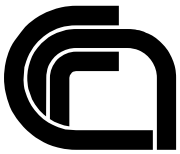


Consiglio Nazionale delle Ricerche

Minimum-Delay Service Provisioning in Opportunistic Networks

A. Passarella, M. Kumar, M. Conti, E. Borgia

IIT TR-19/2010

Technical report

Maggio 2010

Istituto di Informatica e Telematica

Minimum-Delay Service Provisioning in Opportunistic Networks

Andrea Passarella, Mohan Kumar, Marco Conti and Eleonora Borgia

Abstract

Opportunistic networks are (ad hoc) networks created dynamically by exploiting contacts between pairs of mobile devices that come within communication range. This networking paradigm overcomes main limitations of conventional MANETs, related to the fact that, due to mobility and energy conservation issues, it is often not practical to maintain connected multi-hop paths among nodes. While forwarding in opportunistic networking has been explored, investigations into asynchronous service provisioning are unique contributions of this paper. Mobile devices are typically heterogeneous, possess disparate physical resources, and can provide a variety of services. During opportunistic contacts, the pairing peers can cooperatively provide (avail of) its (other peer's) services. This service provisioning paradigm is a key feature of the emerging opportunistic computing paradigm. We develop an analytical model to study the behaviors of service seeking nodes (seekers) and service providing nodes (providers) that spawn and execute service requests, respectively. The model considers the case in which seekers can spawn parallel executions on multiple providers for any given request, and determines: i) the delays at different stages of service provisioning; and ii) the optimal number of parallel executions that minimizes the expected execution time without saturating providers' resources. The analytical model is validated through simulations, and exploited to investigate the performance of service provisioning over a wide range of parameters.

Index Terms

Opportunistic networks, service provisioning, performance evaluation, analytical modelling

1 Introduction

Opportunistic networks are mobile ad hoc networks in which the existence of simultaneous end-to-end paths between a sender and a receiver is not assumed. An

- *A. Passarella, M. Conti and E. Borgia are with IIT-CNR, Pisa I-56124, Italy, e-mail:{a.passarella,m.conti,e.borgia}@iit.cnr.it*
- *M. Kumar is with The University of Texas at Arlington, Arlington, TX 76019, USA, email: mkumar@uta.edu*

opportunistic contact occurs when pairs of mobile devices come within communication range of each other. An opportunistic network is created when several such opportunistic contacts occur between pairs of devices, distributed in time and space [1], [2]. As opposed to traditional MANETs, opportunistic networks are better equipped to deal with prolonged and frequent disconnections and partitions, as they exploit mobility as an opportunity rather than a challenge. Thanks to these features, opportunistic networks are capable of supporting delay tolerant applications in mobile and pervasive environments.

Opportunistic networks are the underlying enabler for the emerging concept of *opportunistic computing* [3]. The key observation of opportunistic computing is that the environment around (mobile) users features a steadily (if not exponentially) increasing set of heterogeneous resources available on fixed and mobile devices with wireless networking capabilities. In this context, resources include heterogeneous hardware components, software processes, multimedia content, sensors and sensory data. While not all resources can be available on any single device, they can be collectively available to anyone through the deployment of effective middleware in such a pervasive networking environment. Opportunistic computing is a natural paradigm that allows users to avail of those resources, by exploiting the fact that couple of devices come in contact with each other due to mobility, and can perform opportunistic computing actions (such as invoking resources of each other or retrieve content). In general, exploiting opportunistic contacts among nodes, complements existing wireless infrastructure including cellular networks and WiMAX. Sometimes, they actually represent the most natural option, as the required resource can be available on devices nearby, and hence an opportunistic pair-wise communication may simply be the most efficient way to access it. Essentially, we envision opportunistic computing and networking as evolutions of the distributed computing and networking paradigms in pervasive environments characterized by intermittent communication patterns.

In the envisioned opportunistic computing environment, resources can be abstracted, as services that can be shared and executed (perhaps remotely). A key research challenge for realizing the vision of opportunistic computing is therefore a comprehensive investigation of opportunistic service provisioning. This is a new topic in the area of opportunistic networks, as researchers in the past have explored

opportunistic networking mainly as a means for forwarding message packets. In this paper we start investigating this new challenging topic.

Service discovery, provisioning and composition in pervasive environments have been investigated by many researchers in the past, and there have been several successful implementations (see Section 2). In all past research on this topic, the existence of traditional, wireless, mobile ad hoc, or a combination of networks has been assumed. To the best of our knowledge, this is the first attempt at service provisioning using only opportunistic contacts.

Service provisioning in the framework of opportunistic computing enables interesting applications that have already started to emerge in the research landscape. The MetroSense project at Dartmouth College (http://metrosense.cs.dartmouth.edu/) investigates the concept of people-centric sensing [4]. They consider an environment similar to the one featuring opportunistic computing, and exploit mobile devices' resources and sensory readings to infer users activities and provide improved mobile social networking services to facilitate social interactions among users. Other application areas for opportunistic computing include pervasive healthcare, where opportunistic computing platforms can be used to implement continuous monitoring and filtering of patient data; intelligent transportation systems, where opportunistic computing paradigms can be used for exploiting vehicle-to-vehicle and vehicle-to-roadside devices communications to provide infomobility services; crisis management, where opportunistic computing paradigms can support quick establishment of networking and computing services in emergency scenarios when the primary infrastructures might not be available (e.g., because disrupted) or too congested. More details and examples related to these areas can be found in [3], [5], [6], [7], [8], [9], [10], [11], [12], [13].

As described in Section 3, we consider a very simple architecture in which nodes can opportunistically request service executions to directly encountered peers, and collect results *the next time* peers are encountered after the execution (on those peers) has been completed. Hereafter, we denote by *request* a request for the execution of a given service, and by *results* the output results of the service execution. The main focus of this paper is on optimizing the way in which a seeker (a node requiring a service) should spawn executions onto encountered providers. Specifically, for each request, a seeker might spawn multiple (parallel) executions to different

providers. Clearly, the time taken to receive the results will be the minimum of the times taken to get results from each provider. However, uncontrolled replication of parallel executions might lead to saturating providers' computational resources. As, in opportunistic networks, services are provisioned by resource limited devices, this is a very important issue to be considered. To investigate the performance of service provisioning in opportunistic networks, in Sections 4 and 5 a model is developed to depict the behavior of the service provisioning system as a function of the number of replicas spawned by seekers[1]. The model accounts for the effect of varying number of seekers, providers, seekers' request loads, providers' computational capabilities, and users' mobility parameters. The model determines the optimal number of parallel executions that a seeker should spawn in order to minimize the expected service completion time, given the computational capabilities of the providers. The model is validated against simulation results, showing very good accuracy, and is then used to analyze the behavior of the service provisioning system with respect to a set of key parameters (Section 6), spawning much larger parameters' ranges with respect to what is possible by simulations.

Results of our studies show that a policy using the optimal replication level indicated by our model is able to greatly outperform both a policy that greedily replicates requests on all encountered providers, and a policy that generates just a single execution on the first encountered provider (i.e., policies that work without any background information). We show that, for a wide range of scenarios, the optimal policy adjusts the replication level so as to always achieve the minimum expected service time. Unlike the other policies, the optimal policy is also able to significantly limit the increase of the expected service time when the load on the system increases even by orders of magnitude.

2 Related work

Service-oriented architectures have been investigated in the area of ubiquitous and pervasive computing. The heterogeneous devices in a pervasive environment possess varied resources and capabilities. In order to utilize the collective capabilities of all available resources, service oriented architectures have been developed [14]. The

1. Hereafter, the terms "replicas", "replications", and "parallel executions" are used interchangeably.

dynamic nature of pervasive systems makes service provisioning a challenging task. Many solutions available for service provisioning are essentially extensions to service discovery that introduce transparency between the user and the services. In [15] Ravi et al., develop a mechanism for accessing pervasive services across the network of devices using cell phones. Much effort has been devoted by the research community to service composition in pervasive networking environment. Service composition entails stitching together two or more basic services to create composite application level services. Essentially, there are two types of service composition mechanisms - static and dynamic. In static mechanisms, a template with place holders represents the user request/task as the input to service composition while the output is a plan containing services identified to populate the place holders within the request template. Examples of such service composition mechanisms in pervasive systems include [16], [17], [18]. For dynamic service composition the description of the service and the mechanisms employed to identify matches between a requested service and an available one are critical to the successful operation. In [19] Kalasapur et al., describe services using graph theoretic techniques to facilitate the migration of services across devices and networks and to compose high level application services. Our work differentiates from this body of research as none of these articles has focused on opportunistic networking environments. All of them either consider single-hop wireless access to the infrastructure, or assume conventional stable connectivity among nodes as in the MANET paradigm. Being the networking environment that different and so much more challenging, we limit ourselves - for now - to a very simple service provisioning scheme, in which seekers can just avail of services directly provided by encountered providers.

Service provisioning in opportunistic networks is a challenging problem that, to the best of our knowledge, has not been addressed yet. In a broad sense, opportunism is the key design feature in this kind of networks, as contacts should be opportunistically exploited according to the individual users' goals (be it sending messages across the network, or avail of a service not available locally). Research on opportunistic networks has mainly focused on routing issues (e.g. [20], [21], [22], [23], [24], [25], [26], [27]), mobility analysis (e.g. [28], [29], [30], [31], [32]) and, more recently, on data-centric architectures for content delivery (e.g., [33], [34]). To the best of our knowledge, the problem of service provisioning during opportunistic contacts

has not been investigated in the past.

The work presented in this paper can be seen as one of the building blocks of the recently proposed concepts of opportunistic computing [3] and people-centric sensing [4], [5], [6], [7]. The latter area already produced significant results, which however are mainly focused on inferring people activities through sensors available on single pervasive user devices (such as last-generation smartphones), and how to exploit such information for augmented mobile social networking services. The process necessary to perform such inferences is mainly executed on single devices, by exploiting sensory data available on the device itself. The work presented in this paper starts investigating opportunistic service provisioning between pervasive devices, and is thus complementary.

3 Service provisioning in opportunistic networks

Overall, the service provisioning model we support is as follows. A service execution entails different phases. Upon a contact between any two nodes, the nodes exchange information about the services they provide and wish to avail, decide whether to spawn a new service execution on the encountered peer, and download results for previously spawned service executions, if any. We assume that the seeker and provider may or may not be connected when the service is executed. We also assume that the mobility model is such that the probability of seeker and provider to meet again after the completion of the service, if they were disconnected during the execution, asymptotically tends to 1 in the limit $t \rightarrow \infty$.

From a system design standpoint, we assume that nodes in the network run the following protocol. They keep a Service Index *SI* that contains service information: available services (e.g., $S_a = [S1, S3, S7]$), services needed (e.g., $S_n = [S2, S8]$), services whose execution is locally ongoing (pending services e.g., $S_p = [S4]$) and services completed (e.g., $S_c = [S3]$). These data are used to manage service execution, as follows. During a contact time, first of all, if a pending service of one device has been completed by the other, the results are transferred. Then, if services needed on one device are available on the other, the service inputs *may* be transferred, according to a local *replication policy* run by the seeker. The algorithm in Figure 1 describes this process ($n_a \Leftrightarrow n_b$ denotes the fact that nodes n_a and n_b are in contact and exchange information). Note that from a networking standpoint this algorithm just requires

Algorithm for Node n_a
while $n_a \Leftrightarrow n_b$ **do**
  $n_a(SI) \leftrightarrow n_b(SI)$ ▷ assuming n_a and n_b trust each other
  if $n_a(S_p)$ **in** $n_b(S_c)$ **then**
    n_b sends output(S_c) $\rightarrow n_a$
  end if
  if $n_b(S_p)$ **in** $n_a(S_c)$ **then**
    n_b sends output(S_c) $\rightarrow n_b$
  end if
  if $n_a(S_n)$ **in** $n_b(S_a)$ **then**
    n_a sends input(S_n) $\rightarrow n_b$
  end if
  if $n_b(S_p)$ **in** $n_a(S_a)$ **then**
    n_b sends input(S_n) $\rightarrow n_a$
  end if
end while

Fig. 1. Algorithm for service provisioning during contacts.

reliable single-hop communications between nodes, that we assume available. It is also assumed that the two nodes trust each other and exchange each other's information. In future work, we will address the issue of trust and authentication related to opportunistic contacts.

