Content Discovery in Mobile Networks Using the Publish-and-Subscribe Paradigm

Francesco Malandrino
DELEN – Politecnico di Torino
Torino, Italy
Email: malandrino@tlc.polito.it

Claudio Casetti
DELEN – Politecnico di Torino
Torino, Italy
Email: casetti@tlc.polito.it

Carla-Fabiana Chiasserini
DELEN – Politecnico di Torino
Torino, Italy
Email: chiasserini@tlc.polito.it

***Abstract*—We present Figaro, a content discovery solution for mobile ad hoc networks. Our main focus is on urban environments, featuring high densities of users in relatively narrow, circumscribed areas where wireless connectivity is provided by an infrastructure (e.g., bus stops integrating a wireless access point). Figaro superimposes an overlay network onto the mobile ad hoc network, exploits the the well-known publish-and-subscribe paradigm for content discovery and resorts to peer-to-peer communication for content delivery. In our work, we study the performance of Figaro and address important issues in content sharing systems, such as the identification and isolation of misbehaving nodes and the cooperative caching of rare contents.**

I. Introduction

It is foreseeable that one of the most appealing applications for mobile ad hoc networks (MANETs), as soon as they reach the expected maturity and the number of users attains critical mass, will be those that support content sharing among users. Much like in today's wired counterpart, the ubiquity and ease-of-access of future wireless networks will encourage users to share the content they own and look for contents that match their interests. The peer-to-peer model seems well suited to the MANET environment, where fleeting connectivity and limited infrastructure support give user interaction preeminence over server access-based applications. In this context, it is of paramount importance that the content carried by users is "discoverable" with ease and that its carriers are reliable when it comes to providing the content to others.

In this work we present Figaro, a content discovery solution for mobile environments. Our main focus is on urban networks, in which high densities of pedestrian and vehicular users coexist in relatively narrow, circumscribed areas reached by an infrastructure (e.g., bus stops integrating an access point). Figaro builds on the well-known publish-and-subscribe (pub/sub) paradigm and aims at defining a network architecture that provides high performance in terms of reliability, content availability and robustness to adverse conditions, such as the presence of misbehaving nodes or particularly rare contents.

More specifically, Figaro includes (i) mobile user nodes, named Agents, which may request contents of their interest and make information and services available to other users, and (ii) infrastructure nodes, named Brokers, which assist Agents in the content discovery process. In this scenario, we first enhance the basic pub/sub scheme by introducing a Proxy entity that exploits the benefits of a backbone connecting the Brokers. Secondly, we define a feedback and reputation mechanism that counteracts free riders, i.e., Agents that do not provide advertised contents. Finally, we consider that Agents have caching capabilities and devise a cache management policy to improve the performance of the content sharing system in presence of rare information items. We define a scheme in which (i) Brokers, using their knowledge on the contents advertised by Agents, ask them to cache rare information items so as to increase the average hit ratio (global objective), and (ii) incentives and penalties make Brokers' requests match the Agents' goal to maximize their hit ratio and minimize their load (individual objective).

The remainder of the paper is organized as follows. Previous work is discussed in Section II. We detail the pub/sub system and its enhancements in Section III. Section IV introduces the reference scenario and assumptions used for performance evaluation, while results derived through ns3 emulations are presented in Section V. At last, Section VI concludes the paper and highlights future work.

II. Related work

There are many works focusing on the use of pub/sub in mobile networks, but most of them refer to infrastructure-less scenarios. As an example, [5] presents an algorithm, based on Voronoi regions, to split the nodes into colonies and elect Brokers inside them, while [6] describes a pub/sub protocol which exploits social information to optimize routing. The presence of an infrastructure and the opportunities it implies are taken into account (among others) in [1], which deals with infrastructure-powered bus stops and buses integrating a (mobile) Broker. With respect to [1], our work includes some novel protocol entities and addresses additional concerns, such as security, reputation and caching issues.

Several works deal with the specific problem of reputation management. For example, [7] presents a way to exploit ad-hoc networking to reduce the load over a 3G cellular network, focusing on incentives and penalties for collaborating and misbehaving users; [8] devises a reputation-based mechanism to allow a reliable deployment of large-scale ad-hoc networks. Thanks to the presence of the Broker, our approach is simpler than the proposed ones, and more suitable for devices with

low computational capabilities (e.g., we do not require, as [9] does, to locally optimize an objective function). Additionally, we do not assume, as [10], [8] do, that Agents are associated to a billing account; indeed, in Figaro incentives and penalties are circumscribed to Figaro itself.

As far as caching is concerned, most works focus on decentralized, infrastructure-less scenarios. In particular, [11] leverages the estimated information density in the network, the work in [12] follows a data-centric approach to proactive data dissemination, and [13] exploits the system's knowledge of the user's social behavior. As a result, the presented solutions tend to be more complex than what is proposed in Figaro.

III. The Figaro system

Figaro networks are overlay networks that operate according to the pub/sub paradigm, featuring two main types of nodes [1]: Agents and Brokers. Agents produce and consume the content, while it is the Brokers' task to let demands and offers meet. Agents are mobile (possibly hand-held) devices, while Brokers are middle-end devices, integrated in the infrastructure and interconnected via a reliable (typically wired) backbone.

