Content Discovery and Caching in Mobile Networks with Infrastructure

Francesco Malandrino, *Student Member, IEEE,* Claudio Casetti, *Member, IEEE,* and Carla-Fabiana Chiasserini, *Senior Member, IEEE*

Abstract—We address content discovery in wireless networks with infrastructure, where mobile nodes store, advertise and consume content while Broker entities running on infrastructure devices let demand and offer meet. We refer to this paradigm as *match-making*, highlighting its features within the confines of the standard publish-and-subscribe paradigm. We study its performance in terms of success probability of a content query, a parameter that we strive to increase by acting as follows. (i) We design a credit-based scheme that makes it convenient for rational users to provide their content (thus discouraging free-riding behavior), and it guarantees them a fair treatment. (ii) We increase the availability of either popular or rare content, through an efficient caching scheme. (iii) We counter malicious nodes whose objective is to disrupt the system performance by not providing the content they advertise. To counter the latter as well as free riders, we introduce a feedback mechanism that enables a Broker to tell apart well- and mis-behaving nodes in a very reliable manner, and to ban the latter. The properties of our match-making scheme are analysed through game theory. Furthermore, via ns-3 simulations, we show its resilience to different attacks by malicious users and its good performance with respect to other existing solutions.

Index Terms—Architectures and protocols for mobile networks, content discovery, content caching.

1 Introduction

Since peer-to-peer content sharing took the Internet by storm a few years ago, there has been a constant drive to export such an approach to mobile wireless networks. Although 2- or 3-G cellular connectivity is available, it is undesirable to use it to transfer large data files. It is often better to use the cellular network to download (e.g., from the Internet) the data items of interest to some nodes of the wireless network, and then use other techniques, e.g., peer-to-peer, to deliver them to the mobile devices. This approach is especially beneficial if data access patterns are localized, i.e., some items are more likely to be requested in a certain area. This is the case of local news or traffic information, or of several mobile users simultaneously trying to fetch a multimedia content from the Internet, as recently happened to AT&T [1] as a consequence of the introduction of iPhones.

Network dynamics in wireless and wired environments, however, are fundamentally different. Node churning, for example, is a common hurdle in peer-to-peer systems for wireline networks: mobility and variable channel conditions in wireless networks only exacerbate it. Thus, it is of paramount importance that the content carried by mobile users is easily, promptly "discoverable" and that its carriers are reliable when it comes to providing the content to others.

To this end, an efficient content discovery paradigm for mobile networks is needed. One possible candidate is the publish/subscribe (pub/sub) paradigm, which allows an asynchronous content exchange between publishers (providers) and subscribers (consumers). Content attributes are specified by publishers and, through filtering techniques, subscribers are delivered content whose attributes match constraints defined by them. However, in pub/sub systems the implication is usually that content is delivered to subscribers as soon as it becomes available through one or more publishers. Given the fleeting connectivity mobile users experience, this behavior may quickly lead to bandwidth waste and low hit probability. An alternative is represented by quorum-based replication schemes, where content update and request operations are carried out in interacting subsets of nodes, called read quorum and write quorum. Again, although specifically designed for distributed systems, quorum schemes are hardly a good choice in mobile networks, mainly due to the overhead they generate and the complexity in controlling the topology.

In this work we take a different approach. We present a content discovery solution, called Figaro, where mobile users are supported by an infrastructure – a scenario that finds wide application in the real world. In Figaro, mobile users, named Agents, request content items of their interest and, in their turn, make content items available to others. To ease the information sharing, users advertise, i.e., inform infrastructure nodes, named Brokers, about which content they are willing to provide, and Brokers assist requesting Agents in the content discovery process. Content is assumed to be static, as we leave content consistency out of the scope of the paper. Without loss of generality, content updates are considered as merely other content that Agents may wish to download. Additionally, we base our system on the transfer of an indivisible content unit (considering a molteplicity of such units as they represent different content types). Again, with marginal changes to the Figaro system, but with considerably heavier notation, content units could be replaced by pieces, or chunks, of a single unit

• *F. Malandrino, C. Casetti and C.-F. Chiasserini are with the Dipartimento di Elettronica, Politecnico di Torino, Torino, Italy.*
E-mail: lastname@tlc.polito.it.

in order to exploit a finer granularity in data dissemination.

To distinguish Figaro from standard pub/sub systems, we refer to its paradigm as *match-making*, highlighting its capability to let demand and offer meet *when the need arises*, while arbitrating the information flow between providers and consumers to account for the specific characteristics of mobile networks. Also notice that, in contrast with the pub/sub paradigm, the Agents selected as providers do not have the possibility to (legitimately) refuse to provide a content. In Figaro, the underlying assumption is that the Broker "knows better", i.e., it has a more reliable knowledge of whether an Agent has to provide a service.

One of the most important performance metrics we consider in our system is the query success probability, that is, the probability that a content query is matched with an Agent that owns the desired content, is willing to provide it, and faultlessly sees the transfer through. To ensure high success probability for content queries as well as a fair treatment to users, we act as follows.

(i) We associate to each Agent a *credit balance*, which increases when the Agent provides a requested content and decreases when it consumes a content. The same concept has been previously exploited to favour traffic routing in ad hoc networks [2]. We revise this approach and apply it to content discovery, showing that it can make content provisioning a rational choice for self-interested Agents, hence it can discourage rational Agents from acting as free riders. Also, by letting the Agents' balance depend on the size and popularity level of the provided/requested content, we guarantee fairness in spite of the different characteristics of the information items.

(ii) We define a feedback mechanism that allows Brokers to track the success of a content transfer, identifying and banning those Agents that do not provide the content they advertise. They can be either Agents acting as free riders or malicious users that aim at disrupting the system. We refer to the latter as *disruptors*, since, regardless of whether they request content or not, their main goal is to disrupt the success probability of queries issued by others. The feedback mechanism is designed so that Brokers can detect and discard negative feedbacks that are likely to be part of a bad-mouthing attack.

(iii) To guarantee a fair treatment to Agents providing content items with different characteristics, as well as to evenly distribute the load of content provisioning, we exploit Agents' caching capabilities. We formulate caching as an optimization problem that aims at maximizing the system fairness and efficiency, and design a heuristic that closely approximates the optimal solution while accounting for the system dynamics.

The rest of the paper is organized as follows. Sec. 2 reviews previous work, while Sec. 3 describes our content discovery scheme. Sec. 4 introduces the credit, feedback and banning mechanisms, whose effectiveness is analysed in Sec. 5 through game theory. The resilience of Figaro to different attacks is discussed in Sec. 6, and confirmed by the results in Sec. 7, which however highlights the aforementioned fairness issue related to different characteristics of the content. In order to address it, in Sec. 8 we introduce our caching strategy, which is investigated in Sec. 9. Finally, Sec. 10 concludes the paper.

2 Related work

Our match-making paradigm draws from the pub/sub approach, which has been widely investigated in the literature. We point out from the outset, however, that most works focus on wired scenarios, or on wireless ad hoc networks without any infrastructure. The opportunities offered by the presence of an infrastructure in a wireless environment are investigated in [3], which, however, does not address fairness, cooperation, or caching.

Associating network nodes with a balance is an idea that has been often exploited to enforce cooperation among self-interested nodes in wireless ad hoc networks, either for traffic routing [2], [4] or for channel access [5]. Note, however, that the seminal work in [2] requires the nodes to be equipped with a tamper-resistant hardware (i.e., a security module manufactured by a limited number of trusted manufacturers), in order to prevent attacks. The study in [4], instead, does not deal with attackers at all. More recent works, e.g., [6], [7], still rely on the assumption that a security module is available, and propose a distributed incentive protocol for multi-hop routing in mobile networks. We point out that in Figaro nodes are not required to embed any tamper-resistant device; indeed, through a balance- and feedback-based mechanism, the scheme itself ensures resilience to both free riders and attackers.

As for feedback-based schemes, of particular relevance is the pioneering work in [8], which introduces a reputation mechanism to enforce cooperation among rational nodes of a mobile ad hoc network. Many later studies have focused on cooperative routing in ad hoc networks [9] and in overlay networks [10]. Note that the proposed solutions refer to a different type of cooperation with respect to Figaro, i.e., message forwarding instead of content transfer. Consequently, the attack they consider is packet dropping, while in Figaro we address the problem of nodes that do not provide an advertised content when requested by the Broker. Also, in Figaro there is no need for sophisticate misbehavior detection and identification schemes of misbehaving nodes, as the Agents know exactly when they are victim of either a free rider or a disruptor, and can notify the Broker of the identity of the attacker (i.e., the Agent who did not to provide them with a content).

At the application layer, solutions for content provisioning have been presented in [11], where a reputation-based scheme is used to reduce the load over a 3G network. Monetary penalties and incentives are given to non-cooperative and caching nodes, respectively, while the choice of the content to cache is left to the Agents and modeled as a market sharing game. Unlike previous work, our approach is simple, lightweight and it does not assume that Agents are associated to a billing account: in Figaro incentives and penalties are circumscribed to Figaro itself and, since Agents do not directly choose which peer they will retrieve content from, Figaro has high resiliency to reputation attacks.