A key part of the above protocol is the replication policy, that is used by each seeker to decide whether or not to spawn a service execution to an encountered provider. As mentioned in Section 1, uncontrolled replication may saturate providers resources and should, in general, be avoided. By following the replication policy, each seeker can decide how many parallel executions to spawn in order to minimize the service completion time without saturating providers resources. Addressing this issue is the main goal of this paper, and we address it through the analytical model described in Sections 4 and 5. The model actually tackles a more general issue than this, as it describes the time required by a seeker to receive results as a function of the number of spawned replicas. The model allows us to highlight the impact of a number of parameters characterizing the system and the networking environment, on the system's performance. Possible directions to further generalize the model are discussed in Section 7.

4 Model for optimal service provisioning

The analytical model presented hereafter focuses on a *tagged* seeker (wishing to avail of a given service), and assumes that the seeker is allowed to spawn parallel exe-

cutions of the service on different providers. We assume that all the seekers behave according to the same stochastic processes (i.e., the tagged seeker is statistically representative for a generic seeker). We provide a model of the expected service time (i.e., the time the seeker has to wait to receive the results), as a function of the number of allowed parallel executions. To better clarify the rationale of studying this scenario, we introduce the following terminology.

(Service) Request: a request for a service, generated at a seeker.

(Service) Execution: an execution of a request, spawned by a seeker on a provider; note that, following a request, a seeker may spawn multiple parallel executions on different providers.

(Service) Replicas: the set of parallel executions related to the same request.

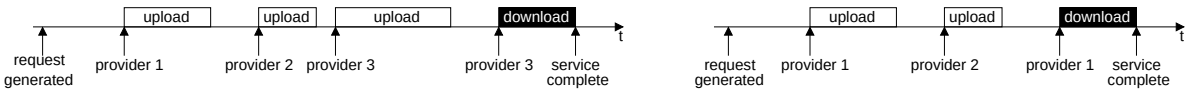
Execution Time: the time elapsed from when a request is generated at the seeker, and the time when the seeker receives the results from a particular provider on which it has spawned an execution.

Service Time: the time elapsed from when a request is generated at the seeker, and the time when the seeker receives the results from *any* provider on which it has spawned an execution. Note that the service time is the minimum of the execution times of the providers on which a request execution is spawned.

Intuitively, spawning more executions can result in a lower service time, as the service time is the minimum over the execution times of the replicas. However, on the other hand, spawning more executions also increases the computational load on the providers, and results in longer execution times at individual providers. In the remainder of the section we describe a model to determine the *optimal number of replicas* representing the optimal trade-off between the two effects described above. Analytical results of the model are presented in Section 5.

4.1 Replication behavior

In our model, the (tagged) seeker manages each request as described below (refer to Figure 2 for an example). Let us denote with m the maximum number of parallel executions the seeker can generate for each request ($m = 3$ in the Figure). The seeker opportunistically uploads the input parameters to all encountered providers, until either of the following occurs: i) the seeker completes m uploads, thus generating m



(a) all replicas are spawned (b) a subset of replicas are spawned

Fig. 2. Scenarios for service replication.

replicas (case in Figure 2(a)); or ii) the seeker downloads the results from a provider before generating all the m replicas (case in Figure 2(b)). Specifically, the latter case occurs when the seeker downloads the results from a provider that already completed the service execution before uploading the input parameters to all the m distinct providers. The model provides the optimal value (m_{opt}) of the m parameter, the optimal number of maximum allowed replications. Focusing on the maximum number of allowed replications is the correct way of approaching the problem, as the actual number of replications may change between each request (see Figure 2), and is thus not a controllable parameter. The download of the output results works similarly to the upload phase, i.e., service results are opportunistically downloaded when the seeker meets any provider on which an execution was spawned. In general, both uploads and downloads may take one or more contact events to complete. In this case, on the next contact, the upload (download) resumes from when it was stopped.

We assume that the requests generation pattern at the seeker is independent of the encounter pattern between the seeker and the providers. This implies that a new request may be generated before the previous one has been served (i.e., before the seeker receives the results, or even before it finished to replicate the previous request's executions). Therefore, when the seeker meets a provider, it may upload input parameters for several (successive) requests, all generated at the seeker itself. Similarly, it is also possible that the seeker downloads, from an encountered provider, several output results, related to different requests generated at the seeker and whose executions has be spawned at that provider. In both cases we assume request uploads and downloads are served according to a FIFO policy.

4.2 General Model

To model the replication behavior described in Section 4.1 we consider the scheme in Figure 3. For the reader's convenience, Table 4.2 summarizes all the symbols used throughout the model.

TABLE 1
List of symbols

R_i	execution time at the i-th provider
$R(m)$	service time when m replicas are allowed
B_i (D_i, θ_i)	delay of the first (second, third) stage of pipe i
m (m_{opt})	maximum (optimal) number of allowed replicas
p_i	probability that i executions are spawned for a given request $1 \leq i \leq m$
m^*	average number of spawned executions when m replicas are allowed
λ	request rate each seeker
λ_i	total offered load at provider i
μ_i	service rate of provider i
N	number of nodes in the network
$p^{(s)}$ ($p^{(p)}$)	probability of being a seeker (provider)
$k = p^{(s)}N$	average number of seekers
$M = p^{(p)}N$	average number of seekers
$p_s(\mathcal{A})$	success probability, i.e., probability of meeting a provider in the set $\mathcal{A}$
$T(\mathcal{A})$	time to meet any provider in the set $\mathcal{A}$ from a random point in time
$L(\mathcal{A})$	time to meet any provider in $\mathcal{A}$ after the end of a contact
Et	average inter-contact time between any two nodes
Ec	average contact time between any two nodes

We assume that the tagged seeker issues requests according to a Poisson process with rate λ. Although we focus on a tagged seeker, more seekers can be present. To simplify the presentation, without loss of generality, we assume that each seeker issues requests at the same rate λ, and denote with $p^{(s)}$ the probability that a node in the network is a seeker. The case in which the requests rate is different for each seeker is developed in [35]. If N is the number of nodes in the network $k \triangleq p^{(s)}N$ is the average number of seekers. Similarly, $p^{(p)}$ is the probability that a node is a provider of the sought service, and $M \triangleq p^{(p)}N$ is the average number of providers.

Each horizontal pipe in Figure 3 represents a different execution (on a different provider) for a request issued by the tagged seeker[2]. Therefore, there are m pipes, as a *maximum* of m executions can be spawned. As will be clear in the following, the model also accounts for the cases in which less than m executions are actually spawned. Specifically, we derive closed form expressions for p_i, which denotes the probability that a request is completed (i.e., the seeker receives the output results) after spawning i executions ($i = 1, \ldots, m$). Accordingly, we will also consider in the model the average number of spawned replicas (also referred to as effective replication level), defined as $m^* = \sum_{i=1}^{m} ip_i$.

Let us now consider the individual execution times. For each request, pipe i

2. As each pipe corresponds to a particular provider, we use these terms interchangeably

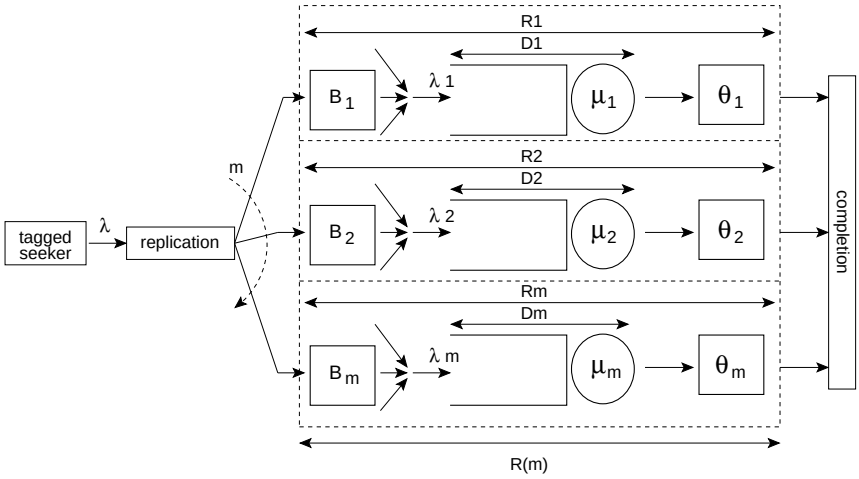


Fig. 3. General scheme of the replication process.

corresponds to the i-th provider starting the computations, i.e., pipes are ordered in increasing temporal order of computation starting time. Each pipe consists of three stages. The first stage represents the time required to complete the i-th upload of the input parameters, which is throughout referred to as B_i. The second stage represents the time required to execute the request after it is spawned. The second stage thus models the computation at the i-th provider, and can be represented by a queue with service rate μ_i, as discussed in detail in Section 5. Note that more seekers can spawn requests on each provider. Thus, the providers offered load (λ_i in the Figure) is the result of the joint request pattern of all the seekers. We will model it in detail in Section 5.3. Finally, the third stage represents the time required by the seeker to complete the download of the output results from the provider "corresponding" to pipe i (i.e., the i-th provider to complete the upload of input parameters during the first stage), and is denoted by θ_i. Based on these definitions, the delay on the i-th pipe is given by

$$R_i = B_i + D_i + \theta_i \qquad i = 1, \dots m. \tag{1}$$

The model firstly provides closed form expressions for the expected execution time at each provider (ER_i). Specifically, according to equation (1), one can derive the expressions for the expected delays of the three stages, EB_i, ED_i, $E\theta_i$, respectively. This is carried out in Sections 5.2, 5.3, and 5.4. Then, in Section 5.5 the expected service time (denoted by $ER(m)$) is derived, as a function of m, by considering all the possible completion cases (see Figure 2). Specifically, the model for $ER(m)$ accounts for the possibility that less than m executions be generated, i.e., that some of

the horizontal pipes are not used. The optimal number of allowed replicas, denoted by m_{opt} is the value of m that minimizes $ER(m)$, i.e., $m_{opt} = \arg\min_{1 \leq m \leq M}\{ER(m)\}$.

5 Analytical results

Before presenting the detailed analytical results, it is worth stating the assumptions under which they are derived.

A1. The tagged seeker encounters any *specific* provider at the same rate at which it encounters any given node in the network. Therefore, the probability that any given contact occurs with any specific node (and, as a special case, with any specific provider) is $1/N$. Furthermore, the encounter process between the seeker and any specific provider is memoryless with respect to the sequence of previously encountered nodes, contact and inter-contact times. We consider the same assumptions also for the contact process between a tagged provider and a specific seeker.

A2. A contact time is always sufficient to upload the input parameters (and to download output parameters) for all the pending requests between the seeker and the provider. Furthermore, while a node is engaged in a contact, it cannot communicate with any other node in its transmission range.

A3. Executions on providers start after the end of the contact time used to upload the input parameters.

A4. Inter-contact (contact) times are independent and identically distributed (iid). Inter-contact and contact times are mutually independent.

A5. Both inter-contact and contact times between pairs of nodes are exponentially distributed.

Assumptions A1, A2 and A3 are necessary for making the analysis tractable, and could be released at the cost of a drastic increase of the model's complexity. In particular, assumption A2 is equivalent to assuming infinite bandwidth during any contact, which is a common assumption in the literature on opportunistic networks. This assumption could also be dropped, and load on the shared medium can be considered. However, this comes at the cost of a huge increase in the analysis complexity, as is shown in [35]. Note that even with a mere 11Mbps WiFi link (whose net transport-level throughput is notoriously about 5 Mbps), in the scenario

we consider in the analysis (in which the contacts last for about 15s), about 10MB can be exchanged on average. Therefore, for a large range of cases, assuming infinite bandwidth is a pretty reasonable approximation, and we prefer to keep it, to make the analysis more readable.