When they discover a Broker, Agents can register with it declaring the services or contents (files, items of information...) they are willing to share with other Agents. The Broker maintains a content-based routing table, where it stores the identifier of the registered Agents, their IP address, as well as the contents and services that they make available to others. Once the registration is complete, Agents can ask the Broker for the services they need. The Broker queries the content-based routing table to answer such requests.

The set of Agents exchanging services through a Broker is called *colony*. In a pure overlay fashion, colonies do not necessarily correspond to a set of nearby Agents. Rather, a colony is a logical aggregation of Agents with the purpose of enhancing the performance of the Figaro system.

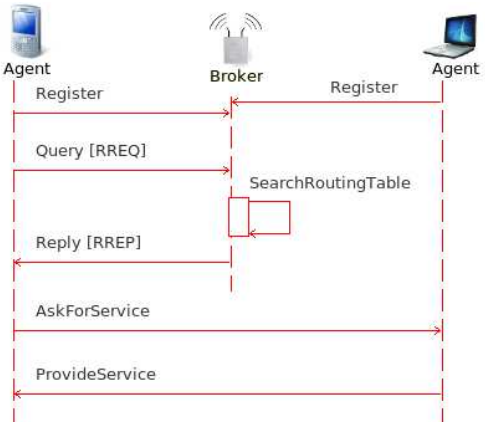

Figure 1. UML sequence diagram for the interaction between Agents and Brokers in the basic version of the publish/subscribe protocol

The diagram in Figure 1 refers to the basic version of the publish-and-subscribe message exchange that we consider. Note that, after registering with a Broker, an Agent can issue content queries to the Broker, which answers them using its content-based routing table. All the packets sent by the Broker are digitally signed. This simple security scheme, well suited for low-power devices, provides integrity and authenticity. If the requested content is stored by a member of the colony, the Broker sends back a Reply message including the IP address of the Agent that can provide the desired information; otherwise the Reply includes a negative response. Once the requesting Agent receives the IP address of the possible provider, a peer-to-peer exchange is established between the two Agents for the content transfer. Such a peer-to-peer transfer is implemented through (possibly) a multihop path that can be found and set up using any of the routing protocols specifically designed for MANETs.

Starting from this basic scheme, we devise several protocol and architecture improvements in order to enhance performance and reliability, or to exploit additional capabilities of the involved entities. In particular, we consider that

- a backbone connecting Brokers can be used to find additional/missing services;
- Agents and Brokers can cooperate to identify and isolate free riders;
- caching capabilities of the Agents can be exploited to make the retrieval of rare contents more efficient.

Each of these enhancements is discussed below.

A. Using a Proxy

When Brokers are connected through a backbone, the information exchange between Brokers can be effectively exploited to improve the system performance. To this end, we introduce a new protocol entity, called Proxy, which is connected to all Brokers via the backbone. When a Broker receives a request for a service or content which is unavailable in its colony, it forwards the request to the Proxy, which in turn asks the other Brokers. In this way, the request is successful if the service is found in *any* of the colonies composing Figaro. Introducing a Proxy does not impact on the basic behavior of the colony: queries that can be solved within the colony to which the requesting user belongs, are handled exactly as explained before.

Intuitively, we expect that such an enhancement significantly increases the probability that an Agent finds and retrieves the desired content (i.e., the hit ratio).

B. Counteracting free riders

With the aim to identify and isolate Agents that do not actually provide the services they claim to provide, the so-called free riders, Figaro includes a reputation mechanism: Agents can report to the Broker the outcome of both successful and unsuccessful peer-to-peer exchanges with Agents identified as content carriers. The exchange outcome is notified to the Broker through a *feedback* message, which the requesting Agent sends to the Broker after the peer-to-peer transfer shown in Figure 1 ends. Based on the received feedbacks, the Broker can therefore assign a reputation score to each Agent and

take it into account in its routing decisions. Notice that such reputation score is not advertised – and actually the Agent itself may not be aware of it.

The Broker computes the reputation score as follows. Let us denote the set of active Agents (i.e., members of the colony) at time t by $\mathcal{A}(t)$, and its cardinality by $A(t) = |\mathcal{A}(t)|$. Additionally, $F_P(i,\tau)$ and $F_N(i,\tau)$ are counting processes identifying, respectively, the number of positive and negative feedbacks assigned to Agent i from time 0 to time τ. The validity of feedbacks is limited to w seconds, i.e., only the feedbacks received within the sliding window $W(t) = [\max\{0, t-w\}, t]$ are used to compute the reputation value at time t.

Thus, at the generic time instant t, each Agent i has $P(i,t) = F_P(i, \sup W(t)) - F_P(i, \inf W(t))$ positive valid feedbacks and $N(i,t) = F_N(i, \sup W(t)) - F_N(i, \inf W(t))$ negative valid feedbacks.