The study in [12] focuses on a mixed pedestrian and vehicular scenario, where users can provide and consume content items. Queries are opportunistically propagated through the network, but only within a few hops from the originator. When one of the nodes receiving the query is aware of another

providing the desired content, the query is solved. The main difference with respect to our work is that [12] does not address the presence of an infrastructure, and how it can be used. Still in the context of vehicular networks, in [13] the popularity of content items is taken into account by vehicles when deciding i) which queries should be answered first and ii) which contents the vehicles shall retain in their cache. The overall objective is to enhance the availability of the most popular items. On the contrary, Figaro aims at providing a fair service to all users, including the ones requesting or providing scarcely popular content. Also, similarly to [12], in [13] the role of the infrastructure is not addressed. Among the works setting in infrastructured scenarios, [14] envisions roadside "digital billboards," pushing location-specific advertisements to the vehicles passing by. Such information can be further propagated by vehicles themselves. With respect to [14], we do not restrict to a specific type of content (i.e., advertisement) and use the infrastructure for a different purpose (i.e., to arbitrate the content transfer between users rather than to push new content to them).

As far as caching is concerned, again most schemes designed for wireless networks, e.g., [15], [16], focus on distributed, infrastructure-less scenarios. As a result, they imply a complexity level that is exceedingly high for Figaro, which leverages the presence of Brokers and their centrality in the system architecture to simplify the network management.

Finally, we mention BubbleStorm [17], a well-known scheme for content replication and provisioning, which is based on a probabilistic exhaustive search paradigm in wired overlay networks. Unlike Figaro, BubbleStorm assumes the nodes to be always willing to store a copy of the content. As soon as it is generated, BubbleStorm propagates the content on a random graph defined on the overlay network. Queries are propagated following the same strategy, and they succeed if at least one copy of the query reaches a node that stores a copy of the content. In Sec. 9.2, we will use BubbleStorm as a benchmark for the performance of Figaro.

An early version of this work, sketching the match-making paradigm for content discovery, can be found in [18].

3 The Figaro system

Unlike most existing overlay networks, Figaro operates according to a match-making paradigm. Its operation and the relationship among the different entities can be described at a logical as well as at a physical level.

At a *logical* level, Figaro features two main types of nodes: Agents and Brokers. Agents store, advertise and consume content items, while it is the Brokers' task to let demand and offer meet. Agents store either self-produced content (e.g., own videos one wants to share) or content they have a specific interest in spreading (e.g., road maintenance vehicles may advertise an unscheduled road closure). As previously remarked, content items are assumed to be static with respect to the system time scale, i.e., they are updated at intervals that are much longer than any interaction between Agent and Broker, e.g., once a day.

At a *physical* level, Agents are mobile devices (either on-board units or hand-held), while Brokers are middle-end devices, integrated in an roadside infrastructure and interconnected via a reliable wired or wireless backbone. As far as network connectivity is concerned, each Broker is colocated with a router and associated to an IP subnet. The router also features one or more IEEE 802.11 interfaces acting as Access Points (APs) and providing hot spot connectivity to mobile devices in the area. Mobile nodes are thus hosts of the router subnet to which the Broker is associated.

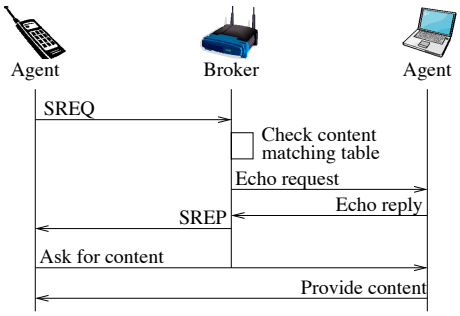


Fig. 1: Basic message exchange between Agents and Broker.

When the mobile device embedding the Agent associates to an AP, it discovers the Broker that controls it. If the Agent chooses to register with this Broker, it becomes part of the Figaro system and starts advertising to the Broker the content it is willing to share with others. The Broker maintains a content-based matching table, where it stores the following information for each Agent: 1) a unique Agent identifier; 2) its IP address; 3) the MAC addresses of any interface (e.g., 802.11, Bluetooth) the mobile node carries; 4) the content it makes available to others. The set of Agents registered to the same Broker is called *Colony*. In keeping with our previous definitions, a Colony corresponds to the hosts of the subnet associated to the Broker (i.e., at the IP layer, all Agents of a Colony have the same subnet ID as the Broker).

As illustrated in Fig. 1, a registered Agent can ask the Broker to identify another Agent carrying the content it needs, through a Service REQuest (SREQ) message. The Broker queries its own content-based matching table to identify a candidate Agent that can provide such content. As detailed in Sec. 4, these Agents are selected as candidate according to Colony-wise policies, aimed at pursuing specific objectives (e.g., high success probability and even load distribution on Agents). Next, the Broker checks that the candidate provider Agent is still reachable by a `ping` at the IP address with the Colony subnet ID (a simple solution in the spirit of [19]). If not, it selects another candidate provider from the matching table[1]. To avoid unpredictable iterations, the selection does not account for lower-layer metrics, such as the SNR on the links between Agents and AP, which can only be established upon checking the provider Agent's reachability.

If a candidate provider Agent is found, the Broker returns to the querying Agent a Service REPly (SREP) message carrying the IP address and the MAC address(es) of the candidate provider Agent. A transport-layer connection between the two Agents for the purpose of content transfer is subsequently

1. Deregistration of an Agent is enforced by a Broker after the Agent is found unresponsive to a number of consecutive attempts at pinging it.

established. Such a connection runs within the coverage area of the APs connected to the router interface.

If, instead, no candidate provider is found or if none of them replies to the `ping`, the search is relayed to a higher hierarchical level. To this end, we introduce an architectural entity, called Proxy, which is connected to all Brokers via the backbone. When a Broker receives a request for a content that is unavailable in its Colony, it forwards the request to the Proxy, which in turn queries the other Brokers. Also, when an Agent moves to a different Colony, the new Broker informs the previous one of the Agent's migration. For scalability reasons, a hierarchy of Proxies can be deployed, although we leave it out of the scope of this paper. The request is successful if the content is found in any of the Colonies composing Figaro. In this case, the connection between Agents runs through the routers colocated with the Brokers that control the Agents.

Agents report to their Broker the outcome of successful and unsuccessful content transfers with Agents identified as candidate providers. The outcome is notified through a *feedback* message, which the requesting Agent sends to the Broker after the content transfer.

4 Matching demands and offers

We consider a mobile system where Agents are rational and follow the same behavior in terms of querying activity. Let I be the number of content items that exist in a Figaro system composed by C Colonies; the items may differ by size and popularity. Let Λ be the per-Agent query generation rate. Upon a query generation, an Agent selects the item to ask for according to its popularity level, i.e., with probability $\pi(j,t)$, $1 \leq j \leq I$ ($\sum_{j=1}^{I} \pi(j,t) = 1$). Consequently, at time t within Colony k, each content j is requested with rate $\lambda_k(j,t)$, which is equal to $\Lambda\pi(j,t)$ multiplied by the number of Agents in Colony k at time t. We also denote by $P_k(j,t)$ the number of Agents (either under or out of coverage) advertising item j at time t in Colony k.

We design our match-making system in order to achieve the following goals:

1) high query success probability, in spite of the rational behavior of users and the different characteristics of the requested content (i.e., size and popularity level);

2) fair treatment of the Agents, i.e., the amount of service they provide is comparable to the amount of service they obtain;

3) resilience to Agents who do not provide the content they advertise (either free riders or disruptors).

To meet these objectives, we associate to each content j a Colony-wise value, denoted by $G_k(j,t)$, which is expressed in credits and may vary in time: an Agent that provides (receives) a content item earns (spends) an amount of credits equal to the content value. For each Agent i, we can therefore define a balance $b(i,t)$, expressed again in credits, which reflects the difference between the value of the content the Agent has provided and the value of the content it has obtained. Note that the exact, up-to-date value of each Agent's balance is only known by the Broker (although the Agents can compute their own rough estimate). We stress that since Agents are not expected to store or advertise any authoritative information about their balance, there is no need for tamper prevention, or tamper-resitant equipment.

We define $G_k(j,t)$ so as to take into account both the different size and popularity level of the content items. More specifically, for each Colony k we introduce the *content burden* metric, $B_k(j,t)$, which is the ratio of the query rate associated to content j within the Colony to the number $P_k(j,t)$ of Agents providing it at time t, i.e.,

$$B_k(j,t) = \frac{\lambda_k(j,t)}{P_k(j,t)} \tag{1}$$

Note that, in defining $B_k(j,t)$, we consider the query rate for content j coming from other Colonies to be negligible. Also, the burden takes larger values for popular content (characterized by high values of $\lambda_k(j,t)$) and for rare content (for which $P_k(j,t)$ is low), while it is smaller for content with low popularity or that can be easily found in the Colony. By denoting with $s(j)$ the size of the file representing content j, we define $G_k(j,t) = (g + B_k(j,t))s(j)$, where $gs(j)$ represents the baseline value of the content. It follows that large-sized, highly-popular content items, as well as rare items, will all be highly valuable. Also, we associate a value of g credits to each feedback message that an Agent belonging to Colony k sends to the Broker at time t to notify the outcome of the transfer of the requested content.