Assumption A5 is necessary to derive the analytical model, and is a typical assumption in the field of opportunistic networking analysis (e.g., [29], [30], [21]). Assumption A5 is reported here for avoiding even simple inaccuracies in the mathematical derivations. However, the only point in the analysis where this is necessary is to derive a minor component of the expected time to meet providers described in Theorem 1 (see Appendix A). Therefore, even if this assumption is dropped, the resulting inaccuracy is very limited. Furthermore, the debate whether inter-contact times in opportunistic networks are exponential or not is still not completely settled (see, e.g., the different conclusions drawn by [28] and [36]. Recently, the work in [37] has analyzed some of the largest available traces (MIT Reality) and ran a χ^2 test to verify the hypothesis of inter-contact times being exponentially distributed, finding very good statistical support for this hypothesis.

The rest of the analysis proceeds as follows. In general, $R(m)$ should be derived as the minimum over m generic random variables (r.v.) R_i, of which we are able to characterize the distributions and compute closed form expressions for the expectations (as shown in the following sections). However, to have tractable analytical expressions, we compute $R(m)$ as the minimum over a set of *exponentially* distributed, independent r.v., i.e. we assume that the r.v. R_i are exponential with expected value ER_i. We assess the approximation level of this choice through validation with simulation results in Section 6.1.

As a preliminary step, in the next section we model the time required by a tagged seeker to encountered a provider on which a request execution can be spawned. This will be a key building block for computing the delay of the three stages in the following sections.

5.1 Time to meet providers

The main result we derive in this section is the analytical expression of the average time required by a tagged seeker to encounter a provider in a given set $\mathcal{A}$, where $\mathcal{A}$ can be any subset of the set of providers, possibly coinciding with the whole set. In

the following of the analysis we need to distinguish whether such time intervals are measured from a random point in time, or from the end of a contact time. Therefore, we provide the expressions for both figures. Specifically, the following theorem holds.

Theorem 1: If contact and inter-contact times are exponentially distributed, iid[3], and mutually independent, the average time required by a tagged seeker to meet any provider in a set $\mathcal{A}$ starting from a random point in time ($ET(\mathcal{A})$), and starting after the end of a contact time ($EL(\mathcal{A})$) are, respectively

$$ET(\mathcal{A}) = \frac{Ec + Et}{p_s(\mathcal{A})} - Ec\left(1 + \frac{Et}{Ec + Et}\right) \tag{2}$$

$$EL(\mathcal{A}) = \frac{Ec + Et}{p_s(\mathcal{A})} - Ec, \tag{3}$$

where Ec and Et are the average contact and inter-contact times, and $p_s(\mathcal{A})$ is the probability that a generic node encountered by the seeker belongs to $\mathcal{A}$.

Proof: See Appendix A. □

The expressions of ET and EL are quite intuitive. Specifically, EL is basically made up of the average value of a geometric number of intervals with average value $Ec + Et$, and success probability $p_s(\mathcal{A})$. The corrective term Ec in equation (3) accounts for the fact that the time to meet a provider should not include the time of the contact between the seeker and the provider. ET is basically similar to EL, with a correcting term accounting for the fact that T starts at a random point in time instead of after the end of a contact. Finally, note that, exploiting assumption A1, it is easy to show that $p_s(\mathcal{A})$ just depends on the number of providers in the set (and not on the particular providers), and is equal to $\frac{|\mathcal{A}|}{N}$.

5.2 Delay of the first stages

The closed form expression of the delay of the first stage of pipe i (B_i) is provided in Lemma 1. While we provide the formal proof in Appendix B, we hereafter describe the main line of reasoning of the proof. Any new request is generated at a random point in time with respect to the underlying mobility process. The first execution is thus spawned, on average, after $ET(\mathcal{A})$ seconds, where $\mathcal{A}$ is the whole set of

3. More specifically, we mean that contact times are iid, and inter-contact times are iid. Contact times can be clearly distributed differently from inter-contact times.

the providers (as we have assumed that the probability of meeting each provider is the same, we can replace the indication of $\mathcal{A}$ with its cardinality, and use the notation $ET(|\mathcal{A}|)$ and $EL(|\mathcal{A}|)$). According to assumption A3, the first replica is then spawned at the end of the following contact time with the encountered provider. After spawning each replica, in order to spawn the next one, the seeker has to meet a *new* provider on which a replica of that request has not already been spawned. This occurs, on average, after a time interval equal to $EL(M-k)$ where k is the number of replicas already spawned. Therefore, the execution on the i-th provider starts after a time interval whose average value is $ET(M) + iEc + \sum_{k=1}^{i-1} EL(M-k)$. The closed form expression in Lemma 1 follows after routine manipulations.

Lemma 1: The average delay of the first stage on pipe i ($i = 1, \ldots, m$) can be evaluated as follows:

$$EB_i \simeq \frac{Ec+Et}{p^{(p)}} - \frac{EcEt}{Ec+Et} + N\,(Ec+Et)\ln\frac{M-1}{M-i}, \quad i < M \tag{4}$$

$$EB_M \simeq \frac{Ec+Et}{p^{(p)}} - \frac{EcEt}{Ec+Et} + N\,(Ec+Et)\left[\gamma + \ln(M-1)\right].$$

where γ is the Euler constant ($\gamma \simeq 0.577$), and $M = p^{(p)}N$.

Proof: See Appendix B. □

Note that equation (4) requires to estimate the number of nodes in the network N (or, more in general any two out of the triplet $p^{(p)}, N, M$). When this is not practical, Lemma 2 provides an approximated form, which is precise as long as $\frac{i}{2M}$ is close to 0 (see Appendix B).

Lemma 2: The average delay of the first stage on pipe i ($i = 1, \ldots, m$) can be approximated as follows:

$$EB_i \simeq i \cdot \frac{Ec+Et}{p^{(p)}} - \frac{EcEt}{Et+Ec}. \tag{5}$$

Proof: See Appendix B. □

Note that the approximation of EB_i provided in equation (5) is correct when $\frac{i}{M}$ is small, and predicts a linear increase of the delay with i (i.e., the provider one considers). When i/M is small, the number of providers available to spawn the i-th replica is approximately equal to M. In order to spawn the i-th replica, a seeker has

to find i providers, which requires, approximately, i times the time required to find any provider among the available M, i.e. $i \cdot \frac{Et+Ec}{M/N}$. Also equation (4) predicts that EB_i linearly increases with i when i/M is small. However, it provides precise results also when i/M is not small. Intuitively, in the latter region, EB_i increases more than linearly with i, as the set of available providers becomes smaller and smaller.

5.3 Delay of the second stages

As shown in Figure 3 we model the computation process at each provider with a queue. The service time of the queue represents the computation time at the provider's CPU. To simplify the analysis, and without loss of generality, we assume that the computation time at any provider is exponentially distributed with average value $1/\mu$, i.e., we assume that computation times at various providers are iid (the case in which the rates of the computation times' distributions are different on different providers is considered in [35]). Note that the fact of considering a random computation time in the model allows us to account for: i) the variability of different providers' CPUs; and ii) the different computations required by different requests for a given service (e.g., because the input parameters are different). Note however that, as the computation times are assumed iid, the delay on the second stage of a given pipe i does not depend on the particular provider corresponding to pipe i.

According to assumption A2, when a seeker meets a provider it spawns all the replicas it can execute on that provider. This can be nicely captured by modelling the computation process as a batch arrival system (see, e.g., [38]). Focusing on any particular provider, a batch (of requests) arrives whenever the provider meets a seeker. The size of the batch is the number of requests generated for that provider by the encountered seeker since the last time the two nodes came into contact. As we show in Appendix C, in our model batches arrive according to a Poisson process, the batch sizes are iid, and the computation time of each request is also an independent r.v. As none of the involved stochastic variables depends on the particular provider one consider, we can model the second stage of each pipe as an $M^{[X]}/M/1$ system (following the notation used in [38], where X is the r.v. denoting the size of the batch). Accordingly, Lemma 3 provides a closed form expression for the average delay of the second stages. For the reader's convenience, we recall that λ is the rate of the request generation (Poisson) process at each seeker, $p^{(s)}$ and $p^{(p)}$ are the probability of

each node being a seeker and a provider, respectively, m^* is the average number of replicas actually generated for each request (when the maximum number of allowed replicas is m), Ec and Et are the average contact and inter-contact times, respectively.

Lemma 3: The average delay of the second stages is

$$ED = \frac{1}{\mu - \frac{\lambda p^{(s)} m^*}{p^{(p)}}} + \frac{2\frac{\lambda m^*}{p^{(p)}}(Ec + Et) + 1}{2\left(\mu - \frac{\lambda p^{(s)} m^*}{p^{(p)}}\right)}. \tag{6}$$

Furthermore, the utilisation of the providers is $\rho = \frac{\lambda p^{(s)} m^*}{\mu p^{(p)}}$, and thus the providers are not saturated as long as the following equation holds

$$\frac{\lambda p^{(s)} m^*}{p^{(p)}} < \mu. \tag{7}$$

Proof: See Appendix C. □

Note that, as shown in Appendix C the term $\frac{\lambda p^{(s)} m^*}{p^{(p)}}$ represents the total load on each provider which, as expected, increases with the number of actual replicas (m^*), the requests generation rate (λ) and the probability of nodes being seekers ($p^{(s)}$), while decreases with the probability of nodes being providers ($p^{(p)}$). It is also interesting to note that, while the load does not depend on the characteristic of the inter-contact process, the expected delay of the second stages linearly increases with the average inter-contact time. This is a side effect of the fact that the expected delay of a batch arrival system depends on statistics of the batch size, which increase with the expected inter-arrival times between batches (see [38] for the details).

Also note that equation (7) already provides a limitation on the actual number of replicas that each seeker should spawn. This confirms our initial intuition that replicating executions can be an advantage just up to a certain point, as replicating too aggressively (e.g., as in a greedy policy) saturates the providers. Note that hereafter we define the saturation threshold as $m_c \triangleq \frac{\mu p^{(p)}}{\lambda p^{(s)}}$, and the saturated region as the range of m values such that the corresponding value of m^* is greater than m_c.

5.4 Delay of the third stages

Lemma 4 provides a closed form expression for the average delay of the third stage. We discuss its derivation directly in the proof, as it is straightforward. Note that we

drop the indication of the particular pipe from the notation $E\theta_i$, as the average value does not depend on the pipe.

Lemma 4: The average delay of the third stages is

$$E\theta = ET(1) + Ec, \tag{8}$$

where $ET(1)$ is the average time required to meet any given provider starting from a random point in time.

Proof: Under assumptions A2 and A3 the time required to download a request from a given provider is the time required to meet that provider, plus a contact time. Furthermore, as a tagged seeker meets any provider with the same probability (according to assumption A1), the delay of the third stages of the pipes are identically distributed, and thus the average value does not depend on the pipe. Deriving the expression in equation (8) is straightforward by noticing that the delay of the third stage starts when the execution of the request has been finished by the provider. In the case of exponentially distributed contact and inter-contact times[4], the time to meet the sought provider is therefore the time required to meet any given node starting from a random point in time, i.e., $ET(1)$. □

5.5 Optimal replication

In the previous sections we have characterized the expected *execution* time of requests, i.e. the average time for the tagged seeker to receive the results *form each particular provider*. In this section we model the expected *service* time, i.e. the average time for the tagged seeker to receive the results from *any* provider, and show how to select m_{opt}, the optimal number of maximum allowed replications.