We define a significance threshold, $T_S(t)$, as the minimum number of feedback reports necessary to assign a nonzero feedback value. The goal of such a threshold is to prevent an unfortunate combination of events (or an attack) from banning a "good" Agent. $T_S(t)$ is linked to the average number of feedbacks received by each node during a time window, according to the following expression:

$$T_S(t) = \frac{\sum_{i=0}^{A(t)} [P(i,t) + N(i,t)]}{2A(t)} \quad (1)$$

The rationale behind (1) is that, in order to be significant, the number of feedbacks received by an Agent should be comparable to the average number of feedbacks received by all other Agents in the same colony. Then, the reputation value of the generic Agent i, at a given time instant t, is given by:

$$r(i,t) = \begin{cases} \frac{P(i,t)-N(i,t)}{P(i,t)+N(i,t)} & \text{if } P(i,t) + N(i,t) \geq T_S(t) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Nonzero reputation values are assigned only to those Agents for which sufficient evidence of bad (or good) behavior exists, i.e. having a number of feedback reports greater than T_S; note that newly arrived Agents always have a zero reputation value.

The reputation formula itself is relatively simple, however it has several advantages over equivalent simpler formulations (such as $\frac{N(i,t)}{P(i,t)+N(i,t)}$):

- it is bounded between -1 and 1;
- the limit values 1 and -1 correspond to the limit cases in which an Agent has only positive (or negative) feedbacks;
- it has 0 as both a default value *and* a watershed value.

The latter also means that the value assigned to newcomers is different from the typical values of both "good" and "bad" known Agents (which is not the case of simpler bad-boy indices such as $\frac{N(i,t)}{P(i,t)+N(i,t)}$, which assume newcomers to be good until evidence to the contrary). Additionally, the reputation value assigned to newcomers is exactly the border value between "mostly good" (more positive than negative feedbacks) and "mostly bad" (more negative than positive feedbacks) Agents.

A Broker has evidence of an Agent being a free-rider when $r(i,t)$ falls below a reputation threshold T_R. When this happens, the Agent is banished from the colony: in the routing table the entries related to that Agent are marked as invalid, and any traffic it sends to the Broker is ignored. Banned Agents do not receive any feedback so, due to the sliding window mechanism and the significance threshold $T_S(t)$, their reputation will return to 0 after (at most) the duration w of the window, i.e., after a sufficient number of its negative feedbacks have been forgotten.

Summarizing, the reputation mechanism works as follows. All Agents start with $r(i,0) = 0$; whenever a feedback about Agent i is received by the Broker:

- the significance threshold $T_S(t)$ is updated according to (1);
- if the number of feedbacks related to Agent i received in the last w seconds is higher than or equal to T_S, the reputation score $r(i,t)$ is updated according to (2), otherwise $r(i,t)$ is set to zero;
- the feedback scores of all Agents are updated according to the new $W(t)$ (possibly allowing banned Agents to be re-admitted).

Notice that the reputation scores assigned to the same Agent in different colonies are independent.

At last, it is worth discussing the value of the reputation threshold T_R. This value is application-dependent and should be chosen according to the following guidelines: (i) since all Agents start with $r(i,0) = 0$, T_R must be strictly less than zero; (ii) too low a T_R may prevent the system from detecting free-rider nodes, i.e. cause false negatives; (iii) too high a T_R may trigger the system into banning good nodes that fail to provide a service without fault (e.g., due to connectivity problems), thus causing false positives. However, in Section V we show that, when the time window duration w is sufficiently large, any value of T_R in the interval [-0.1,-0.9] enables the Broker to clearly identify good and bad Agents.

C. Exploiting caching capabilities

We further extend Figaro taking into account the case in which Agents have caching capabilities. In other words, some of the Agents that obtain a service can decide to retain it in their cache (which has, of course, a limited size). Subsequently, those Agents will be able to provide the cached service to other Agents.

This scheme is especially effective when some of the existing services are significantly rarer (i.e., provided by a lower number of Agents) than others. The goal is not only to increase the availability (i.e., the hit ratio) for rare services, but also to curb backbone usage – especially if using the backbone is slow and/or expensive.

We assume that a service j is rare if the number $\Gamma(j)$ of Agents providing it is less than the half of the average number $\overline{\Gamma}$ of Agents providing each service. The Broker is in a good position to identify rare services and, by exploiting the Reply

message, to ask the Agents that are retrieving them to add these services to their cache. The Agent registers Broker's suggestions and decides whether to actually cache the service. It uses a special field of the feedback packet to inform the Broker of its caching decision.

We assume the Agents to have a selfish, rational behavior. Therefore, they will not follow the Broker's caching suggestions unless they can take some advantage from it. In Figaro, as a reward for the Agents that follow the Broker's caching suggestion, the Broker tries not to select collaborating Agents[1] to provide common services. In other words, collaborating Agents end up providing only rare services. In this way, collaborating Agents will experience a lower load: collaborating thus becomes a rational way to pursue a selfish goal.

The scheme described so far does not provide any warranty that the Agents pretending to have cached a service (i) actually have cached it, (ii) actually provide it to other Agents. Some Agents, unwilling or unable to cache a service, may set the caching flag in the feedback packets they send with the purpose of enjoying a lower load.