For the sake of clarity, let us first consider the case where there are no malicious Agents, and assume that Agents start with a zero balance. The balance of the generic Agent is updated as described below.

- When an Agent u, belonging to Colony k, requests a content j by sending an SREQ to the Broker and the Broker finds a candidate provider:
  (i) $b(u,t)$ is decreased by $G_k(j,t) + g$;
  (ii) if Agent u sends a feedback to the Broker, related to the transfer outcome of a requested content, $b(u,t)$ is increased by g;
  (iii) if Agent u is not satisfied by the transaction (e.g., the transfer fails to complete or does not occur at all), it sends a negative feedback; it is entitled to request the same content again (provided that the new query is made within a given time window) without further decreasing its balance[2].
- When an Agent v, belonging to Colony k, is selected as provider for a content query issued in Colony l ($l \neq k$):
  (i) $b(v,t)$ is increased by $G_l(j,t)$;
  (ii) if a negative feedback about the data transfer is received by the Broker, $b(v,t)$ is decreased by $G_l(j,t)$.

Note that feedback messages play a very important role in Figaro's credit scheme and are therefore awarded additional credits: by doing so, rational Agents will always provide a feedback if they can. Also, Agents have no incentive to provide a falsely negative feedback, as this would not restore their balance, but only give them the opportunity to request the same content again – which would be useless if the content

2. Subsequent feedbacks related to the same content do not bring any further increase in the balance of the issuing Agent.

has already been successfully received. In case no feedback is received for a content transfer, the transfer is assumed to be successful (i.e., the candidate provider is awarded $G_k(j,t)$ credits). The rationale is the following: since requesting Agents always have an interest in sending a feedback, connectivity problems likely prevented the Broker from receiving a feedback. If this were the case, the same connectivity problems would be the cause of the transfer failure (if any), and taking actions against the candidate provider would be unnecessary. Finally, we remark that, unlike voting schemes (e.g., [9]), in Figaro Agents do not choose their content providers, and thus whom they give a feedback about. This makes Figaro robust to attacks by malicious Agents, as discussed later.

Given the above credit scheme, the Broker can exploit the value of the Agents' balance to ensure Agents' cooperation in providing content. In particular, the Broker can determine whether a requesting Agent is entitled to receive further service and which Agent should be selected as candidate provider, according to the following rules:

- upon receiving an SREQ from Agent u, the Broker discards the SREQ if $b(u,t) < T_r$, where T_r is a negative threshold value (i.e., the requesting Agent has too low a balance to request a content);
- otherwise, a candidate provider is selected and the Broker appoints the Agent that has the lowest balance among the Agents advertising the requested content.

The choice of the lowest-balance Agents as providers may at first appear questionable and unfair to well-behaving Agents issuing the query, as such providers may not actually deliver the content. However, free riders and disruptors are handled by the banning mechanism about to be introduced. The choice of the Broker has instead the positive fallout that fairness among Agents sharing common and rare contents is improved.

The credit system described above makes cooperation, i.e., providing a content when requested by the Broker, necessary for the Agents in order to be able to obtain the content they need later on. The higher the T_r, the higher the amount of cooperation required. When all Agents are rational, game-theoretic methods can be used [4] to assess the value for T_r that yields optimal performance. However, in Figaro we also take into account the presence of malicious Agents, whose only purpose is to disrupt the system performance. To counter them, we introduce a banning mechanism, which changes the nature of the problem and, as a positive side-effect, also represents a further incentive for rational Agents to cooperate.

4.1 The banning mechanism

Figaro uses banning to keep malicious Agents out of the Colony and, thus, impair their actions. Every time the Broker receives a negative feedback related to an Agent (and it deems it credible, as described below), that Agent is banned for a certain period of time. While banned, the SREQs transmitted by the Agent are dropped by the Broker and the Agent cannot be selected as a candidate provider. We stress that when a banned Agent issues a SREQ, its balance is decreased anyway by the value of the requested content. Recall that Agents are not aware of their balance. To prevent an Agent from foreseeing the ban periods and avoiding to request content items while being banned, ban periods start after a random time since banning is triggered.

The ban duration grows exponentially: on the n-th time that an Agent is banned, the duration is computed as $T_b(n) = T_0 a^{n-1}$, with $a \geq 1, n \geq 1$. T_0 is set small enough so as not to excessively penalize those Agents that occasionally fail to provide a content, and a large enough so as to rapidly and effectively exclude malicious Agents from the Colony. The actual choice of T_0 and a depends on the application and, as shown later in the paper, on the system status; also, the counter recording the number of bans for each Agent can be periodically reset.

We stress that the ban mechanism not only allows Brokers to counteract malicious Agents but it also serves as further incentive for rational Agents to provide the requested content when selected as providers. This makes our study significantly different from the one in [4]. Also, Figaro is immune to the adverse impact that mobility may have on the effectiveness of reputation and ban schemes: banned Agents may try to move to a different Colony to nullify their banning, or newly arrived Agents may be unable to figure out the trustworthiness of their neighbors. Indeed, in Figaro (i) Brokers can exploit the backbone to exchange information about banned Agents or about the balance of Agents that move from one Colony to another, and (ii) newcomers (like all other Agents) rely on Brokers for the selection of the candidate provider.

4.2 Feedback credibility

As is evident from the description above, in Figaro bad-mouthing attacks, in which attackers assign falsely negative feedbacks to the Agents that provide them with a content, would cause a serious malfunctioning. To ensure robustness to bad-mouthing attacks, each Broker implements a simple, yet effective, credibility filter, based on the notion of *negative feedback ratio*.

We begin by introducing some definitions. Given Agents u and v, the negative feedback ratio $\nu_I(u,v)$ is the fraction of negative feedbacks issued by Agent u on Agent v's behavior. We then define the following Colony-wise average values: $\bar{\nu}_I(u) = \frac{1}{N_u}\sum_v \nu_I(u,v)$, i.e., the ratio of negative feedbacks issued by u averaged over the number (N_u) of nodes that have served as candidate providers for u and belonging to the same Colony as u; $\bar{\nu}_R(v) = \frac{1}{N_v}\sum_u \nu_R(u,v)$, i.e., the ratio of negative feedbacks received by v averaged over all Agents belonging to the same Colony as v and for which v has acted as candidate provider.

Then, let us consider that the Broker receives from Agent u a feedback on the content provider v, and both u and v belong to the Broker's Colony. The Broker deems the feedback to be not credible if both the conditions below hold:

1) the ratio of negative feedbacks issued by Agent u, $\bar{\nu}_I(u)$, is higher than the average value computed over all Agents belonging to the same Colony as u (i.e., the issuer's view of the Colony is more negative than the average one);
2) the ratio of negative feedbacks given by u to v is higher than the average negative feedback ratio received by v:

$\nu_I(u,v) > \bar{\nu}_R(v)$ (i.e., the issuer's view of the provider Agent v is more negative than the average one).

The ratio behind the above conditions is the following: the Broker will not deem credible those feedbacks that portrait the target Agent *and* its Colony worse than other Agents do. Assume, for example, that the target Agent v has a negative feedback ratio of 0.1, and its Colony c has a negative feedback ratio of 0.05 (these are typical values we found in our simulations). Now, if an Agent u always issues negative feedbacks, u is either participating in a bad-mouthing attack, or experiencing severe connectivity problems: in any case, its negative feedbacks should not trigger a banishment of v or decrease v's balance.

If, instead, the candidate provider v and the Agent u issuing the feedback belong to different colonies, typically $\nu_I(u,v)$ is not statistically meaningful due to a small number of occurrences. Hence, the Broker of Agent u only evaluates the first condition and notifies the outcome to the Broker of Agent v. Based on this condition only, the Broker of Agent v assesses the feedback credibility. Notice that, when a feedback is considered not credible, it does not trigger the banning of the candidate provider and no action is taken against its issuer. The rationale of the latter choice is that, once the bad-mouthing Agent is identified and made harmless, the Colony can still benefit from its presence, as long as it correctly provides its content when asked. Also, Agents found to issue unreliably negative feedbacks are not necessarily attackers; they may simply be Agents whose ability to receive content items is impaired by some external reason (e.g., connectivity issues): banning them would be unfair.

In the following, we highlight the ability of the presented credit scheme and candidate provider selection policy to ensure a high query success probability, and we discuss the robustness of Figaro to the possible attacks by malicious Agents.

5 Ensuring cooperation in Figaro: A game-theoretic analysis

We now adopt a game-theoretic approach to show that our credit scheme, jointly with the banning mechanism, makes cooperation (i.e., providing the requested content when selected by the Broker as a candidate provider) the best choice for a rational Agent. We therefore focus on rational Agents and assume that none of them is malicious.

We model the system dynamics as a game, where, when selected by the Broker as candidate provider, an Agent can play two possible moves: to provide or not to provide the content. We first compute the payoffs corresponding to these moves. Then, we derive the strategic form of the game and show, by iterated dominance, the condition under which there is a unique Nash equilibrium, in which all players cooperate. We show that such an equilibrium is Pareto-optimal and also attains the maximum efficiency. These results hold in the homogeneous case, i.e., when content items have the same characteristics, as well as in the inhomogeneous case.