Although the seeker is allowed to spawn m replicas for each request, it may well happen that it receives results before spawning all requests, as already discussed in Section 4.1. Specifically, we denote with $p_i(m^*)$ the probability that the seeker spawn only i replicas, $i = 1, \ldots, m$, and with $EH_i(m^*)$ the expected service time under the condition that just i replicas are spawned (the dependence on m^* is explained immediately in the following of this section). The expected service time when a

4. As already mentioned when discussing Assumption A5, in the case of non exponential distributions, Equation (8) is a very good approximation, as the inaccuracy is only on a small initial component of the estimated time interval (details are presented in Appendix A.

maximum of m replicas are allowed can thus be computed as

$$ER(m) = \sum_{i=1}^{m} p_i(m^*) \cdot EH_i(m^*) \tag{9}$$

In order to expand $ER(m)$ we need the expressions of EH_i, p_i and m^*, which are provided by Lemma 5 and 6, and Theorem 2. The formal proof of the Lemma 6 and Theorem 2 are provided in Appendix D. We provide here only the proof Lemma 5, as it is helpful to clarify some aspects of the general line of reasoning we use also for the other proofs, as well as the dependence of EH_i and p_i on m^*.

Lemma 5: The expected service time when just i replicas are spawned ($i = 1, \ldots, m$) can be computed as follows:

$$EH_i(m^*) \simeq \frac{1}{\sum_{j=1}^{i} \beta_j(m^*)} \ , \tag{10}$$

where $\beta_j(m^*)$ is equal to $\frac{1}{EB_j + ED(m^*) + E\theta}$, and EB_j, $ED(m^*)$ and $E\theta$ are as in equation (4), (6), and (8), respectively.

Proof: H_i is the minimum over the execution times of the first i providers, i.e., $H_i = \min_{j=1,\ldots,i}\{R_j\}$. As discussed in Section 4, to keep the analysis tractable we assume that the r.v. R_j are independent and exponentially distributed with average $ER_j(m^*) = EB_j + ED(m^*) + E\theta$. It is well known that the minimum over a set of independent exponential r.v. is also exponential with rate equal to the sum of the rates of the individual r.v. (see, e.g., [39]). By defining $\beta_i(m^*)$ as the rate of the r.v. R_j (i.e., $\beta_j(m^*) = \frac{1}{EB_j + ED(m^*) + E\theta}$), the rate of H_i becomes equal to $\sum_{j=1}^{i} \beta_j(m^*)$, and its average value is as shown in equation (10). The dependence of EH_i on m^* comes from the fact that the rates β_j are functions of the average delay of the second stages ED, which is a function of m^*. □

Lemma 6: The probability that the tagged seeker receives the results after spawning exactly i executions ($i = 1, \ldots, m$) can be computed as follows:

$$\begin{aligned} p_1(m^*) &= \frac{\delta_1(m^*)}{\delta_1(m^*) + \psi_2} \\ p_i(m^*) &= \frac{\delta_i(m^*)}{\delta_i(m^*) + \psi_{i+1}} - \frac{\delta_{i-1}(m^*)}{\delta_{i-1}(m^*) + \psi_i} \\ p_m(m^*) &= \frac{\psi_m}{\delta_{m-1}(m^*) + \psi_m} \end{aligned} \tag{11}$$

where $\delta_i(m^*) \triangleq \frac{1}{EH_i(m^*)}$, and $\psi_i \triangleq \frac{1}{EB_i}$.

Proof: See Appendix D. □

The expressions of p_i can be found by noticing that the probability that i replicas are spawned is the probability that the seeker *has not* yet received the results when completing the upload of the input parameters on the i-th provider, but *has* received the results before completing the upload of the input parameters on the $(i+1)$-th provider.

Theorem 2: The average value of the number of executions actually spawned when m replicas are allowed (m^*) can be found by solving the following fixed point equation:

$$m^* = \sum_{i=1}^{m} i \cdot p_i(m^*) \tag{12}$$

where p_i are as in equations (11). Specifically, equation (12) admits either one or three solutions. In the former case, the unique solution is stable. If it falls in the saturated region, then $m^* = m$ (and is greater than the saturation threshold $m_c = \frac{\mu p^{(p)}}{\lambda p^{(s)}}$). Or, it can fall in the non-saturated region (i.e., $m^* < m_c$). When equation (12) admits three solutions, one of them falls in the saturated region, and is stable, while the other two fall in the non-saturated region, and one of them is stable.

Proof: See Appendix D. □

The form of equation (12) basically comes for the definition of m^*, which is the average value of the actual number of spawned replicas (i.e., $m^* = \sum_{i=1}^{m} i \cdot p_i$). However, as the probability of spawning a given number of replicas depend on m^*, m^* can only be found by solving the fixed point equation in (12). Note that Theorem 2 guarantees that a stable solution m^* can always be found with standard iterative methods that are guaranteed to converge in a finite number of steps. From a systems perspective, Theorem 2 tells that the only one case in which the system works in saturation is when the unique solution is equal to m (in this case m^* falls in the saturated region). In the other cases, the fact that the fixed point equation admits one stable solution (x_1) in the non-saturated region is very important. Although another stable solution ($x_2 = m$) may exist in the saturated region, seekers receive results for requests after spawning - on average - x_1 executions only, and experience a finite service time. Therefore, they never replicate executions so aggressively to make the system operate in the saturated condition corresponding to x_2.

The results in equations (10) and (11) allows us to find the numerical solution of m^* for any given value of m, and thus to find the corresponding value of the expected service time $ER(m)$. This also provides a way to find the optimal value for the maximum allowed number of replications, m_{opt}, as shown in Theorem 3.

Theorem 3: The optimal value for the maximum allowed number of replications is:

$$m_{opt} = \arg \min_{m=1,\dots,M} \left\{ \sum_{i=1}^{m} p_i(m^*) \cdot EH_i(m^*) \right\} \tag{13}$$

where $EH_i(m^*)$ and $p_i(m^*)$ are as in equations (10) and (11), and, for any value of m, m^* can be found by solving the following fixed point equation

$$m^* = \sum_{i=1}^{m} i \cdot p_i(m^*) \ .$$

Proof: This is straightforward from the definition of m_{opt}, and the results in Lemma 5, 6, and Theorem 2. □

We wish to highlight that Theorems 2 and 3 allows us to compute the optimal operating point for seekers, in order to minimize the expected time to receive results from providers. They are thus the main result of the model derived in this paper.

The model derived so far is very powerful to characterize the behavior of service invocation in opportunistic networks, as we show in details in Section 6. From a practical standpoint, the only complexity of the model is solving the fixed point equation in (12). This must be done for increasing values of m, until the expected service time $ER(m)$ starts increasing. As discussed in Section 6, this turns out much more efficient than using alternative evaluation tools (such as, for example, simulations), for a large range of relevant system's parameters.

6 Performance evaluation

In this section we analyze the performance gain that can be achieved by exploiting the results of our analytical model. Our main performance index is the expected service time. We compare three policies: i) in the *single* policy each seeker generates a single request on the first encountered provider; ii) in the *greedy* policy each seeker replicates each request on all providers it meets, until the request is satisfied; this corresponds to setting the maximum number of allowed replicas to M; iii) in the *optimal* policy

the maximum number of allowed replicas the one achieving the minimum expected service time. The idea is thus to highlight the advantage of using an intelligent policy based on our theoretical results with two straightforward policies that can be implemented without any background information.

In order to validate the analytical model, we also developed a simulation model based on the OMNeT++ simulator (http://www.omnetpp.org/). Simulation results are presented with confidence intervals computed with 95% confidence level. In the simulated environment the nodes move in a square, according to the RWP mobility model, with the modifications described in [40] to guarantee that the model's distributions are stationary (the nodes' average speed is 1.5 m/s, representative of walking speeds). This scenario also corresponds to mobility models more realistic than RWP such as recently proposed social-oriented mobility models (e.g., [31]), when users belong to the same social community. We control the density of the network, and, thus, the value of the contact and inter-contact times, either through the number of nodes, or the transmission range of each node, or the size of the simulation area (precise details are provided for each simulation scenario). In the simulation, seekers and providers are chosen at the beginning of each simulation run according to the $p^{(s)}$ and $p^{(p)}$ parameters. Seekers then generate requests according to a Poisson process with configurable rate. The maximum number of allowed replications is an input parameter for the simulator. In the simulations, we assume that, whenever a seeker meets a provider, it is able to upload the service parameters for all the requests' replica that the provider can possibly satisfy, i.e., our simulation model shares assumption A2 with the analytical model (recall that on overage, in our scenario, on any contact nodes can exchange about 10MB of data). Finally, request executions are queued at the providers according to a FIFO policy, and computation times are exponential with a configurable rate.

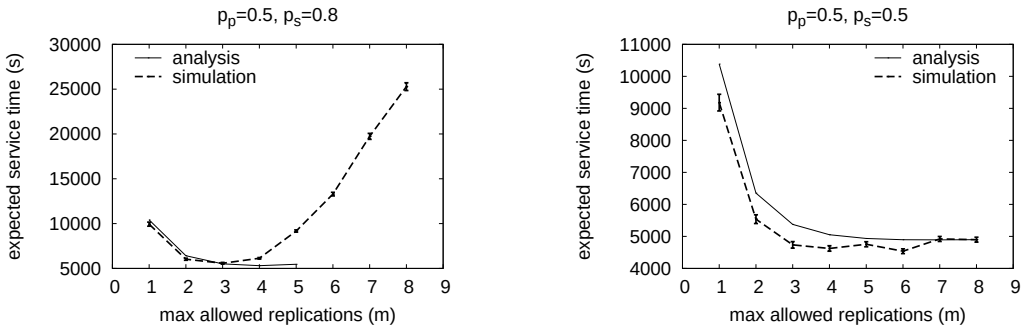
The main purpose of simulation results is validating the analytical model. Therefore, the main indices we measured are the optimal number of allowed replications (m_{opt}) and the optimal expected service time ($ER(m_{opt})$). The value of the allowed replications (m) is an input parameter of the simulation runs, and we repeated the simulation runs with increasing values of m. At the end of each run, we computed the corresponding average service rate (with 95% confidence intervals), and we identify the optimal case as the value of m achieving the minimum average service rate.

Results presented hereafter show a very good agreement between the simulation and the analysis, which allows us to conclude that the analytical model is accurate. It is worth noting that the analytical model provides a much more flexible tool than the simulation model. Specifically, the inherent complexity of the simulation model (mainly, the number of events that are generated) make it practically impossible to explore the system's behavior through simulation over a large range of key parameters. This is possible instead by using the analytical model. For example, in the following section we study the system behavior in dense networks, when the number of nodes increases up to 200, or heavy-load conditions in which services require long computation times. In these cases the simulation model becomes too complex (in terms of execution time and memory space) to study the system, while the analytical model allows us to completely investigate the system's behavior.

6.1 General properties

Before comparing the three policies in a number of scenarios, in this section we discuss important general properties of the system. To this end, we consider a scenario where the average computation time at providers is 30s (i.e., $\frac{1}{\mu} = 30$s), the request rate at seekers is about 1 request every 3 minutes (precisely, $\lambda = 0.005$ req/s), the probability of node being a provider ($p^{(p)}$) is 0.5. We show two representative cases, in which the probability of node being a seeker ($p^{(s)}$) is 0.5 and 0.8 respectively. In both cases the number of nodes is 20, the nodes' transmission range is 20m, and the simulation square is 1000x1000m large. The resulting scenario is very sparse, as nodes' average intercontact time is about 10 minues, while the average contact duration is about 16s.