To cope with this issue, the negative feedbacks related to a service which should have been cached, i.e., revealing a misbehaving Agent, result in a particularly severe penalty. Similarly to III-B, we define as $F_M(i,\tau)$ the counting process of misbehaving feedbacks; thus, $M(i,t) = F_M(i, \sup W(t)) - F_M(i, \inf W(t))$ is the number of misbehaving feedbacks given to Agent i within the time window $W(t)$. Equation (3) is modified as follows:

$$r(i,t) = \begin{cases} \frac{P(i,t)-N(i,t)-\alpha M(i,t)}{P(i,t)+N(i,t)+M(i,t)} \\ \qquad\qquad \text{if } P(i,t)+N(i,t)+M(i,t) \geq T_S(t) \\ 0 \qquad\qquad \text{otherwise} \end{cases} \tag{3}$$

with $\alpha > 1$ to account for the fact that misbehaving feedbacks must have more severe consequences than negative feedbacks. Notice that this modified version of the reputation index can fall below -1. Anyway, it is still guaranteed to return to 0 in (at most) a time w.

Summarizing, the following caching policies are foreseen: (i) *Collaborating*: Agents store the content in their cache, and inform the Broker via the feedback packet. Then, they provide the cached content to other Agents when needed. As a consequence, the Broker lightens the load on collaborating Agents, not using them to provide common services. (ii) *Non-collaborating*: Agents cannot, or do not want to, cache the content. They fill in the feedback packet accordingly. No further action is taken by the Broker (i.e., these Agents will receive nor benefit neither punishment for their behavior). (iii) *Misbehaving*: Agents declare that they will cache a content, but do not do so. Since this behavior disrupts the balance of caching content that the Broker is trying to establish, it assigns misbehaving Agents a very low reputation score, possibly leading to their ban from the colony.

[1]More exactly, the Agents that have collaborated within a given time frame. This prevents Agents from collaborating once and exploiting the reduced load forever. Additionally, the duration of the time frame can be adjusted to fit the application needs.

IV. Network Scenario and assumptions

We use ns-3 [2] emulation to study the behavior of the system in a pedestrian scenario. Four points of interest (POI) are placed at the edges of a 1000×850 m rectangular area. In correspondence to each POI there is a (fixed) Broker, integrating a wireless Access Point (AP). Mobile Agents, whose number is a variable system parameter, roam between the POIs, following the random-trip mobility model [3], with an average pause time of 300 s and an average speed of 1.8 m/s.

Agents are equipped with an IEEE 802.11 interface, and each of them shares a portion p of the available contents, each represented by a file of size 1 KB. The total number of existing contents is $M = 100$. Additionally, Agents "feel the need" for a content (randomly chosen among the ones not being provided) according to a Poisson process with rate equal to 0.05 requests/s. The needed content is requested from the Broker for a maximum of 30 s; if no reply ensues (e.g., because the Broker is not within radio range), a timeout expires and the Agent loses interest in the service. Upon receiving a positive Reply from the Broker, the requesting Agent finds a route toward the provider Agent by using the AODV protocol.

As far as storage is concerned, Agents are assumed to memorize contents in two distinct areas:

- a *long-term memory*, which stores the contents the Agent uses and publishes (or has a strong interest in sharing). This area is initialized at the beginning of the simulation (in real life, this may correspond to initializing it once a day) and is not altered during the emulation process;
- an optional *cache* (of finite, limited size), where the Agent can store information for which it has no use (e.g., because the Broker instructed it to do so).

The Agents are able to provide the contents stored in either area. Notice that, unless otherwise specified, the Agents are assumed *not* to have a cache (or, equivalently, to be unwilling to use it).

When moving from a coverage area to another, Agents choose (with equal probability) one of the following behaviors: registering to the local colony (leaving the previous one) or keeping the former affiliation and using the backbone to send and receive data. This has manifold implications: on the one hand, registering to the local colony allows Agents to leverage the direct connection to the local Broker; on the other, additional overhead is required to handle the (de-)registration process, and therefore it might not be advisable for highly-mobile nodes or in case of small Brokers' coverage areas.

V. Performance evaluation

In this section, we provide a thorough evaluation of Figaro's basic features, in terms of hit ratio (i.e., the probability that a content requested while the Agent is in the coverage area of a Broker is *discovered and retrieved* before the timeout expires) and overhead, as well as in terms of its more advance features, i.e., the reputation and caching mechanisms. Unless otherwise

specified, the results were obtained averaging 10 independent simulation runs.

A. Basic features

We evaluate the basic features of Figaro by looking at the hit ratio and at the overhead it generates. Specifically, we analyze scenarios with different values of p (0.1 or 0.4) and with/without the presence of a backbone and a Proxy. Additionally, as a benchmark, we compare Figaro with a simple content-retrieval mechanism, referred to as "flat flooding", which exploits a flat peer-to-peer exchange on ad-hoc node connectivity and it lets queries propagate according to the Mitigated Flooding scheme in [4]. Such scheme spatially limits the propagation range of a query through a Time To Live (TTL) value; also, it avoids the rebroadcasting of already solved requests by means of a query lag time[2].

The hit ratio is reported in Figure 2 for Figaro and flat flooding with $p = 0.1$, and in Figure 3 for Figaro and flat flooding with $p = 0.4$. The curve related to the case with Proxy is the same for both figures ($p = 0.1$) and is used for comparison purpose, as will be explained later.

