupied by each channel for the utility-based policies (CA and aP only). CA results refer to different simulation runs. Note that aP does not suffer from this problem as it does not take into account the dynamically changing state of the nodes' buffers.

A simple fix to this problem for the CA policy is as follows. When nodes meet, they compute the utility of the channel as usual, but then they use the utility values to define the *share* of local buffer to be occupied by each channel's objects. The last row in Table 3 shows that this simple modification is able to fix the CA misbehaviour. We are currently investigating the applicability of this modified CA policy in more general settings.

	ch1	ch2	ch3
aP	56.04 ± 0.00	24.81 ± 0.00	18.54 ± 0.00
CA (orig)	1.79 ± 1.23	0.00 ± 0.00	92.44 ± 1.69
	0.61 ± 0.60	0.56 ± 0.62	98.10 ± 1.54
	89.66 ± 0.66	9.92 ± 0.10	0.00 ± 0.00
	0.00 ± 0.00	99.25 ± 1.04	0.00 ± 0.00
	96.62 ± 1.59	0.00 ± 0.00	2.77 ± 1.38
CA (fix)	42.04 ± 8.59	34.33 ± 7.46	22.95 ± 15.94

Table 3: Percentage of nodes' buffer occupied by each channel.

7 Conclusions

In this paper we have introduced a new autonomic utility-based data-dissemination framework for opportunistic networks which heavily exploit context information about the users' social behaviour. We have compared the performance of the framework with other solutions considered among the best in the literature. Results have shown that our approach is flexible enough to achieve different targets, and specifically can be customised either to achieve the maximum hit rate for most popular data, or to achieve a fairer behaviour with acceptable reduction of the hit rate. We have also shown preliminary yet interesting results showing that, in particularly challenged scenarios, network elements, such as Access Points, that can store *all* available data, are key. We have also highlighted some possible unstable behaviour of our framework, and shown a simple modification to fix it. The presented results are still quite preliminary, however they highlight that a utility-based framework is an interesting direction to be further investigated to provide data dissemination services in opportunistic networks.

References

[1] A. Balasubramanian, B. N. Levine, and A. Venkataramani. Dtn routing as a resource allocation problem. *Proc. of SIGCOMM*, 2007.

[2] C. Boldrini, M. Conti, I. Iacopini, and A. Passarella. HiBOp: a History Based Routing Protocol for Opportunistic Networks. In *Proc. of IEEE WoWMoM*, June 2007.

[3] C. Boldrini, M. Conti, and A. Passarella. Impact of social mobility on routing protocols in opportunistic networks. In *Proc. of the IEEE AOC*, 2007.

[4] C. Boldrini, M. Conti, and A. Passarella. Users mobility models for opportunistic networks: the role of physical locations. In *Proc. of IEEE WRECOM*, 2007.

[5] P. Hui and J. Crowcroft. How small labels create big improvements. In *Proc. of IEEE PERCOMW*, 2007.

[6] P. Hui, E. Yoneki, S. Chan, and J. Crowcroft. Distributed community detection in delay tolerant networks. *Proc. of ACM MobiArch*, 2007.

[7] R. Jain, D. Chiu, and W. Hawe. A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. *Arxiv preprint cs.NI/9809099*, 1998.

[8] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[9] V. Lenders, G. Karlsson, and M. May. Wireless Ad hoc Podcasting. In *Proc. of IEEE SECON*, 2007.

[10] W. Navidi and T. Camp. Stationary distributions for the random waypoint mobility model. *IEEE Transactions on Mobile Computing*, 3(1):99–108, 2004.

[11] L. Pelusi, A. Passarella, and M. Conti. Opportunistic Networking: data forwarding in disconnected mobile ad hoc networks. *IEEE Communications Magazine*, 44(11), Nov. 2006.

[12] J. Shim, P. Scheuermann, and R. Vingralek. Proxy cache algorithms: design, implementation, and performance. *IEEE Transactions on Knowledge and Data Engineering*, 11(4):549–562, 1999.

[13] T. Small and Z. J. Haas. Resource and performance tradeoffs in delay-tolerant wireless networks. *Proc. of ACM WDTN*, 2005.

[14] A. Vahdat and D. Becker. Epidemic Routing for Partially Connected Ad Hoc Networks. Technical Report CS-2000-06, CS. Dept. Duke Univ., 2000.

[15] L. Yin, G. Cao, and Y. Cai. A generalized target-driven cache replacement policy for mobile environments. *Applications and the Internet, 2003. Proceedings. 2003 Symposium on*, pages 14–21, 2003.

[16] Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges. *IEEE Communications Surveys & Tutorials*, 8(1):24–37, 2006.