Figure 4(a) allows us to highlight three important aspects. First of all, the shape of the curve $ER(m)$ clearly highlights the existence of a trade-off that manifests itself as the number of replicas increases. Increasing the number of allowed replicas allows seekers to run more executions in parallel, and pick the minimum execution time among the set of the replicas. On the other hand, it also increases the load at providers and brings the system closer to the saturation point (recall equation (7)). Note that the analytical curve stops when the system enters in the saturated region, as the expected service time is infinite in this case. In the simulations, after this point the service time is clearly not stationary anymore, and therefore we plot, in that region, the average



(a) $p^{(p)} = 0.5$, $p^{(s)} = 0.8$ (b) $p^{(p)} = 0.5$, $p^{(s)} = 0.5$

Fig. 4. Representative cases of the expected service time as a function of m. Both analytical (solid) and simulation (dashed) curves are shown.

over the service requests actually completed within the simulation time, to give an indication of the trend of the service time. Secondly, this plot also motivates why looking for the system's optimal operating point is important. Note that the optimal policy achieves a far lower expected service time with respect to both the single and the greedy policy. Finally, results show that the analytical model is able to capture the trend of the simulation results. According to the analytical model, the system saturates as soon as m is greater than 5, and the optimal operating point is achieved for $m = 4$. Simulation results confirm this trend with good accuracy.

Figure 4(b) shows a case in which the system never saturates. The plots confirms that the analytical model is able to capture the trends of the simulation results as far as the expected service time. Both the greedy and the optimal policies outperform the single policy. The optimal and the greedy policies result in almost equivalent expected service time, although the optimal point does *not* correspond to that of the greedy policy. The important point to note is that in such cases the optimal policy is anyway more efficient than the greedy policy, as it generates less executions, and thus avoids useless resource consumption. Note that in this case (which is representative of several configurations of the parameters we have tested) the plot of the expected service time flattens out after a certain value of m. This means that, in certain configurations, there is a large range of m values over which the service time varies very little. Jointly with the inherent statistical fluctuations of simulation results, which makes it difficult to compare results when they are close, this results in the fact that the simulation and analytical models may not always be in good agreement as far as the optimal value of m. This is not a matter of concern, because

TABLE 2
Default analysis parameters

$1/\mu$	30s
λ	0.005 req/s
N	20
tx_range	20m
Area	1000m x 1000m
avg speed	1.5m/s
$p^{(s)}$	0.1,0.2,0.5,0.8
$p^{(p)}$	0.1,0.2,0.5,0.8

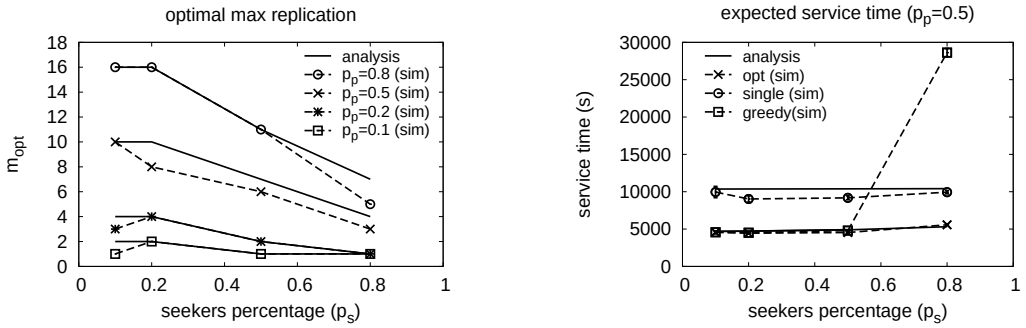
of the flat shape of the expected service time curves. As we clearly show in the following, the analytical model predicts with good accuracy the simulation results with respect to the expected service time.

In other cases, shown in details in the next sections, the optimal policy coincides either with the greedy policy (typically, when the system's load is low), or with the single policy (typically, when the system's load is high). In general, the results we present hereafter show that the optimal policy adapts to the system's configuration, converging to the greedy or the single policy when appropriate, and achieving lower expected service time when neither of them is the best policy.

6.2 Performance in sparse scenarios

We hereafter consider a sparse opportunistic network, in which nodes inter-contact times are about 10 minutes, while contact times last, on average, about 16s. The values used to obtain this scenario are shown in Table 2. This is the main scenario we consider in our performance evaluation, as it is typical of the opportunistic networking environment that we consider as our reference in this work.

Figure 5(a) shows the optimal value of the maximum allowed number of replicas for different values of both the seekers and the providers probability. For a small percentage of seekers, the optimal policy is greedy, i.e. m is equal to $M = p^{(p)}N$. This is because the load generated by such small number of clients is not enough to saturate the computational resources of the providers. However, as soon as the number of seekers increases beyond 20%, the optimal policy spawns parallel executions less aggressively than the greedy policy. Note that, in this scenario, the single policy (which spawns just 1 execution for each request) is optimal just when the number of seekers is high (beyond 50%), and the number of providers is low (below



(a) m_{opt} (b) $ER(m_{opt})$, $ER(1)$, $ER(M)$

Fig. 5. Performance of the three policies in sparse scenarios. For each configuration, both analytical (solid) and simulation (dashed) curves are presented. Analytical points are not plotted for saturated conditions.

20%). In this case, the number of providers is so low that even replicating requests more than once results in significant congestion. Finally, note that, as expected, for a given number of seekers, m_{opt} increases with the number of providers, as increasing the number of providers means increasing the overall computational capacity of the system, and thus shifting the saturation point towards higher replication levels.

An interesting remark from Figure 5(a) is the fact that a system using our model to identify the optimal policy is able to autonomically switch either to the greedy or the single policy when appropriate, or work in between these two extremes. Also in this case the analytical and simulation results show good agreement. It is indeed counter-intuitive that, in the simulation results, m_{opt} increases when moving from $p^{(s)} = 0.1$ to $p^{(s)} = 0.2$ when the provider probability is 0.1 and 0.2 (the two curves at the bottom). When the number of seekers increases, the system becomes more loaded, and thus the optimal number of allowed replication should not increase (which is, by the way, the behavior predicted by the analytical model). By looking at the simulation results, it clearly appears that the expected service time at $p^{(s)} = 0.1$ obtained at the indicated optimal point, and that obtained by the greedy policy are statistically equivalent, being both in the confidence interval of each other (and far apart by about 1%). Therefore, the fact that for $p^{(s)} = 0.1$ the optimal policy indicated by simulations does not correspond to the greedy policy is just an artefact of statistical fluctuations.

Figure 5(b) shows the expected service time for the three policies in the case of $p^{(p)} = 0.5$ (similar remarks can be done with respect to the other values of $p^{(p)}$, as well). The figure shows the simulation plots and the corresponding analytical plots

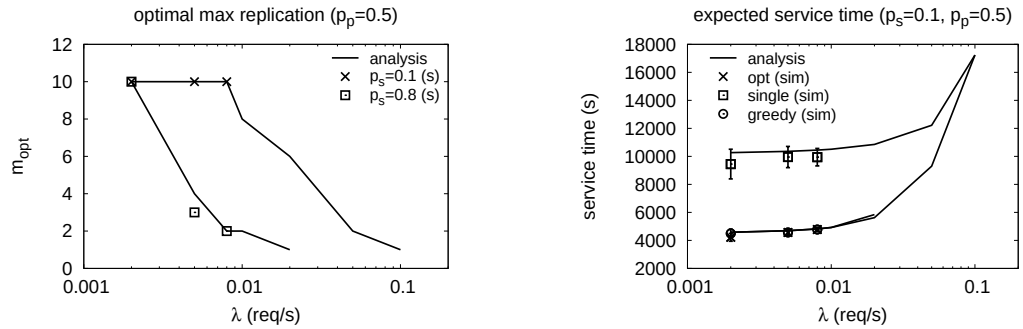
for all the three policies (single, greedy, optimal). Analytical plots are drawn with solid curves, while dashed curves represent simulation results. For the optimal policy, the analytical and simulation curves can hardly be distinguished, as they overlap. Furthermore, for the analytical plot of the greedy policy, no result is available for $p^{(s)} = 0.8$ as the system is in the saturated region. Before that point, also for the greedy policy the analytical and simulation plots overlap. Furthermore, as explained below, for $p^{(s)}$ less or equal to 0.5, the optimal and greedy policy coincide, and therefore the four corresponding plots also overlap.

As noted before, for a small number of seekers, the optimal policy coincides with the greedy policy. Actually, the curves of the greedy and optimal policies almost overlap up to $p^{(s)} = 0.5$ included, as the two policies provide almost the same expected service time in this range. However, when the number of seekers increases beyond this point, the greedy policy saturates the system (recall that in this case the expected service time in the analysis is infinite, and thus the analytical curve for the greedy policy stops at $p^{(s)} = 0.5$). On the contrary, beyond $p^{(s)} = 0.5$ the optimal policy significantly outperforms both the single and the greedy policy. Also note that the optimal policy results in just a slight increase of the expected service time as the number of seekers increases. By jointly looking at this plot and at the plot in Figure 5(a) one can see that, as the number of seekers increases, the optimal policy reduces the number of spawned execution, and *thanks to this*, limits the increase of the expected service time.

6.3 Performance with varying request and computation loads

In this section we study the system's sensitiveness with respect to the request load (λ), and the computation load of executions ($1/\mu$). Specifically, the cases λ=0.002, 0.005, 0.01, 0.02, 0.05, 0.1 req/s and $1/\mu$=15, 30, 60, 90s are considered. Figure 6(a) shows the optimal number of allowed replications for varying request loads when the providers probability is $p^{(p)} = 0.5$, and for the extreme cases of the seekers probability, $p^{(s)} = 0.1$ and 0.8. Figure 6(b) shows the expected service time of the optimal, single and greedy policies for $p^{(s)} = 0.1$ and $p^{(p)} = 0.5$. The reason why we have selected the smallest possible seekers probability will be explained immediately. All the other parameters are as in Table 2.

First of all, note that simulation results are presented for small values of λ only (up



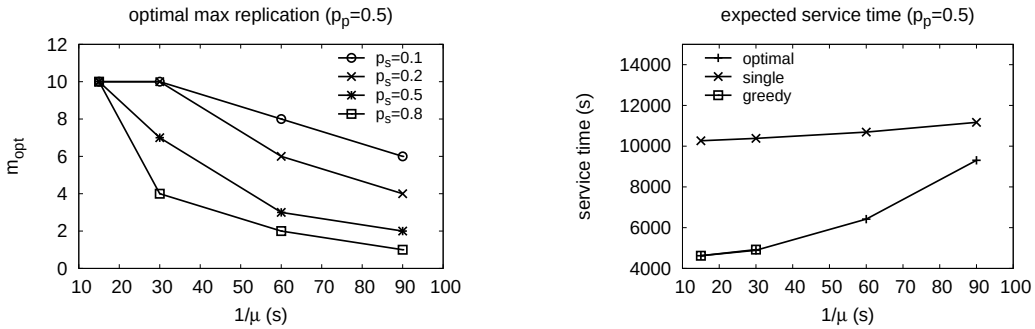
(a) m_{opt} (b) $ER(m_{opt})$, $ER(1)$, $ER(M)$

Fig. 6. Performance for varying request loads (λ).

to $\lambda = 0.008$), as, beyond this point, simulation results are practically unmanageable (due to the exponential increase of computation and memory requirements). Instead, the analytical model allows us to explore the system's behavior also for higher loads, and this shows very important features.

Let us analyze the case of $p^{(s)} = 0.1$, i.e. the top curve in Figure 6(a) and Figure 6(b). As expected, for light loads (up to $\lambda = 0.008$) the optimal and the greedy policies coincide (i.e., $m_{opt} = M = 10$). However, the greedy policy saturates as soon as λ increases beyond 0.002 (shown by the fact that the analytical curve for the greedy policy in Figure 6(b) stops at this point). Between $\lambda = 0.01$ and $\lambda = 0.1$ neither the greedy nor the single policy are optimal (i.e., $1 < m_{opt} < M = 10$), and the optimal policy significantly outperforms both in terms of expected service time. Finally, the optimal policy converges to the single policy for very high load (at $\lambda = 0.1$), i.e., m_{opt} becomes equal to 1. The analytical model also shows that in certain cases even the single policy saturates the system. For example, this is the case when the seekers percentage is high ($p^{(s)} = 0.8$), and the load increases beyond 0.02. In this region (see Figure 6(a)) there is basically no policy that can avoid the system's saturation. Indeed, the case of $p^{(s)} = 0.1$ is the only one in which at least the single policy can be used at all the considered loads, and that is why this case is used for Figure 6(b).