As far as scalability is concerned, the plots show that Figaro is able to support a quite heavy load – in terms of both Agents and queries – without performance problems. The physical layer of the colony (e.g., network congestion, collisions...) represent the primary cause of performance degradation. Conversely, the logical layer (i.e., the overlay network) has a positive effect over scalability, as it effectively reduces the overhead.

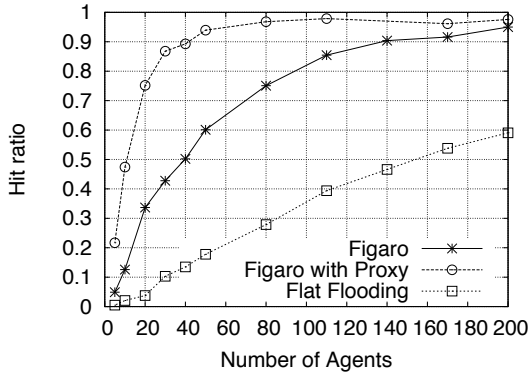

Figure 2. Hit ratio comparison between Figaro, flat flooding and Figaro with Proxy, with low content density ($p = 0.1$)

By looking at the plots, it can be seen that the hit ratio in Figaro grows almost exponentially (actually following the quantitative behavior of the birthday problem), and the speed at which the saturation zone is reached depends on the number of services the Agents share. The hit ratio for the flat flooding scheme grows much more slowly, given the blind, inefficient content search mechanism. Then, comparing Figaro with and

[2]Nodes receiving a query message wait for a query lag time: if they observe responses to the query in the meantime, they refrain from forwarding the query any further

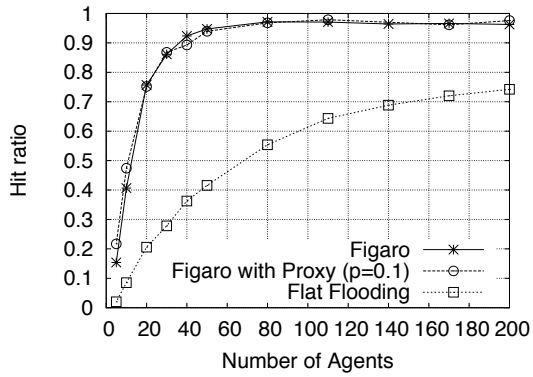

Figure 3. Hit ratio comparison between Figaro and flat flooding, both with medium content density ($p = 0.4$), and Figaro with Proxy for low content density ($p = 0.1$)

without Proxy, it is clear that using a Proxy dramatically improves the hit ratio, especially when the content density is low. Intuitively, it is like having an n-times larger colony (where n is the number of Brokers connected to the same Proxy). This intuition is confirmed by the matching curves of Figaro with and without Proxy in Figure 3: in a scenario where $p = 0.1$, using a Proxy guarantees the same hit ratio as that of a scenario without Proxy and with $p = 0.4$.

Another important parameter to evaluate is the overhead, i.e., the portion of the total traffic which is not represented by information content. It includes both the headers of the lower protocol levels and packets that Agents and Brokers exchange to manage the colony and find content. Figures 4 and 5 show how the overhead in Figaro depends on the number of Agents and the presence of a Proxy. In particular, it is shown to be nearly inversely proportional to the hit ratio, as a higher hit ratio implies that fewer queries are sent over the network. In the most favorable cases, the overhead stabilizes around 30% of the total traffic. Figure 6 and Table I summarize the components that form this value and that were used in our emulation. Conversely, in the flat flooding case, the overhead appears to be lower (but still 50% or more of

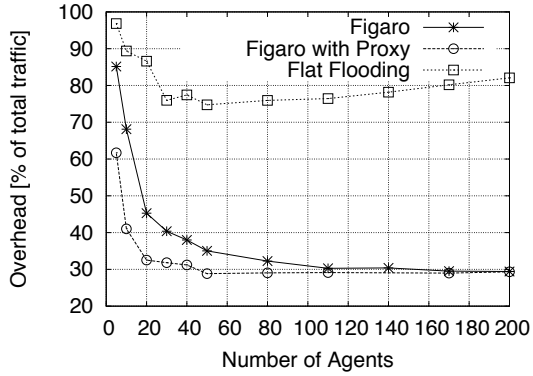

Figure 4. Overhead comparison between Figaro, flat flooding and Figaro with Proxy, with low content density ($p = 0.1$)

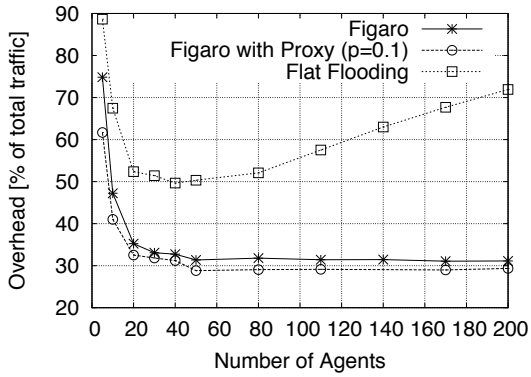

Figure 5. Overhead comparison between Figaro and flat flooding, both with medium content density ($p = 0.4$), and Figaro with Proxy for low content density ($p = 0.1$)

the total traffic) only for "critical numbers" of Agents, i.e., between 20 and 80: fewer Agents depress the hit ratio and thus increase the overhead, while a higher number of Agents triggers an overwhelming number of replies to a single query.