5.1 Payoffs and game solution

For the sake of simplicity, we first assume homogeneous conditions, i.e., independently of the considered Colony, all content items are represented by a file of the same size and have the same popularity level, and for each content there is an equal number of Agents storing the content. Hence, $G_k(j,t) = G(t), \forall k, j$.

We consider a generic Agent i belonging to Colony k and assume that, at a generic time t, it is selected as a candidate provider. Let $V(i,t)$ be the utility that Agent i can expect to obtain, i.e., the amount of service it will be able to receive in the future (not considering the possibility to be subsequently selected as a candidate provider). In Figaro, this corresponds to having $V(i,t) = b(i,t)$, i.e., the Agent's current balance. We denote with $V^+(i,t)$ and $V^-(i,t)$ the new utilities of Agent i in case it chooses, respectively, to cooperate and not to cooperate at time t. Also, let $c(j)$ be the cost of providing content j. The cost $c(j)$ is assumed to directly reflect the size $s(j)$ of the file representing content j, e.g., $c(j) = Ks(j)$, where K is a constant positive value. However, due to the assumption of homogeneous content, $c(j) = c = Ks, \forall j$.

Now, if Agent i decides to cooperate, it pays a cost c for providing the content, and its utility $V^+(i,t)$ will change due to the increase of its balance by $G(t)$. Its payoff will be:

$$U_c(i,t) = -c + V^+(i,t) = -c + [G(t) + b(i,t)]. \quad (2)$$

Conversely, if Agent i decides not to cooperate, it will not pay any cost. However, its utility $V^-(i,t)$ will reflect the fact that not only will its balance $b(i,t)$ not increase, but the Agent will also be banned for a time interval, whose duration depends on the number of bans already received. Let us assume that the Agent has already been banned $(n-1)$ times; considering that, during the ban period, it will issue an average of $\Lambda T_b(n)$ SREQ messages, and a decrement of its balance will correspond to each of them, its payoff becomes

$$U_{nc}(i,t) = V^-(i,t) = \max\left\{[b(i,t) - \Lambda T_b(n) G(t)], 0\right\}. \quad (3)$$

Note that, obviously, an Agent cannot retrieve less than 0 items thus $V^-(i,t) \geq 0$.

This is a rather peculiar game, as each player's payoff solely depends on its own move, and not on the opponent's one. As for the equilibrium, we can prove the following result:

Theorem 1. *The game described above admits (Cooperate, Cooperate) as the unique Nash equilibrium if*

$$T_b(n) > \frac{c - G(t)}{\Lambda G(t)} \quad \forall t, n. \quad (4)$$

Proof: The proof can be found in the Appendix. □

Thus, if the condition in (4) is met, the game will reach an equilibrium in which all rational Agents always cooperate, i.e., they provide the content the Broker asks them. Condition (4) must hold for any time instant t and for every n; hence, considering the expressions of c and $G(t)$, a possible choice for T_0 is given by: $T_0 > (K - g)/\Lambda g$. Such an expression can be read as: the higher the request rate Λ, the more severe the penalty that banned Agents receive during the ban period; thus, the smaller the value of T_0 needed to make cooperation a convenient strategy. Clearly, if $K \leq g$, banning is not necessary to make cooperation a rational choice for the Agents.

5.2 Optimality and efficiency of the equilibrium

When the condition in (4) holds, the strategy profile (Cooperate, Cooperate) is a Nash equilibrium and, since $U_c(i,t) > U_{nc}(i,t)$ for every Agent i, it is also Pareto optimal, i.e., cooperation is the best strategy that an Agent can follow without making someone else worse off. In order to assess the efficiency of this equilibrium, we determine the price of anarchy (PoA), which is defined as the ratio of the payoffs obtained by the players when the Nash equilibrium holds, to the payoffs obtained by the players if a globally optimal solution is enforced [20]. The following theorem holds.

Theorem 2. *The equilibrium (Cooperate, Cooperate) of the game described in Sec. 5.1 yields the same efficiency as the globally optimal solution.*

Proof: The proof can be found in the Appendix. □

Intuitively, when Agents are not free to chose whether to cooperate or not, an Agent that is selected as a candidate provider pays the cost c while its balance is increased by $G(t)$. Thus, $U_{opt}(i,t) = -c + [G(t) + b(i,t)] = U_c(i,t)$, i.e., PoA $= 1$. In other words, not only is the equilibrium (reached when all Agents cooperate) fair and Pareto-optimal, but it yields the very same efficient behavior as the globally optimal solution. This is not surprising, since cooperation means that Agents follow the suggestions of the Broker, which is in an excellent position to determine the optimal strategy.

5.3 The inhomogeneous case

We generalize the previous analysis to the case where items have different size, popularity level and availability, and popularity and availability may depend on the Colony. Thus, we now consider a cost $c(j) = Ks(j)$ and a content value $G_k(j,t)$ ($1 \le k \le C$, $1 \le j \le I$).

In this setting, the payoffs for a cooperating and not cooperating Agent i, which belongs to a generic Colony k and is requested to provide content j at time t, are given by:

$$U_c(k,j,i,t) = -c(j) + V^+(i,t) = -c(j) + G_k(j,t) + b(i,t)$$

$$U_{nc}(k,j,i,t) = V^-(i,t) = \max\left\{\left[b(i,t) - \Lambda T_b(n)\sum_{h=1}^{I}\pi(h,t)G_k(h,t)\right], 0\right\}$$

where we assumed that i has already received $n-1$ bans. In order for the Agents to cooperate, the condition $U_c(k,j,i,t) > U_{nc}(k,j,i,t)$ has to hold for any k, j, i and time instant t, i.e.,

$$T_b(n) > \frac{\max_j\left[c(j) - G_k(j,t)\right]}{\Lambda\sum_{h=1}^{I}\pi(h,t)G_k(h,t)} \quad \forall t, n. \tag{5}$$

Then, considering the expressions of $c(j)$ and $G_k(j,t)$, a possible choice for T_0, so that the condition in (5) is always satisfied, is given by:

$$T_0 > \frac{(K-g)\max_j s(j)}{\Lambda g \min_j s(j)}.$$

6 Resilience to attacks

Malicious Agents may try to break Figaro's balance mechanism by performing several types of attacks. In particular, they may behave as disruptors with the sole purpose of degrading the system performance. Figaro counteracts this behavior through the ban mechanism, which leaves out of the system an Agent for a given period of time, as soon as it receives a credible negative feedback. The effectiveness of banning is shown in Sec. 7.1, through ns-3 simulation.

Below, instead, we discuss the resilience of Figaro to the typical attacks that may be launched (independently or in a collusive manner) in online trading communities or, more in general, in reputation-based systems.

Ballot stuffing: a group of Agents collude to give each other positive feedbacks, in order to get an incorrectly high balance. In Figaro, Agents cannot freely choose whom they ask for the content they need, since the selection is performed by the Broker. Therefore, while it is possible to indiscriminately give a colluder a positive feedback, a large number of colluders is needed to make this attack viable (i.e., by increasing the likelihood that one of the colluders is selected as provider). Furthermore, an Agent has no practical way to artificially increase the number of feedbacks it is entitled to issue regarding its own colluders, since the number of requests it can issue is limited by T_r.

Bad-mouthing: a group of Agents collude to give negative feedbacks to others, so as to incorrectly lower their balance, and having them repeatedly banned. Again, the effectiveness of this attack is dampened by the Broker likely choosing a different provider for every request.

Negative/positive discrimination: an Agent provides the requested content only to a selection of other Agents, neglecting those it "does not like". This behavior will draw bans upon the Agent and is hardly effective in the long run.

Sybil attack: an Agent uses a large number of pseudonyms, thus gaining a disproportionately large influence on its own reputation scores, as well as the other peers'. Agents are identified via their IEEE 802.11 MAC address. An attacker could modify its MAC address to assume a new identity, but in this way its former identity would become unreachable and would be automatically de-registered from the colony, i.e., an Agent cannot have multiple *contemporary* identities, unless relatively complex hardware and software are used.

Whitewashing: an Agent misbehaves until it is banned and then assumes a new, clean identity. Whitewashing is normally impervious to detection attempts. The only protection that Figaro can deploy is by not letting Agents know they are banned (hence not letting the attacker know explicitly when it is time to assume a new identity). Still, Figaro is vulnerable to whitewashing by knowledgeable attackers who are aware that a negative feedback will get them banned: they can stay in the system until they are first requested to provide a content, ignore it and then assume a clean identity. Thus, in the case of applications for which resilience to whitewashing (and Sybil) attacks is highly critical, Agents may be required to perform a una-tantum, web-based registration (possibly with a CAPTCHA [21]) before they can register to a colony.

Furthermore, Agents could be required to have a private key and sign the messages they send.

7 Evaluating the credit and banning schemes

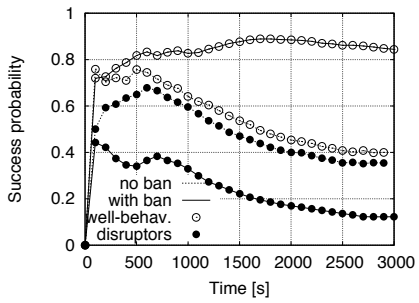
We provide an evaluation of Figaro's features described so far by using ns-3 simulations: firstly, by looking at its resilience against several types of attacks, and then by establishing to which extent it can actually enforce cooperation and fairness among the Agents.