Another way of varying the load is through the average time required by providers to execute each request, $1/\mu$. Figure 7(a) shows the optimal number of allowed replications when the providers probability is 0.5, while Figure 7(b) shows the expected service time of the three policies when $p^{(p)}$ and $p^{(s)}$ are 0.5. Note that we use only the analytical results, as i) the analytical model has shown to be well in agreement with simulations, and ii) for most of the cases it is practically unmanageable to obtain



(a) m_{opt} (b) $ER(m_{opt})$, $ER(1)$, $ER(M)$

Fig. 7. Performance for varying average computation time ($1/\mu$).

statistically sound results from simulations.

Qualitatively, Figure 7 confirms the properties already highlighted before. In general, the optimal number of allowed replication decreases when either the computational load or the seekers probability increase (Figure 7(a)), until a point where the optimal policy coincides with the single (e.g., for $p^{(s)} = 0.8$ and $1/\mu = 90s$). As far as the expected service time (Figure 7(b), note that the greedy policy saturates quite soon (beyond 30s), and beyond this point the optimal policy outperforms either of the other two. Also note that when $1/\mu$ is 30s and $p^{(s)}$ and $p^{(p)}$ are 0.5, the greedy policy is *not* optimal (as shown by Figure 7(a)), but achieves almost the same expected service time of the optimal policy (as shown by Figure 7(b)). This is indeed the case already discussed in Figure 4(b).

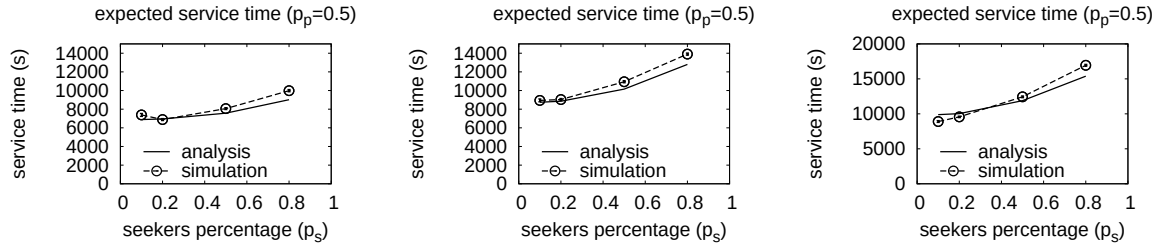
6.4 Performance with varying number of nodes

In this section we study the behavior of the system for an increasing number of nodes N to study scalability properties. In this part of the analysis we keep the density constant by scaling up the size of the simulation area with the number of nodes. Specifically, in addition to the case $N = 20$ analyzed in the previous sections, we consider the cases $N = 50, 80, 100$. As the density is constant, we still consider sparse scenarios. The performance in dense scenarios is investigated in Section 6.5.

Table 3 shows the optimal values of m (the maximum allowed replications) for the various cases considered. We will analyze the dependence of m on N in detail later on in this section. The results clearly confirm that in several configuration the analytical and simulation models are in good agreement, while in other cases

TABLE 3
m_{opt} as a function of N

$\mathbf{N = 20}$	$p^{(p)} = .1$		$p^{(p)} = .2$		$p^{(p)} = .5$		$p^{(p)} = .8$	
	an	sim	an	sim	an	sim	an	sim
$p^{(s)} = .1$	2	1	4	3	10	10	16	16
$p^{(s)} = .2$	2	2	4	4	10	8	16	16
$p^{(s)} = .5$	1	1	2	2	7	6	11	11
$p^{(s)} = .8$	1	1	1	1	4	3	7	5
$\mathbf{N = 50}$								
$p^{(s)} = .1$	5	5	10	6	22	13	34	12
$p^{(s)} = .2$	3	3	7	5	18	10	29	13
$p^{(s)} = .5$	1	1	2	2	7	5	14	7
$p^{(s)} = .8$	NaN	1	1	2	4	3	7	5
$\mathbf{N = 80}$								
$p^{(s)} = .1$	6	6	13	13	33	14	48	14
$p^{(s)} = .2$	3	3	7	5	22	14	37	17
$p^{(s)} = .5$	1	1	2	2	7	5	13	8
$p^{(s)} = .8$	NaN	1	1	2	4	4	7	6
$\mathbf{N = 100}$								
$p^{(s)} = .1$	7	4	15	9	36	16	57	18
$p^{(s)} = .2$	3	3	7	5	24	11	41	16
$p^{(s)} = .5$	1	1	2	2	11	5	13	8
$p^{(s)} = .8$	NaN	1	1	2	4	4	7	5



(a) $N = 50$ (b) $N = 80$ (c) $N = 100$

Fig. 8. Optimal service time for different values of N.

they are not. When discussing the example in Figure 4(b) we explained that such mismatches are due to the fact that, in certain configurations, the curve of the expected service time (as a function of m) flattens out after a certain point, and there is therefore a significant range of m values for which the expected service time is almost constant. To confirm that this is the case, it is sufficient to analyze the optimal service time. Specifically, Figure 8 shows the expected optimal service time according to the analytical and simulation models for the cases $N = 50$, 80 and 100, for $p^{(p)} = 0.5$. Other configurations provide similar results. For the sake of space,

these are reported in Appendix E.[5]. It is clear that the estimates of the simulation and analytical models are in good agreement, which confirms that the mismatch on the optimal values of m is not particularly significant. Finally, we have also checked that the analytical and the simulation models agree not only on the expected optimal service time, but also on the service time for the greedy and single policy, as in the case of $N = 20$ shown in Figure 5(b).

Having further discussed the related properties of the analytical and simulation models, we can analyze in more details the dependence of the optimal value of m and the optimal service time on N. Specifically, in Figure 9 we consider the case of $p^{(s)} = 0.5$. It is quite clear that the optimal number of maximum replications varies very little with N, while there seems to be a linear increase of the expected optimal service time with N.

The latter results can be actually confirmed with simple arguments by looking at the analytical model presented before. Recall that the delay on each pipe R_i is made up of the delay of three stages. The delay of the first stage (Equation 4) is composed by a part that is independent of N, and a part that is linear with N (although this dependence disappears when i/M is small, see Equation 5). The delay of the second stage (ED, Equation 6) can be shown to be basically independent of N. First of all, note that the only component that may depend on N is m^*, i.e. the average number of replicas actually generated for each request. The values of m^* in the optimal case are plotted in Figure 9(c), which shows just a sublinear increase with N when the number of providers is high. We can thus reasonably approximate ED as independent of N. Finally, the delay of the third stage ($E\theta$, Equation 8) depends on the expected time to meet a *specific* node (the provider corresponding to the pipe), $ET(1)$. By inspecting the expression of $ET(1)$ it is clear that it linearly depends on N. Summarizing, we can write the expected delay on pipe i as the sum of two components, one independent of N, and another one linear with N, i.e., $ER_i = EX_i + NEY_i$. Furthermore, it is easy to see that EY_i is dominated by the delay of the third stage, and can thus be approximated by $E\theta$. Assuming, again, that the r.v. R_i are exponential, and recalling that the optimal service time is the minimum over the delays of m_{opt} pipes, $R(m_{opt})$ is also an exponential r.v. with rate $\gamma_R = \sum_{j=1}^{m_{opt}} \frac{1}{EX_i+NE\theta}$. Finally, it can also be shown

5. Without loss of generality, we hereafter show results in representative configurations. The same insights can be derived also by looking at other configurations, which are reported in Appendix E.

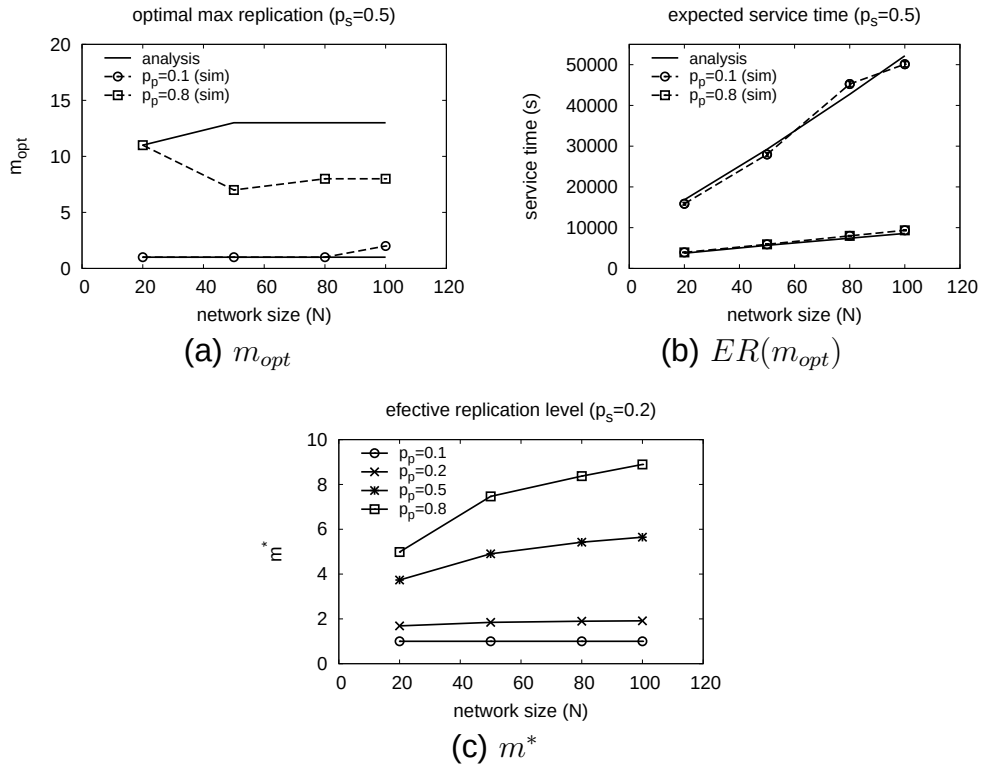


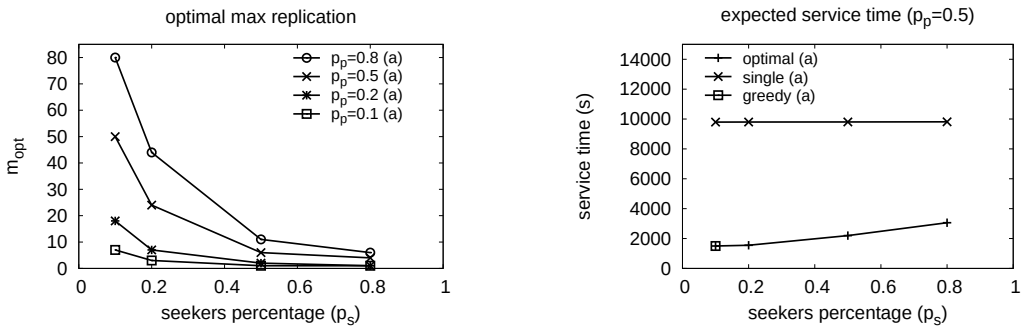
Fig. 9. optimal maximum replication (m_{opt}), optimal service time ($ER(m_{opt})$) and effective replication level (m^*) as functions of N.

that, unless when the system is extremely close to the saturation of the second stage, the factor $NE\theta$ dominates over EX_i. After simple manipulations, it can be found that the expected optimal service time is approximately equal to $E\theta\frac{N}{m_{opt}}$. This confirms the linear dependence of $ER(m_{opt})$ with N found in Figure 9(b). Furthermore, it also justifies the fact that the slope of the line decreases with $p^{(p)}$, because, for a given value of $p^{(s)}$, m_{opt} clearly increases with the average number of providers (i.e., with $p^{(p)}$).