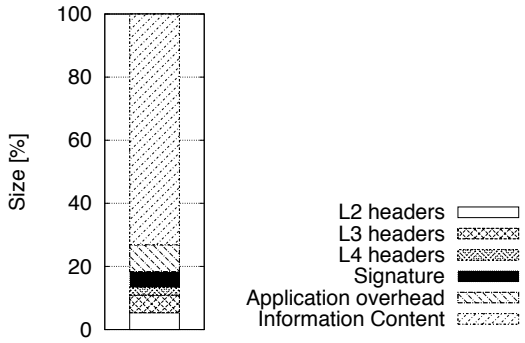

Figure 6. Composition of the overhead for a typical transaction (content is assumed to be 1 KB in size)

B. Agents' reputation

As far as the reputation mechanism is concerned, we are interested in evaluating:

- whether the system can effectively tell good and bad Agents apart;
- the impact of the duration w of the time window;
- the impact of the reputation threshold T_R.

Figures 7 and 8 plot the time evolution of the significance threshold $T_S(t)$ and of the reputation score for different values w of the time window. We selected a case study ($p = 0.1$ and 140 Agents, of which half are free-riders, never providing the services they claim to provide) where the small information density and the high number of free-riders require the system to be able to tell precisely on which nodes it can rely upon.

By looking at Figure 7, we observe that the length w of the time window directly affects the value of the significance threshold $T_S(t)$. This is rather intuitive, since a longer time

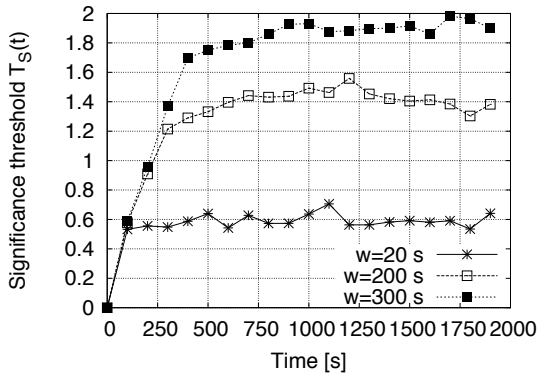

Figure 7. Evolution of the significance threshold $T_S(t)$ over time, for different durations w of the time window $W(t)$

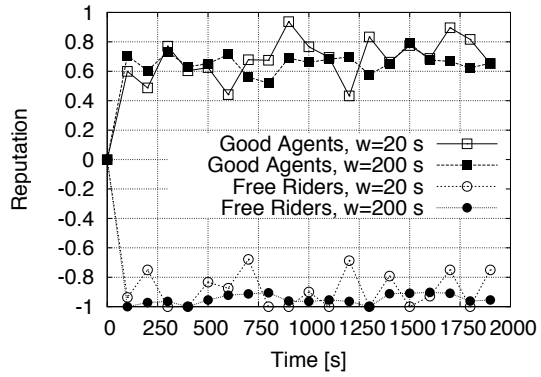

Figure 8. Evolution of the reputation score over time, for different durations w of the time window $W(t)$

window implies that a higher number of feedbacks needs to be taken into account. Also, $T_S(t)$ grows from time 0 to time w and, after this initial transient, it remains quite stable.

Figure 8 shows how the reputation scores of good Agents and free-riders evolve over time. Consistently with the definition in Equation (2), the length of the time window does not affect the average value of the reputation. However, when w is too small, the reputation scores tend to have wider oscillations around their average value. This behavior impacts the choice of the reputation threshold T_R.

As discussed in Section III-B, the threshold T_R must be a negative value, small enough (in absolute value) to ensure that all (or most of) the free riders' score is below it, and large enough to avoid (too many) false positives. Intuitively, with reference to Figure 8, it should clearly separate the regions where good and bad scores fall.

As shown in Figure 9, if w is large enough to guarantee that free riders' score is constantly very low (e.g., $w = 200\,s$), the choice of any threshold value between -0.1 and -0.9 allows the mechanism to work very efficiently: good and bad Agents are quickly and effectively told apart and they are provided with very different qualities of service. Notice that the hit ratio for good Agents in the case $T_R = -0.1$ is slightly lower, due to the contribution of false positives.

Table I
OVERHEAD FOR A TYPICAL FIGARO TRANSACTION

Type	Content	Appl. overhead	Signature	L4 headers	L3 headers	L2 headers	Total
SReq		26		8	20	22	76
SRep		45	64	8	20	22	159
Req		16		8	20	22	66
Data				8	20	22	50
							351

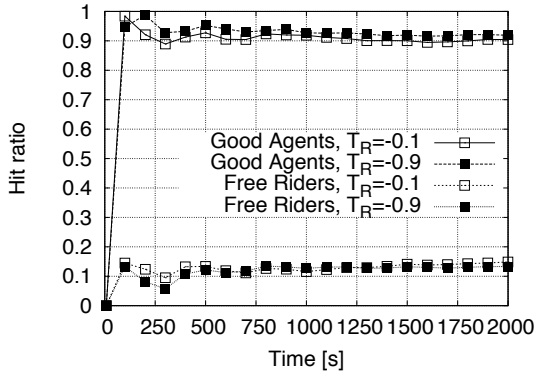

Figure 9. Hit ratio for good and bad Agents, for $T_R = -0.1$ and $T_R = -0.9$ ($w = 200\,s$)

Furthermore, we highlight that if the duration w had been smaller, $T_R = -0.9$ would not have been a good choice: indeed, as shown in Figure 8, the reputation score for free riders occasionally exceeds such threshold and it generates, as it were, false negatives.