Our investigation focuses on a pedestrian scenario, where four points of interest (POI) are placed at the corners of a 1000×850 m^2 rectangular area. In correspondence to each POI there is an 802.11 AP, integrating a Broker, all connected through a backbone. Agents, whose number is fixed to 100 unless otherwise specified, are equipped with an 802.11 interface and roam among the APs. Their movement follows the Random Trip mobility model, with an average pause time of 100 s and an average speed of 1.8 m/s. The Two-ray Ground model is used to represent the channel propagation conditions, and the transmission data rate between Agents and APs is controlled through the AARF technique [22], so as to adapt to the perceived channel conditions.

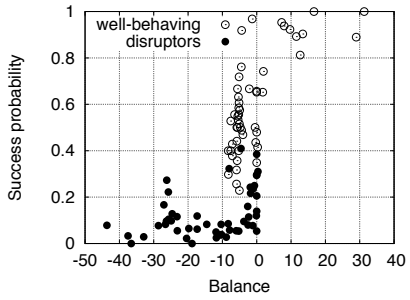
The content items are divided into four classes with different popularity and size. We consider two possible levels of content popularity, as well as two possible content sizes. Both the content sizes and the popularity levels differ by a factor 2. Specifically, class 1 items have size of 200 kB and popularity level equal to $1/3$, class 2 items have size of 100 kB and popularity level $1/3$, class 3 items have size of 200 kB and popularity level $1/6$, and class 4 items have size of 100 kB and popularity level $1/6$. For the sake of clarity while presenting the results, we consider $I = 4$, i.e., one content per class, and assume that each Agent advertises exactly one content (chosen with equal probability among the possible four), so as to associate each Agent to the class of content it provides. We stress, however, that simulations with a larger number of content items yielded qualitatively similar results.

An Agent "becomes interested" in a content according to a Poisson process with rate equal to $\Lambda = 0.02$ req/s. The requested content j is chosen, among those not stored by the Agent, with probability proportional to the content popularity level $\pi(j,t)$. The Agent then issues an SREQ message for that content, which is periodically refreshed until an SREP is returned, or until a timeout (set to 30 s) expires. If no reply ensues before the timeout, the Agent considers the query as failed. Instead, if a positive reply is received from the Broker, the requesting Agent asks the providing Agent for the content. Data are exchanged through a well-known UDP port. Agents are not required to implement any routing protocol, as the content is either available in the same Colony (i.e., subnet) or via the backbone. As for the other parameters, we set: $T_0 = 60$ s, $a = 3$, $K = 1$, $g = 0.5$.

All plots have been obtained by averaging 10 independent simulation runs, so as to obtain a 95% confidence level.



(a)



(b)

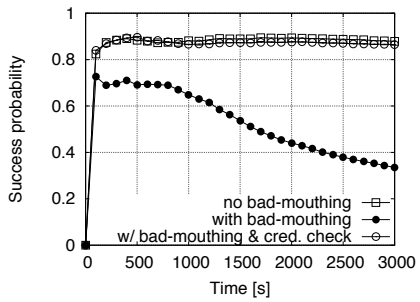
Fig. 2: Resilience to disruptors: (a) time evolution of the success probability for well-behaving Agents and disruptors, with/without banning; (b) success probability vs. balance for well-behaving Agents and disruptors, with banning enabled.

7.1 Counteracting disruptors, bad-mouthers and liars

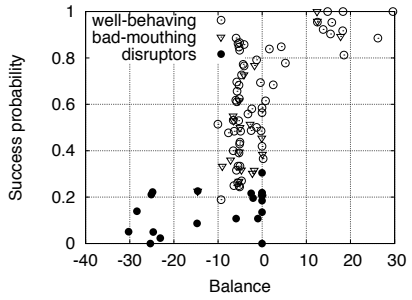
When our credit-based scheme is implemented, we have proved that all rational users have interest in cooperating, hence they will provide the content when they are asked for. Here, we are interested in evaluating to which extent Figaro can (1) protect well-behaving Agents from disruptors; (2) detect and discard falsely negative feedbacks, i.e., the ones issued by Agents taking part in a bad-mouthing attack; (3) make it disadvantageous for Agents to lie about the content they share in the Colony. Note that the plots presented here do not show the class of the content the Agents provide, as this is not significant for the aspects being taken into account.

We start by considering a scenario in which 50% Agents are willing to cooperate while the rest are disruptors. In our simulations, we consider that disruptors also issue content requests and that their behavior is unaffected by the credit and ban mechanisms, and we set $T_r = -5$ (the impact of T_r will be evaluated later).

First, to show the effectiveness of our banning mechanism, Fig. 2a presents the time evolution of the query success probability of well-behaving rational Agents and of disruptors, with and without banning. In absence of banning, well-behaving Agents and disruptors experience about the same success probability: the success probability decreases over time till a saturation value (namely, about 0.4), which is determined by the presence of a large percentage of disruptors. Conversely, with banning, the Broker can tell apart well-behaving nodes and disruptors in a very reliable manner, and the gap in performance between the two types of Agents widens. Indeed,



(a)



(b)

Fig. 3: Resilience to bad-mouthing: (a) time evolution of the success probability of well-behaving Agents, with/without bad-mouthing. The cases with and without feedback credibility check are shown; (b) query success probability vs. balance, for well-behaving, bad-mouthing and disruptor Agents, and with feedback credibility check.

with the passing of time, more and more disruptors are discovered and banned, hence made harmless. Consequently, over time the probability that a disruptor is elected as candidate provider decreases while the success probability of well-behaving Agents grows. However, the success probability of well-behaving Agents does not reach 1 and, similarly, the success probability of disruptors does not drop to 0. This is due to the following reasons: (i) the ban period of disruptors is limited, thus, at any time instant there may be still disruptors active in the Colony (although they will be detected and banned again); (ii) as studied in the next section, the balance of the Agents providing lower-value items (classes 3-4) may drop below the threshold T_r (i.e., they cannot gain as many credits as they would need to obtain the desired content).

Fig. 2b shows that, when the banning is enabled, disruptors have lower query success probability *and* lower balance (most of the times below T_r) than well-behaving Agents. Indeed, requests that come from banned Agents are discarded by the Broker but do trigger a balance decrease.

Next, we consider an even more challenging scenario, where 20% Agents are disruptors and other 20% take part in a bad-mouthing attack. Fig. 3 shows the time evolution of the query success probability for well-behaving Agents in presence of bad-mouthing attackers, in both the cases where the feedback credibility check (described in Sec. 4) is enabled and disabled. Results are compared also with the case where no Agent takes part in the bad-mouthing attack. We first observe that when the credibility check on negative feedbacks is disabled, bad-mouthing Agents slowly but steadily erode into the query success probability of well-behaving users, having them repeatedly banned. Conversely, enabling the credibility check allows well-behaving Agents to achieve the same performance as in the case where no bad-mouthing attack is launched. This behavior highlights two important facts: not only does the credibility check neutralize bad-mouthing (i.e., it has very few false negatives), but it also has very few false positives, i.e., it does not erroneously discard truly negative feedbacks. Indeed, if the feedbacks against real disruptors were discarded, a decrease in the well-behaving Agents' performance would occur, similarly to what is shown in Fig. 2a.

These observations are confirmed by the results in Fig. 3b, detailing the success probability of well-behaving, bad-mouthing and disruptor Agents, versus their balance values. Disruptors still have a lower balance and query success probability than the Agents (either bad-mouthing or well-behaving) that do provide the content they advertise. Recall that, as explained in Sec. 4.1, no action is taken against bad-mouthing Agents, once they are discovered and made harmless.

In addition to disruption and bad-mouthing, there is a further subtle, unfair behavior that Agents may follow: they omit, at registration time, to declare to the Broker which content they wish to share within the Colony, i.e., they pretend they have none. The Broker has no way to find out which content Agents have in their memory, thus these lying Agents will never be selected to provide a content. Also, since there is no evidence of unfair behavior, they will not be banned. However, Figaro effectively tackles this issue: lying Agents do not provide any content and, thus, their balance will soon reach T_r. From then on, they cannot obtain any service, which is the same effect banning would obtain. A similar effect occurs if Agents declare fewer content items than they have.

The benefit of setting T_r to a slightly negative value (namely, -5), as opposed to using a larger negative threshold (namely, -20) is evident from Fig. 4. When $T_r = -20$, lying Agents achieve almost the same success probability as well-behaving Agents. Conversely, when $T_r = -5$, the balance and success probability of liars severely degrades, i.e., lying about the stored content is not a good choice for rational users.

As a conclusion, a small negative value for T_r makes Figaro highly resilient to both disruption and lying about one's ability to contribute to the Colony.

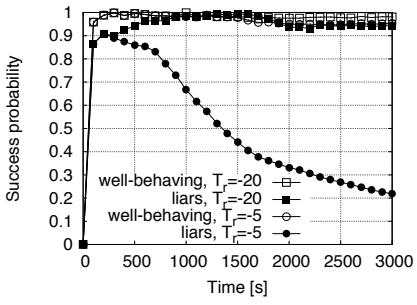
7.2 Cooperation and fairness

We assess the performance of Figaro in terms of fairness, by focusing on the query success probability obtained by the Agents providing the different types of content items, as listed in Sec. 7. In particular, we aim at investigating the relationship between balance and query success probability, and how the threshold T_r affects both.