6.5 Performance in dense scenarios

Although the main reference scenario of this work are sparse opportunistic networks, we show in this session how the system behaves when the network becomes dense. With respect to the set of parameters in Table 2, we increase the number of nodes to 200. This results in decreasing the average inter-contact time (Et) to about 35s, while the average contact time (Ec) is still in the order of 15s.

Figures 10 show the optimal number of allowed replicas and the expected service time (for $p^{(p)} = 0.5$). It is interesting to note that when the number of nodes increases,



(a) m_{opt} (b) $ER(m_{opt})$, $ER(1)$, $ER(M)$

Fig. 10. Performance in a dense scenario ($N = 200$).

the greedy policy is *never* optimal (i.e., $m_{opt} < M$), although it achieves an expected service time comparable with that of the optimal policy in very lightly loaded scenarios ($p^{(s)} = 0.1$ and $p^{(p)} = 0.5$), while for the rest of the configurations reported in Figure 10 the greedy policy brings the system in the saturated region. By recalling from equation (7) that the load on each provider is defined by $\frac{\lambda p^{(s)} m^*}{p^{(p)}}$, it is easy to see that in the case of 200 nodes the greedy policy generates a much higher load on each provider, and this results in overall congestion even for a small number of seekers. Finally, note that also in a dense scenario the optimal policy achieves far lower expected service time with respect to either the single and the greedy policy.

7 Future directions

In this section some generalizations of the service provisioning models proposed in the paper are firstly discussed. Then, we discuss some possible directions to exploit the analytical model in distributed algorithms driving the system close to the optimal configuration.

7.1 Generalizations

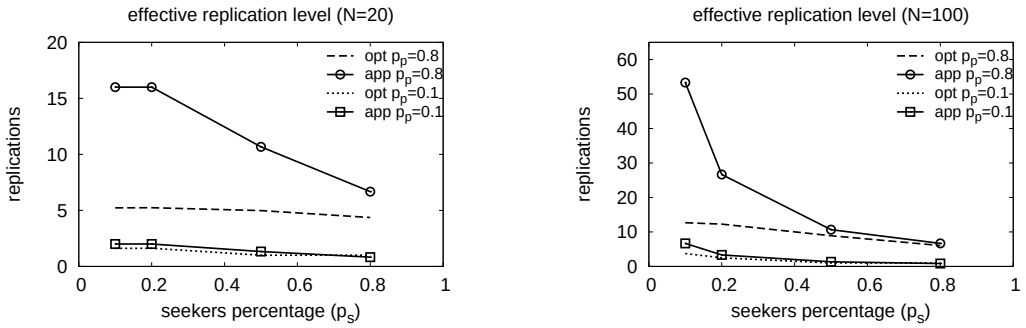
One of the generalizations of the proposed solution is the exploitation of multi-hop opportunistic paths to extend the service provisioning capabilities. This requires non-trivial extensions, as, upon encountering a peer, a node should decide if that peer is a good opportunity to either forward a request or results towards the intended final destination. The brute-force solution to this problem is clearly flooding the network both with requests and results. To achieve a more efficient solution, we can envision exploiting context-aware mechanisms, by which nodes can detect the *fitness*

of an encountered node to carry requests/results to the intended destination. From this standpoint, it would be possible to leverage on similar mechanisms, designed for traditional messaging applications [20]. Such an extension would also require to add more data to our middleware solution, in addition to the described SI field. For example, provision can be made for including personal information of the user (e.g., name, address, work, home) and device information (e.g., Nokia 3362 cell phone, 2 MB memory, camera equipped, Bluetooth capable). Similarly, privacy, trust and QoS parameters can be included if needed, and managed through appropriate extensions of the presented algorithms. Likewise, with some modifications, the service provisioning framework could also lend itself to support content sharing applications, where the two devices exchange each other's content indices, and services are opportunities to disseminate the available content.

7.2 Towards approximate estimations of the optimal configuration

The analytical model presented in this paper allows us, as shown in the previous sections, to achieve a solid understanding of the various properties of the system. It is therefore relevant to discuss how this model can be exploited in distributed algorithms that drive the system in the optimal configuration. This is one of the main subjects of future work. Different options can be already identified. One possibility is to implement distributed algorithms so that seekers can estimate the parameters needed to solve the model, i.e. N, $p^{(s)}$, $p^{(p)}$, Ec and ET, μ and λ. All of these parameters but N could be estimated by a seeker by recording information about other encountered nodes, such as the frequency of encountering other seekers, providers, the sampled average contact and inter-contact times, and so on. As far as the number of nodes in the network, an interesting direction is exploiting recent proposals for distributed counting algorithms in opportunistic networks [41].

By looking at Figure 4(a), one might suggest that an alternative possibility could be to approximate the optimal value of m with the value immediately before the saturation point (recall that this is defined as $m_c = \frac{\mu p^{(p)}}{\lambda p^{(s)}}$ in the model). Figure 4(a) actually suggests that the optimal configuration point may be close enough to the saturation threshold. There is indeed a subtlety here, as the saturation threshold represents the maximum (average) number of *effective* replications for the system not being in saturation. Therefore, m_c would be an estimate for the value of m^*



(a) $N = 20$ (b) $N = 100$

Fig. 11. Replication level in the optimal and approximate cases.

corresponding to the optimal maximum number of replications m_{opt}, rather than for m_{opt} itself. From a distributed algorithm standpoint, this option would be interesting, as it would be sufficient that each provider monitors its request's queue, and accepts requests for execution until it start becoming saturated.

However, preliminary results indicate that, for a significant range of parameters, this approximation is not accurate. Specifically, Figure 11 shows the number of replications for the cases $N = 20$ and $N = 100$ computed thorough the analytical model, in the optimal configuration ($m = m_{opt}$) and when m is set such that m^* is close to m_c (specifically, for the maximum value of m such that $m^* < m_c$). Each plot shows two couple of curves, for the minimum and maximum number of providers ($p^{(p)} = 0.1$ and 0.8, respectively). The plots clearly show that this approximation seems to be quite accurate in some cases (mainly when the number of providers is low), but does not seem precise at all in other cases (mainly when the number of providers is high). This indicates that considering only the saturation point (i.e., a first order approximation) is not enough, and more details of the analytical model must be incorporated to achieve satisfactory approximations.

8 Conclusions

In opportunistic networks, mobile nodes communicate with each other through opportunistic contacts only. Multi-hop communication is accomplished through a series of opportunistic contacts, rather than through continuous multi-hop paths. In this paper, we investigate service provisioning in an opportunistic networking environment. The main contributions of this paper include: a scheme for supporting service provisioning in opportunistic networks; an analytical model to determine the optimal

number of parallel executions required to minimize the service time without saturating the computational resources of the providers; and the performance evaluation of a system that replicates executions according to the model, in comparison with other reference policies. The developed analytical model is validated through simulation studies, and used to characterize the system performance with respect to a number of parameters - number of seekers, number of providers, request load, and providers' computational capabilities. In all investigated cases, we show that the expected service time when executions are replicated according to the model is significantly lower than the service time achieved by using naive policies - working without any background information - that either replicate requests just once, or greedily replicate requests on all encountered providers. To the best of our knowledge this is the first study on service provisioning in opportunistic networks. We are extending this work to address composability and security issues.

REFERENCES

[1] V. Cerf and al., "Delay-tolerant networking architecture," RFC 4838, 2007.

[2] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic Networking: data forwarding in disconnected mobile ad hoc networks," *IEEE Communications Magazine*, vol. 44, no. 11, Nov. 2006.

[3] M. Conti and M. Kumar, "Opportunities in opportunistic computing," *Computer*, vol. 43, no. 1, pp. 42–50, Jan. 2010.

[4] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn, "The rise of people-centric sensing," *IEEE Internet Computing*, vol. 12, no. 4, pp. 12–21, 2008.

[5] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "Soundsense: scalable sound sensing for people-centric applications on mobile phones," in *MobiSys '09: Proceedings of the 7th international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM, 2009, pp. 165–178.

[6] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell, "Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application," in *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*. New York, NY, USA: ACM, 2008, pp. 337–350.

[7] H. Lu, N. D. Lane, S. B. Eisenman, and A. T. Campbell, "Bubble-sensing: Binding sensing tasks to the physical world," *Pervasive and Mobile Computing*, vol. 6, no. 1, pp. 58 – 71, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/B7MF1-4XMD5SV-1/2/d207de79%bfa2733ab04751950e22d357

[8] M. Roccetti, M. Gerla, C. Palazzi, S. Ferretti, and G. Pau, "First responders' crystal ball: How to scry the emergency from a remote vehicle," in *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa*, April 2007, pp. 556–561.

[9] B. Manoj and A. H. Baker, "Communication challenges in emergency response," *Commun. ACM*, vol. 50, no. 3, pp. 51–53, 2007.

[10] R. B. Dilmaghani and R. R. Rao, "Future wireless communication infrastructure with application to emergency scenarios," in *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, June 2007, pp. 1–7.

[11] K. Lee, U. Lee, and M. Gerla, "To-go: Topology-assist geo-opportunistic routing in urban vehicular grids," in *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*, Feb. 2009, pp. 11–18.

[12] U. Lee, E. Magistretti, M. Gerla, P. Bellavista, and A. Corradi, "Dissemination and harvesting of urban data using vehicular sensor platforms," *IEEE Transaction on Vehicular Technology*, vol. 58, no. 2, pp. 882–901, 2009.

[13] U. Lee and M. Gerla, "A survey of urban vehicular sensing platforms," *Computer Networks*, vol. In Press, Corrected Proof, pp. –, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/B6VRG-4WWG3B8-1/2/47ac8371%4a551f9a2541564aa39a4beb

[14] M. Papazoglou, "Service Oriented Computing: Concepts, characteristics and directions," in *Proc. of IEEE WISE*, December 2003.

[15] M. Ravi, P. Stern, N. Desai, and L. Iftode, "Accessing Ubiquitous Services using Smart Phones," in *Proc. of IEEE PerCom*, March 2005.

[16] X.Gu and K. Nahrstedt, "Dynamic QoS-aware multimedia service configuration in ubiquitous computing environments," in *Proc. of IEEE ICDCS*, July 2002.

[17] D. Chakraborty, F. Perich, A. Joshi, T. Finin, and Y. Yesha, "A Reactive Service Composition Architecture for Pervasive Computing Environments," in *Proc. of IFIP PWC*, 2002.

[18] S. Helal, N. Desai, V. Verma, and C. Lee, "Konark - a service discovery and deliveryprotocol for ad-hoc network," in *Proc. of IEEE WCNC*, 2003.

[19] S. Kalasapur, M. Kumar, and B. A. Shirazi, "Dynamic Service Composition in Pervasive Computing," *IEEE TPDS*, vol. 18, no. 7, pp. 907 – 918, 2007.

[20] C. Boldrini, M. Conti, and A. Passarella, "Exploiting users' social relations to forward data in opportunistic networks: The HiBOp solution," *Elsevier Pervasive and Mobile Computing*, 2008.

[21] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Efficient Routing in Intermittently Connected Mobile Networks: The Multiple-copy Case," *IEEE Trans. on Net.*, 2008.

[22] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Performance analysis of mobility-assisted routing," in *MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2006, pp. 49–60.

[23] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: social-based forwarding in delay tolerant networks," in *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2008, pp. 241–250.

[24] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2007, pp. 32–40.

[25] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2004, pp. 187–198.

[26] Y. Wang, H. Wu, F. Lin, and N.-F. Tzeng, "Cross-layer Protocol Design and Optimization for Delay/Fault-Tolerant Mobile Sensor Networks," *IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Delay and Disruption Tolerant Wireless Communication*, vol. 26, no. 5, pp. 809–819, 2008, a preliminary version was presented in IEEE ICDCS'07.