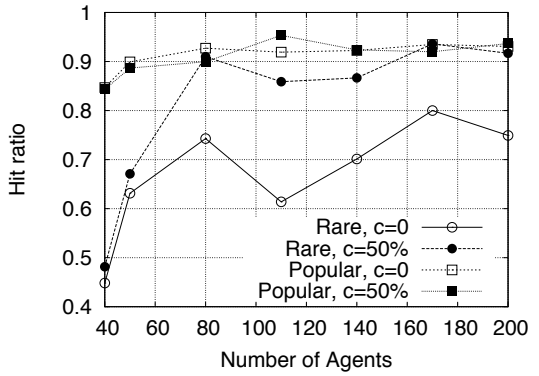

Figure 10. Hit ratio for rare and common contents, with $c = 0\%$ and $c = 50\%$

C. Rare content

We study the way Figaro handles rare content in an emulation scenario in which one out of 10 contents is ten times rarer than any other. In particular, we focus on the case $p = 0.4$ for popular content and $p = 0.04$ for rare content; indeed, using smaller values of p could force rare content into oblivion from the outset. The size of Agents' cache is assumed to be 3 KB – i.e., each Agent can cache up to 3 services.

We call c the percentage of nodes which choose to collaborate with the Broker, i.e., which use their cache according to the Broker's instructions. Comparing the cases $c = 0\%$ and $c = 50\%$ in Figure 10, it is clear that our scheme performs in a very efficient way, allowing rare content to achieve the same hit ratio of popular content, even in a plausible case in which one out of two Agents does not collaborate.

In the following, we investigate the impact of the collaboration level (i.e., the percentage of collaborating Agents) on the hit ratio for rare content. In other words, we answer the following question: *how many Agents should collaborate in order to eliminate, or to significantly reduce, the hit ratio difference between popular and rare services?*

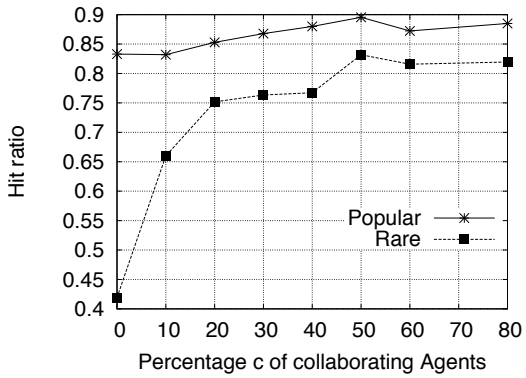

Figure 11. Hit ratio for rare and popular content as the percentage of collaborating Agents varies

Figure 11 provides a qualitative answer for the scenario being tested: while the hit ratio of popular content is minimally affected by the percentage of collaborating Agents, as little as 20% collaborating Agents are needed to bring the hit ratio of rare content within 15% of the hit ration of popular content. This provides a strong indication that our collaboration scheme is effective even in those scenarios where a small number of Agents can (i.e., has cache room) or wants to collaborate.

Further increasing the percentage of collaborating Agents brings no additional benefit. This mirrors the Broker's behavior, which stops considering as rare some content when more than half the Agents in its colony provide it. The rationale for this is that if Brokers fill the Agents' caches with semi-popular contents, there would be no room for other rare services.

D. Benefits of collaboration

Here, we show the system performance obtained in presence of collaborating, non-collaborating and misbehaving Agents.

To derive these results, we set $c = 50\%$ and $\alpha = 10$; however, other tests with α in $[5, 20]$ provided similar performances[3].

As shown in Figure 12, collaborating Agents experience a much lower load (i.e., they are required to provide a lower number of services). Therefore, following the Broker's caching suggestions is not only a contribution to the global quality of the service (see Figure 10), but also a rational way to pursue a selfish objective. Furthermore, as shown in Figure 13, being honest (either collaborating or not) is the best *selfish* strategy for Agents.