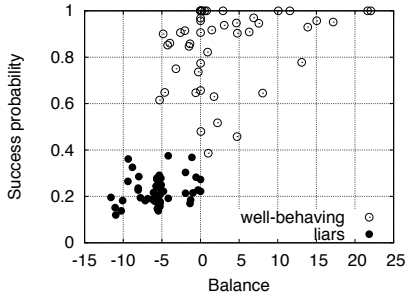
Table 1 presents, for each of the four classes of content, the success probability of a query issued by an Agent requesting that content as well as the success probability experienced by an Agent that advertises that content and provides it upon Broker's request. The results refer to three different settings of the request threshold, namely $T_r = -1, -5, -20$. Recall that

TABLE 1: Query success probability for different content classes and values of T_r

Class	Requester success probability			Provider success probability		
	$T_r = -20$	$T_r = -5$	$T_r = -1$	$T_r = -20$	$T_r = -5$	$T_r = -1$
1	0.910	0.801	0.431	0.937	0.935	0.882
2	0.912	0.820	0.522	0.935	0.922	0.661
3	0.959	0.826	0.532	0.919	0.899	0.441
4	0.966	0.925	0.655	0.904	0.695	0.260



(a)



(b)

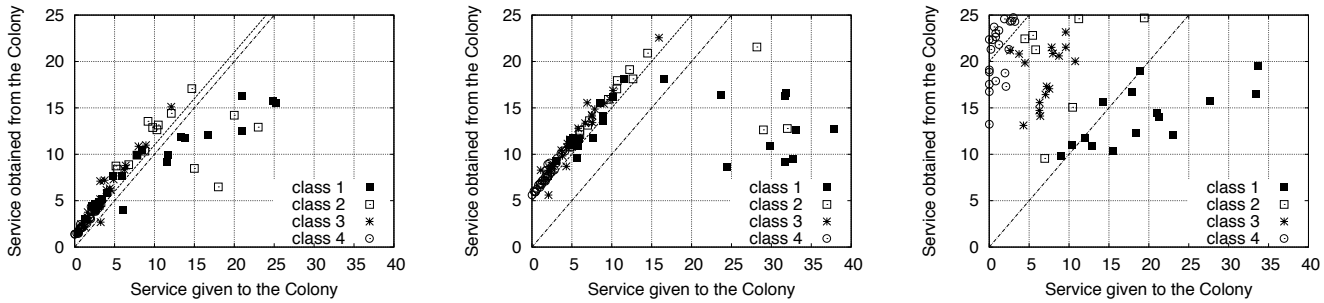
Fig. 4: Resilience to lying Agents: (a) success probability for well-behaving and liars, $T_r = -20, -5$; (b) success probability vs. balance for well-behaving and liars, $T_r = -5$.

the higher the content value (i.e., its popularity level and/or size), the higher the number of credits needed to request the content. It follows that the lower the T_r, the more likely it is that an Agent has enough credits to request a content, even if highly valuable, hence the higher its success probability. Consistently, providing a content that is either popular or large-sized, results in a higher gain, hence in better performance for the Agent storing that content.

Next, Fig. 5 shows how changing T_r impacts on the relationship between the amount of service (expressed in credits) that Agents provide and obtain from the system. The different markers denote Agents that provide content belonging to different classes. From a fairness viewpoint, we make the following observations. First, in each plot, points lying on the bisectrix $y = x$ correspond to Agents enjoying as much service as the amount they give to the Colony, while points above and below the bisectrix represent Agents that, respectively, obtain and provide more than what they should. Secondly, we would like all Agents to experience the same quality of service, i.e., they can access the same amount of content, independently of what they store.

Looking at the figure, we note that the closer T_r to 0, the more points lie on the bisectrix. However, considering the results in Table 1, it is clear that query success probability and fairness are diverging objectives and that properly selecting T_r helps in establishing a tradeoff between the two trends. Specifically, $T_r = -5$ appears to be a good choice, as it both provides high success probability and ensures that each Agent receives about the same amount of service it obtains from the system. However, the content sharing system by itself cannot solve the second issue related to fairness: as shown by Fig. 5, Agents storing low-value items (e.g., class 4) both provide and enjoy little amount of service, with respect to Agents offering more valuable items.

It is clear that additional capabilities are to be bestowed on Figaro, in order to restore fairness. In the following two sections, we will discuss how to exploit the cooperation inherent in a Colony of the Figaro system, by introducing Agent caching. We will then return to the above scenario in order to assess its benefit.



(a) (b) (c)

Fig. 5: Amount of service (in credits) given and obtained by Agents providing different classes of content, when (a) $T_r = -1$, (b) $T_r = -5$ and (c) $T_r = -20$.

8 EXPLOITING CACHING CAPABILITIES

To lessen the effect noted in Fig. 5 and increase the content availability in the system (i.e., the query success probability), we enhance our match-making paradigm by letting Agents have caching capabilities, i.e., the possibility to store content items they are not directly interested in, with the sole purpose of helping the Colony (and get a reward for that). They may use such capabilities following the Broker's directions. In other words, some of the Agents that obtain a content can be asked by the Broker to retain it in their cache. Those Agents will then be able to provide the cached content to others.

Below, we formulate our problem and devise a solution that promotes caching Colony-wise. For clarity, we presently leave disruptors and other attackers out of the picture.

8.1 Problem formulation

Without loss of generality, we assume that if an Agent has caching capabilities, its cache size is equal to σ. We focus on Colony k and denote by $R_k(j,t)$ the number of Agents in the Colony that advertised content j when they registered, and by $L_k(j,t)$ the number of Agents caching content j at time t within the Colony, because asked to do so by the Broker. Note that the latter did not advertise content j during registration, but acquired it through a query. Since the two behaviors are mutually exclusive, $P_k(j,t) = R_k(j,t) + L_k(j,t)$.

Let us first compute the query success probability conditioned to the fact that an Agent issuing the query has enough budget to request a desired content. By restricting our attention to well-behaving Agents, this is given by the joint probability of the following events: (i) neither SREQ nor SREP are lost; (ii) in the Colony there is at least one Agent under coverage advertising the requested content. Since it can be assumed that these events are independent and the number of Agents advertising a content does not vary during an SREQ/SREP exchange, the success probability of the query generated by the generic Agent, for content j in Colony k, is given by

$$S_k(j,t) = (1-q)^2 \left[1 - (1-\rho)^{R_k(j,t)+L_k(j,t)}\right]. \quad (6)$$

In (6), we assumed that SREQ and SREP transfers fail with equal probability q, while `ping` packets (which are very short) are always successfully delivered; ρ is the probability that a generic Agent in the network is under the network coverage.

The following Lemma shows that increasing $S_k(j,t)$ corresponds to increasing the number of Agents providing content j in Colony k.

Lemma 1. *The expression in (6) increases monotonically as* $R_k(j,t) + L_k(j,t)$ *increases.*

Proof: The proof can be found in the Appendix. □

However, just increasing the number of Agents providing any content leads to a waste of caching resources. Our caching strategy, instead, needs to adapt the number of copies of the content to the query rate associated to it. To this end, we resort to the content burden metric introduced in (1) and define $A_k(t)$ as the number of Agents in the Colony that have caching capabilities (i.e., they can cache some content according to the Broker's directions). Assuming that $A_k(t)$ is known by the Broker and that content popularity is negligibly affected by the change in number of providers due to caching, we are in a position to formulate our goal as minimizing the largest content burden, or, equivalently, as

$$\max \min_j \frac{1}{B_k(j,t)} \quad (7)$$

$$s.t. \quad L_k(j,t) \leq A_k(t) - R_k(j,t) \quad \forall j \quad (7.1)$$

$$\sum_{j=1}^{I} s(j) L_k(j,t) \leq \sigma \cdot A_k(t) \quad (7.2)$$

$$L_k(j,t) \in \mathbf{N} \quad \forall j \quad (7.3)$$

Note that such a formulation is an ILP (Integer Linear Programming) problem with I decision variables ($L_k(j,t)$, $j = 1, \ldots, I$). Constraint (7.1) forces the number of cached copies for content j to be not greater than the number of caching-capable Agents not advertising that content, while Constraint (7.2) ensures that the total number of cached items does not exceed the cache capacity available in the system. Constraint (7.3) forces the decision variables to take non-negative integer values. Unfortunately, a polynomial or pseudo-polynomial time solution to the above problem does not exist [23]. Additionally, the system dynamics require the problem to be solved every time Agents enter/leave the system, or they are banned/unbanned ($R_k(j,t)$ changes), or if the content of an Agent's cache is modified ($L_k(j,t)$ changes). Thus, we devise a heuristic to handle the problem solution, and evaluate its performance in Sec. 9.1.

8.2 A heuristic caching strategy

Since Brokers are in a good position to classify content based on its rarity and popularity (knowing number of providers and content query rate), it is at the Broker that our heuristic is implemented. Additionally, Brokers can use SREP messages to inform Agents that they should add to their cache a content they are retrieving. In turn, Agents can inform the Broker via feedback packets whether they followed its directions and which content, if any, they discarded to make room for the new content. Brokers can thus integrate their knowledge of the number $R_k(j,t)$ of initial providers advertising the content, with that of the number $L_k(j,t)$ of caching Agents.