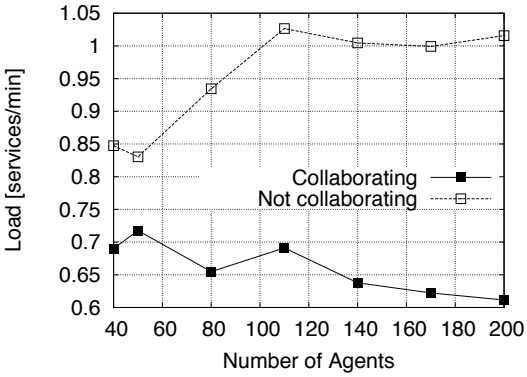

Figure 12. Load upon collaborating and non-collaborating Agents

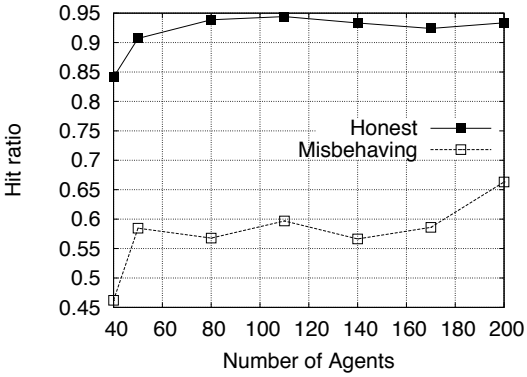

Figure 13. Hit ratio for honest and misbehaving Agents

Interestingly, caching also has the positive side effect of reducing the usage of the backbone. Indeed, caching increases the availability of a service inside the colony, and consequently reduces the need to search for it outside (i.e., asking the Proxy). Figure 14 confirms this statement. This is of particular importance when the backbone is not wired but implemented with cellular technologies such as UMTS. In those cases, reducing its usage results in significant monetary savings.

VI. Conclusions and Future Work

We have presented Figaro, a content discovery solution for wireless mobile environments, based on the publish/subscribe (pub/sub) protocol. Figaro includes several entities, namely Agents, Brokers and Proxies, that cooperate to the task of making content easily discoverable and retrievable. Contributing to the effectiveness of Figaro are its feedback and reputation mechanisms as well as a Broker-driven caching scheme in which the Broker suggests to the Agents what content to cache in order to maximize the hit ratio (global and Agent's) and to minimize the Agent's load. We have evaluated the system performance in terms of both hit ratio and overhead, and benchmarked it with a flat peer-to-peer solution based on the mitigated flooding of queries.

Future work will focus on improving the mechanisms of reputation and caching. At a higher level, we will take into account some additional scenarios, including vehicular Agents, mobile Brokers (e.g., integrated in buses) or infrastructure-less interactions (e.g., in zones outside the Brokers' radio coverage).

[3]Note that with low values of α, there is the risk to water down the penalty for cheating Agents. Conversely, higher values may result in the banishment of Agents which, due to connectivity failures, did not provide a single cached service.

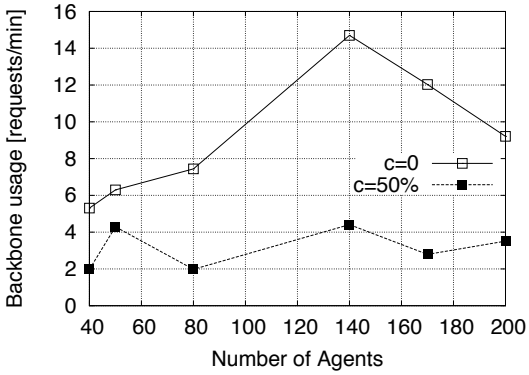

Figure 14. Backbone usage for $c = 0$ and $c = 50\%$

References

[1] L. Liquori, D. Borsetti, C. Casetti, C.-F. Chiasserini, "An Overlay Architecture for Vehicular Networks," *IFIP Networking* 2008.

[2] The ns-3 network simulator, http://www.nsnam.org

[3] J.-Y. Le Boudec, M. Vojnovic, "Perfect Simulation and Stationarity of a Class of Mobility Models," *IEEE INFOCOM* 2005.

[4] N. Chang, M. Liu, "Optimal Controlled Flooding Search in a Large Wireless Network," *WiOpt* 2005.

[5] Q. Yuan, J. Wu, "Drip: A Dynamic Voronoi Regions-based Publish/Subscribe Protocol in Mobile Networks," *IEEE INFOCOM* 2008.

[6] E. Yoneki, P. Hui, S. Chan, J. Crowcroft, "A Socio-Aware Overlay for Publish/Subscribe Communication id Delay Tolerant Networks," *ACM MSWIM* 2007.

[7] M. X. Goemans, L. E. Li, V. S. Mirrokni, M. Thottan, "Market Sharing Games Applied to Content Distribution in Ad-Hoc Networks," *ACM MobiHoc* 2004.

[8] N. B. Salem, J.-P. Hubaux, M. Jakobsson, "Reputation-based Wi-Fi Deployment Protocols and Security Analysis," *ACM WMASH* 2004.

[9] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks," *ACM WWW* 2003.

[10] R. L. Rivest, A. Shamir, "PayWord and MicroMint: Two Simple Micropayment Schemes," *Tech. Rep.,* MIT Laboratory for Computer Science, 2001.

[11] M. Fiore, F. Mininni, C. Casetti, C.-F. Chiasserini, "To Cache or Not to Cache?," *IEEE INFOCOM* 2009.

[12] C. Boldrini, M. Conti, A. Passarella, "Modelling Data Dissemination in Opportunistic Networks," *ACM CHANTS* 2008.

[13] C. Boldrini, M. Conti, A. Passarella, "ContentPlace: Social-aware Data Dissemination in Opportunistic Networks," *ACM MSWIM* 2008.