The Broker considers that a content j is worth to be cached (hereinafter said to be *cacheworthy*) if the content burden, $B_k(j,t)$, outweighs the average value (computed over all content items available in the Colony) by a factor $\phi > 1$. As an Agent in the Colony issues a query for a cacheworthy content, the Broker asks the Agent to cache it if its balance is smaller or equal to the average. Also, it returns to the Agent the burden of the content items, to provide a discard priority for different items if the cache overflows (i.e., content associated to lower burden is more likely to be discarded).

From the Agent's viewpoint, we define the benefit/cost ratio of providing a content j as $\frac{G_k(j,t)}{c(j)} = \frac{g+B_k(j,t)}{K}$. This metric represents how much an Agent's balance increases per unit of effort (i.e., for a unitary amount of transferred data). Then, we prove that rational Agents will follow the Broker's suggestion to cache a content, whenever such a content is cacheworthy.

Theorem 3. *Given a rational Agent currently storing the set of content items $\mathcal{S}$, the Agent always finds it convenient to cache a new content w, upon Broker's suggestion.*

Proof: The proof can be found in the Appendix. □

Based on the above theorem, we conclude that the proposed heuristic can be successfully implemented: rational Agents will follow the Broker's suggestions, as it allows them to become providers of a content with higher benefit/cost ratio and, thus, increase their expected reward.

9 Evaluating Figaro with caching

We now evaluate the effectiveness of Figaro's caching mechanism and compare the performance of Figaro against other existing solutions. Except for the caching capabilities at the Agents, the results below have been obtained for the same network scenario and system parameters introduced in Sec. 7.

9.1 Benefits of caching

We now consider that a certain percentage of Agents have caching capabilities, and, for clarity of presentation, that there are no disruptors. We want to address the following questions:

(i) is caching effective in improving Figaro's performance?

(ii) how many caching-capable Agents are needed for caching to work?

(iii) how does caching impact on the balance distribution in the Colony?

To this end, we set $T_r = -5$, $\sigma = 200$ kB, and $\phi = 1.5$, and show the performance of the proposed heuristic caching strategy as the percentage of caching-capable Agents varies.

Figs. 6a and 6b show, respectively, the query success probability and the number of copies cached in the system, for the different content classes. As expected, as the number of caching-capable Agents increases, the success probability increases as well. However, a query for highly popular, large-sized content (class 1) has lower chances to succeed than others, unless all Agents can cache additional content items. Indeed, less requested or smaller items are less valuable (in terms of credits), hence they are easier to obtain. Nevertheless, Figaro significantly reduces the query success probability gap between the different content classes, already with 40% caching-capable Agents. Interestingly, Fig. 6b shows that what matters for a content to be cacheworthy is mostly its popularity level: most of the cached content items are the popular ones (i.e., classes 1 and 2). However, as the popular items become widely available in the system, room can be devoted to less requested items, especially the large-sized ones (i.e., class 3).

Next, we fix the percentage of caching-capable Agents to 50% and present in Fig. 6c the balance cumulative distribution function (CDF), for caching-capable and not caching-capable Agents. Caching-capable Agents have a significantly higher balance, due to the higher burden (hence value) of the content they provide. Also, notice that with extremely high probability caching-capable Agents have a balance greater than T_r, showing that caching has also the positive effect of distributing more evenly the load among Agents.

This is confirmed by Fig. 7a: caching improves not only the query success probability, but also the system fairness. In contrast to Fig. 5b, the amount of service provided and obtained by Agents does not depend any longer on the class of the content Agents originally advertised: *all* Agents receive from the Colony a level of service that is close to the one they provide *and*, very likely, they have a balance greater than the threshold T_r. In other words, caching is an effective way to achieve *both* a very high success probability and fairness guarantees among the Agents.

Next, for the different content classes, we compare the number of copies cached in the system as obtained through our heuristic (implemented in simulation), with the solution to the optimization problem in (7). The latter is computed by assuming that: (i) conditions are stationary (i.e., Agents are static with probability $\rho = 0.75$ to be under coverage, which is in agreement with the network scenario under study), and (ii) the Broker has knowledge of the number of caching-capable Agents as well as of the status of all Agent caches. The agreement between the results, shown in Fig. 7b, proves the good performance of Figaro, even compared to the case where global knowledge is assumed at the Broker.

Finally, caching also has the positive effect of reducing the usage of the backbone. Indeed, caching increases the content availability inside the Colony and, consequently, reduces the need to search for it outside (i.e., asking the Proxy). Fig. 7c confirms this statement, and suggests that, when used, the backbone is mainly employed to retrieve popular content. This is of particular importance when the backbone is not wired but implemented with cellular technologies such as 3G. In those cases, reducing its usage results in significant monetary savings, and may represent a strong motivation to deploy a peer-to-peer content discovery system like Figaro.

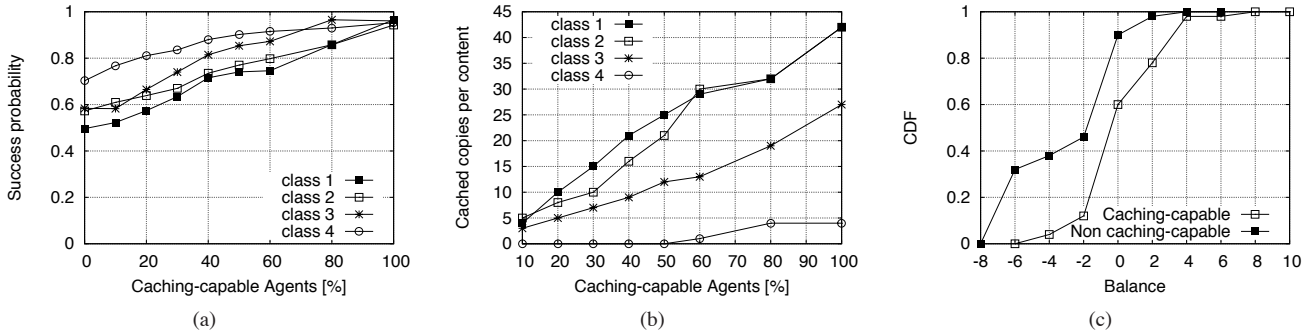
9.2 Benchmarking against other schemes

We finally evaluate Figaro by comparing it with other schemes in terms of query success probability and overhead. In Figaro, we assume that 50% of the Agents have caching capabilities. Since the results above showed that Figaro with caching greatly mitigates the differences in performance among Agents advertising content items with different characteristics, we now show results averaged over the different content classes.

Figaro is compared with a simple content-retrieval mechanism, referred to as Flat Flooding, and the BubbleStorm scheme, adapted to our wireless scenario from its wireline version [17].

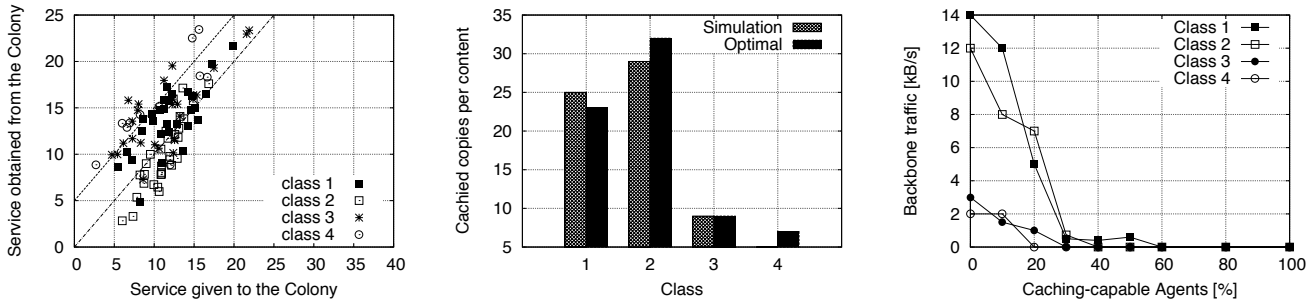
Flat Flooding hinges on a flat peer-to-peer exchange in ad hoc mode connectivity (i.e., without infrastructure). The query propagation range is spatially limited by a Time To Live value (set to 10 hops), and the rebroadcasting of already solved queries is avoided by means of a query lag time (set to 1 s): if an Agent detects responses to a query it has just received, it refrains from forwarding it any further. In this scenario, Agents implement a routing protocol for ad hoc networks (we chose OLSR) and act as relay nodes when needed.

As mentioned in Sec. 2, BubbleStorm [17] is based on a probabilistic exhaustive search paradigm in a overlay network. In our scenario, given the topological constraints, the random graph structure used by BubbleStorm to propagate a content



(a) (b) (c)

Fig. 6: (a) Query success probability and (b) number of cached copies for the different content classes and as the percentage of Agents with caching capability varies; (c) CDF of the balance for Agents able/unable to cache.



(a) (b) (c)

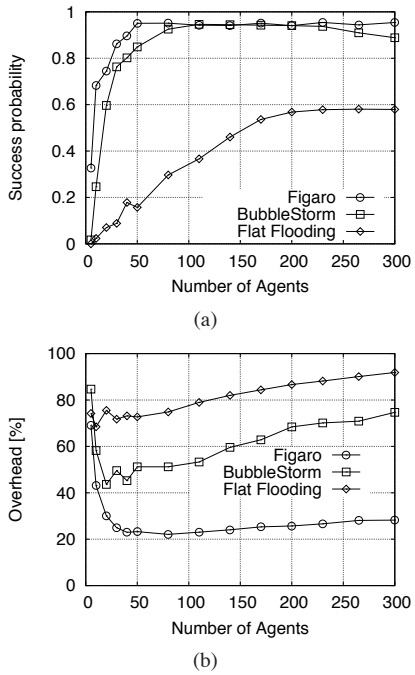
Fig. 7: (a) Service given and obtained by Agents and (b) heuristic caching strategy vs. optimal solution, for 50% caching-capable Agents; (c) Backbone usage as the percentage of caching-capable Agents varies.

is built as a subset of the tree already provided by the infrastructure. Also, our implementation of BubbleStorm provides for nodes to replicate, broadcast and cache content in such a way that the average fraction of Agents storing a content in BubbleStorm is equal to the one set up for Figaro.

While deriving the results, in Figaro Agents are assumed to be rational, i.e., to cooperate only if it is beneficial to them; in the case of the other schemes, instead, an ideal behavior is assumed, i.e., Agents are always willing to cooperate.

The query success probability is reported in Fig. 8a, as the number of Agents varies. Figaro outperforms the other solutions, especially when the number of Agents is low. While the comparison against Flat Flooding (which does not exploit any infrastructure) is not surprising, the improvement with respect to BubbleStorm is less obvious. Indeed, in the wireless scenario under study, BubbleStorm performs slightly worse than its wireline version, whose success probability exceeds 0.99. Figaro, instead, is more suitable for wireless, dynamic scenarios due to the match-making capabilities of the Brokers. The role of the Broker in arbitrating content sharing within a Colony also explains why the performance of Figaro does not deteriorate as the number of Agents grows, as instead happens when BubbleStorm's statistical matching is used.

An area where Figaro provides performance unmatched by BubbleStorm is message overhead. Fig. 8b shows that Figaro exhibits an overhead that is nearly inversely proportional to the query success probability, as a higher success probability implies that very likely (i) a query is satisfied within the



(a)

(b)

Fig. 8: (a) Success probability averaged over the different items and (b) message overhead, as functions of the number of Agents.

Colony where it has been generated and (ii) fewer SREQs are issued. In particular, it is shown that the Figaro overhead stabilizes below 30% of the total traffic. BubbleStorm instead exhibits a significantly higher overhead, due to its proactive

content (and query) propagation over the network. Finally, with Flat Flooding the overhead increases as the number of Agents grows, due to the increased network congestion: large numbers of Agents trigger an overwhelming number of replies to a single query.

We stress that a low overhead implies a low energy consumption: this, along with the very limited amount of status information that Agents are required to store and process, makes Figaro particularly suitable for energy-limited, handheld devices.

10 Conclusions

We presented Figaro, a match-making content discovery solution for wireless networks with infrastructure. In Figaro, mobile users (a.k.a. Agents) provide and request content items, while fixed Brokers help Agents in identifying who owns a desired content. A balance system ensures that Agents are treated in a fair way. Contributing to the effectiveness of Figaro is its feedback mechanism, that allows Brokers to zero in on free riders as well as attackers, and ban them to limit their negative impact. Also, Figaro complements its design with a caching scheme in which the Brokers suggest to the Agents what content to cache, in order to increase the query success probability (global and Agent's) and ensure fairness among Agents. Finally, we proved that it is rationally convenient for Agents to cooperate and to follow the Broker's caching advice. Simulation results showed the resilience of Figaro against different types of attacks, as well as its effectiveness, also with respect to other existing solutions.

References

[1] J. Wortham, "Customers Angered as iPhones Overload AT&T," *The New York Times*, September 2009, http://www.nytimes.com/2009/09/03/technology/companies/03att.html.

[2] L. Buttyan, J.-P. Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks," *Mobile Networks and Applications*, vol. 8, pp. 579-592, 2003.

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

[4] V. Srinivasan, P. Nuggehalli, C.-F. Chiasserini, R. Rao, "An Analytical Approach to the Study of Cooperation in Wireless Ad Hoc Networks," *IEEE Trans. on wireless and communications*, vol. 4, no. 2, 2005.

[5] N. Salem, L. Buttyan, J. Hubaux, M. Jakobsson, "Node Cooperation in Hybrid Ad Hoc Networks," *IEEE Trans. on Mobile Computing*, vol. 5, no. 4, April 2006.

[6] Y. Zhang, W. Lou, Y. Fang, "A Secure Incentive Protocol for Mobile Ad Hoc Networks," *ACM Wireless Networks*, vol. 13, no. 5, pp. 569–582, October 2007.

[7] M. E. Mahmoud, X. Shen, "Stimulating Cooperation in Multi-hop Wireless Networks Using Cheating Detection System," *IEEE INFOCOM*, 2010.

[8] P. Michiardi, R. Molva, "Core: a Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks," *Conference on Communications and Multimedia Security*, 2002.

[9] J. Jaramillo, R. Srikant, "DARWIN: Distributed and Adaptive Reputation Mechanism for Wireless Networks," *ACM MobiCom*, 2007.

[10] L. Xie, S. Zhu, "Message Dropping Attacks in Overlay Networks: Attack Detection and Attacker Identification," *ACM Trans. on Information and System Security*, vol. 11, no. 3, March 2008.

[11] M. X. Goemans, *et al.*, "Market Sharing Games Applied to Content Distribution in Ad-Hoc Networks," *ACM MobiHoc*, 2004.

[12] U. Lee, J. Lee, J.S. Park, M. Gerla, "FleaNet: A Virtual Market Place on Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 1, 2007.

[13] Y. Zhang, J. Zhao, G. Cao, "Roadcast: A Popularity Aware Content Sharing Scheme in VANETs," *ACM Mobile Computing and Communications Review*, vol. 13, no. 4, 2010.

[14] A. Nandan, S. Tewari, S. Das, M. Gerla, L. Kleinrock, "AdTorrent: Delivering Location Cognizant Advertisements to Car Networks," IEEE WONS, Les Menuires, France, January 2006.

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

[16] F. R. Dogar, A. Phanishayee, H. Pucha, O. Ruwase, D. G. Andersen, "Ditto: a System for Opportunistic Caching in Multi-hop Wireless Networks," *ACM MobiCom*, 2008.

[17] W. W. Terpstra, J. Kangasharju, C. Leng, A. P. Buchmann, "BubbleStorm: Resilient, Probabilistic, and Exhaustive Peer-to-Peer Search," *ACM SIGCOMM*, 2007.

[18] F. Malandrino, C. Casetti, C.-F. Chiasserini, "Content Discovery and Caching in Mobile Networks," *IEEE PIMRC*, Istanbul, Turkey, Sept. 2010.

[19] R. Ramanathan, J. Redi, C. Santivanez, D. Wiggins, S. Polit, "Ad Hoc Networking With Directional Antennas: A Complete System Solution," IEEE JSAC, vol. 23, no. 2, March 2005.

[20] M. Felegyhazi, J.-P. Hubaux, "Game Theory in Wireless Networks: A Tutorial," LCA-REPORT-2006-002, 2006.

[21] M. D. Lillibridge, M. Adabi, K. Bharat, A. Broder, "Method for Selectively Restricting Access to Computer Systems," *Tech. Rep., US Patent 6,195,698,* 2001.

[22] M. Lacage, M. H. Manshaei, T. Turletti, "IEEE 802.11 Rate Adaptation: A Practical Approach," *ACM MSWiM,* 2004.

[23] L. A. Wolsey, *Integer Programming,* Wiley, San Francisco, 1998.

PLACE PHOTO HERE

Francesco Malandrino graduated (summa cum laude) in Computer Engineering from Politecnico di Torino in 2008. Since 2009, he has been with the Telecommunication Networks Group at Dipartimento di Elettronica of Politecnico di Torino as a Ph.D. student. From 2010 till 2011, he has been a visiting researcher at the University of California at Irvine. His interests focus on wireless and vehicular networks and infrastructure management.

PLACE PHOTO HERE

Claudio Casetti (M'05) graduated in Electrical Engineering from Politecnico di Torino in 1992 and received his PhD in Electronic Engineering from the same institution in 1997. He is an Assistant Professor at the Dipartimento di Elettronica of Politecnico di Torino. He has coauthored more than 130 journal and conference papers in the fields of networking and holds three patents. His interests focus on ad hoc wireless networks and vehicular networks.

PLACE PHOTO HERE

Carla-Fabiana Chiasserini (M'98, SM'09) received the Electrical Engineering degree (summa cum laude) from the University of Florence in 1996. She received her Ph.D. from Politecnico di Torino in 2000. He is an Associate Professor at the Dipartimento di Elettronica of Politecnico di Torino. Dr. Chiasserini has published about 200 journal and conference papers, and she serves as Associated Editor for several IEEE journals.