[27] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN Routing as a Resource Allocation Problem," in *Proc. ACM SIGCOMM*, August 2007, pp. 373–384. [Online]. Available: http://prisms.cs.umass.edu/brian/pubs/balasubramanian.sigcomm.2007.pdf

[28] A. Chaintreau, P. Hui, C. Diot, R. Gass, and J. Scott, "Impact of human mobility on opportunistic forwarding algorithms," *IEEE Trans. Mob. Comp.*, vol. 6, no. 6, pp. 606–620, 2007.

[29] W. j. Hsu, T. Spyropoulos, K. Psounis, and A. Helmy, "Modeling Time-Variant User Mobility in Wireless Mobile Networks," in *Proc. of IEEE Infocom*, 2007.

[30] H. Cai and D. Y. Eun, "Crossing Over the Bounded Domain: From Exponential To Power-law Inter-meeting Time in MANET," in *Proc. of ACM MobiCom*, 2007.

[31] M. Musolesi and C. Mascolo, "Designing mobility models based on social network theory," *ACM SIGMOBILE Mobile Computing and Communication Review*, vol. 11, no. 3, July 2007.

[32] X. Zhang, J. Kurose, B. N. Levine, D. Towsley, and H. Zhang, "Study of a Bus-Based Disruption Tolerant Network: Mobility Modeling and Impact on Routing," in *Proc. ACM Intl. Conf. on Mobile Computing and Networking (Mobicom)*, September 2007, pp. 195–206. [Online]. Available: http://prisms.cs.umass.edu/brian/pubs/zhang.mobicom.2007.pdf

[33] P. Costa, C. Mascolo, M. Musolesi, and G. P. Picco, "Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks," *IEEE JSAC*, vol. 26, no. 5, 2008.

[34] C. Boldrini, M. Conti, and A. Passarella, "Design and performance evaluation of contentplace, a social-aware data dissemination system for opportunistic networks," *Computer Networks*, vol. In Press, Corrected Proof, pp. –, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/B6VRG-4X8CD4T-1/2/4e4e4b69%5c5c936de7658cbc97c1b480

[35] A. Passarella, M. Kumar, M. Conti, and E. Borgia, "Exploiting opportunistic contacts for service provisioning in bandwidth limited opportunistic networks," in *IIT-CNR Tech. Rep., available at http://bruno1.iit.cnr.it/~andrea/tr/services-tr-bw.pdf*, 2009.

[36] T. Karagiannis, J. Le Boudec, and M. Vojnović, "Power law and exponential decay of inter contact times between mobile devices," in *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*. ACM, 2007, p. 194.

[37] W. Gao, Q. Li, B. Zhao, and G. Cao, "Multicasting in delay tolerant networks: a social network perspective," in *MobiHoc '09: Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2009, pp. 299–308.

[38] H. Takagi, *Queuing Analysis Volume I: Vacation and Priority Systems, Part I*. North-Holland, 1991.

[39] G. Casella and R. Berger, *Statistical Inference*. Brooks/Cole, 1990.

[40] W. Navidi and T. Camp, "Stationary distributions for the random waypoint mobility model," *IEEE Transactions on Mobile Computing*, vol. 3, no. 1, pp. 99–108, 2004.

[41] A. Guerrieri, I. Carreras, F. D. Pellegrini, D. Miorandi, and A. Montresor, "Distributed estimation of global parameters in delay-tolerant networks," *Computer Communications*, vol. In Press, Accepted Manuscript, pp. –, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/B6TYP-4Y95V2R-1/2/c72b8c76%2aaf04c22ce6d80ae776346b

[42] M. Conti, E. Gregori, and L. Lenzini, *Metropolitan Area Networks*. Springer, 1997.

APPENDIX A

TIME TO MEET PROVIDERS

In this Appendix we prove Theorem 1, presented in Section 5.1.

Theorem 1: If contact and inter-contact times are exponentially distributed, iid, and mutually independent, the average time required by a tagged seeker to meet any provider in a set $\mathcal{A}$ starting from a random point in time, and starting after the end of a contact time are, respectively

$$ET(\mathcal{A}) = \frac{Ec + Et}{p_s(\mathcal{A})} - Ec\left(1 + \frac{Et}{Ec + Et}\right)$$

$$EL(\mathcal{A}) = \frac{Ec + Et}{p_s(\mathcal{A})} - Ec,$$

where Ec and Et are the average contact and inter-contact times, and $p_s(\mathcal{A})$ is the probability that a generic node encountered by the seeker belongs to $\mathcal{A}$.

Proof: Let us firstly consider the case in which the request for any provider in the set $\mathcal{A}$ arrives at a random point in time with respect to the seeker mobility process. It is easy to see that the request arrives i) during a contact time with a provider belonging to $\mathcal{A}$ with probability $p_s(\mathcal{A})\frac{Ec}{Et+Ec}$, ii) during a contact time with any other node with probability $(1-p_s(\mathcal{A}))\frac{Ec}{Et+Ec}$, and iii) during an inter-contact time with probability $\frac{Et}{Et+Ec}$. In the first case, $T(\mathcal{A})$ is clearly 0. In the second case, before meeting a provider in $\mathcal{A}$, the seeker will finish the current contact time. Thanks to assumption A1, the probability of meeting any node in $\mathcal{A}$ after each inter-contact time is $p_s(\mathcal{A})$. Thus, if $q_s(\mathcal{A})$ is a geometrically distributed r.v. with parameter $p_s(\mathcal{A})$ ($q_s(\mathcal{A}) \geq 1$), the time required to meet a node in $\mathcal{A}$ is $c^+ + q_s(\mathcal{A})t + (q_s(\mathcal{A}) - 1)c$ where c^+ is distributed as the unfinished contact times. Thanks to assumptions A1, A4 and A5, the r.v. $q_s(\mathcal{A})$, t and c are independent, and, as contact times are assumed exponential, the average value of c^+ is equal to Ec (this is the well-known waiting time paradox, see, e.g. [42]). Therefore, the average value of $T(\mathcal{A})$ in the second case is $\frac{Ec+Et}{p_s(\mathcal{A})}$. Following a similar line of reasoning, in the last case the seeker will finish the ongoing inter-contact time (whose residual length is denoted by t^+), and then go through $q_s(\mathcal{A})$ cycles each lasting for a contact and an inter-contact time, where $q_s(\mathcal{A}) \geq 0$, and is geometrically distributed with parameter $p_s(\mathcal{A})$. In the last case the average value of $T(\mathcal{A})$ is therefore $\frac{1-p_s(\mathcal{A})}{p_s(\mathcal{A})}Ec + \frac{1}{p_s(\mathcal{A})}Et$. The final expression of

$ET(\mathcal{A})$ follows after routine manipulations. Finally, the expression of $EL(\mathcal{A})$ can be derived according the similar derivations, by noting that $L(\mathcal{A})$ always starts at the beginning of an inter-contact time. □

APPENDIX B
DELAY OF THE FIRST STAGES

In this Appendix we provide the formal proof of Lemma 1.

Lemma 1: The average delay of the first stage on pipe i ($i = 1, \ldots, m$) can be evaluated as follows:

$$EB_i \simeq \frac{Ec + Et}{p^{(p)}} - \frac{EcEt}{Ec + Et} + N\,(Ec + Et) \ln \frac{M-1}{M-i}, \quad i < M$$

$$EB_M \simeq \frac{Ec + Et}{p^{(p)}} - \frac{EcEt}{Ec + Et} + N\,(Ec + Et)\,[\gamma + \ln(M-1)]\,.$$

where γ is the Euler constant ($\gamma \simeq 0.577$), and $M = p^{(p)}N$.

Proof: Following the line of reasoning described in Section 5.2 it is straightforward to see that $EB_i = ET(M) + iEc + \sum_{k=1}^{i-1} EL(M-k)$ holds true. The delay (of the first stage) on the first pipe is clearly the time required to meet one of the M providers from a random point in time ($T(M)$), plus a contact time c (to upload the parameters, according to assumption A2). The delay on the second pipe is the time required, after the first execution has been spawned, to find *another* provider (among the remaining $M-1$), plus another contact time c. The time to find the provider starts after the end of a contact time, and is thus $L(M-1)$. The general expression for pipe i follows immediately. The expression provided in Lemma 1 just requires some algebraic manipulation. The only point we wish to highlight, as it involves an approximation, is the derivation of a closed form expression for $\sum_{k=1}^{i-1} EL(M-k)$. By replacing the expression of EL from equation (3), and recalling that, under assumption A1, $p_s(M-k)$ is $\frac{M-k}{N}$, this can be written as $N\,(Ec + Et) \sum_{k=1}^{i-1} \left(\frac{1}{M-k}\right) - (i-1)Ec$. It is easy to show that the term $\sum_{k=1}^{i-1} \left(\frac{1}{M-k}\right)$ is equal to $H_{M-1} - H_{M-i}$, H_n being the n-th harmonic number. Therefore, it can be approximated as $\ln(M-1) - \ln(M-i)$ when $i < M$, and as $\gamma + \ln(M-1)$ when $i = M$, from which the closed form expression in equation (4) can be easily derived. □

The closed form expression in equation (4) provides a very good approximation of the EB_i numeric value, but requires nodes to estimate also any two figures among M, N, and $p^{(p)}$. Lemma 2 provides an approximation which is still precise when the ratio $\frac{i}{M}$ is close to 0, but which just requires an estimate of $p^{(p)}$.

Lemma 2: The average delay of the first stage on pipe i ($i = 1, \ldots, m$) can be approximated as follows:

$$EB_i \simeq i \cdot \frac{Ec + Et}{p^{(p)}} - \frac{EcEt}{Et + Ec}.$$

Proof: The difference with respect to the proof of Lemma 1 is the way of approximating $\sum_{k=1}^{i-1} \left(\frac{1}{M-k}\right)$. Specifically, as long as i/M is small enough, we can approximate it, by using the Taylor expansion, as $\sum_{k=1}^{i-1} \frac{1}{M} \left(1 + \frac{k}{M}\right) = \frac{i-1}{M} \left(1 + \frac{i}{2M}\right) \simeq \frac{i-1}{M}$. The expression in the lemma follows after routine manipuations. □

APPENDIX C
DELAY OF THE SECOND STAGES

In this Appendix we provide the proof of Lemma 3:

Lemma 3: The average delay of the second stages is

$$ED = \frac{1}{\mu - \frac{\lambda p^{(s)} m^*}{p^{(p)}}} + \frac{2\frac{\lambda m^*}{p^{(p)}}(Ec + Et) + 1}{2\left(\mu - \frac{\lambda p^{(s)} m^*}{p^{(p)}}\right)}.$$

Furthermore, the utilisation of the providers is $\rho = \frac{\lambda p^{(s)} m^*}{\mu p^{(p)}}$, and thus the providers are not saturated as long as the following equation holds

$$\frac{\lambda p^{(s)} m^*}{p^{(p)}} < \mu.$$

Proof: First of all, we prove that it is possible to model the second stages with an an $M^{[X]}/M/1$ batch arrival system. According to the definition provided in [38], in an $M^{[X]}/M/1$ the arrival of batches must be Poisson, the batch size must be iid, and the service time of each request must be an independent exponential r.v. We prove these three properties hereafter.

By definition we assume that the computation time of each request at each provider is an exponentially distributed r.v. with average computation time $1/\mu$. Thus computation times are exponential, iid and independent of the queue size.

Batches arrive at each tagged provider when the provider encounters any seeker. According to assumption A1 in our model the inter-contact process between a tagged provider and any seeker in a given set is statistically equivalent to the inter-contact process between a tagged seeker and any provider in a set of the same cardinality. The inter-arrival time between batches is the time interval between the end of a contact between the tagged provider and a seeker, and the end of the next contact with any other seeker. Exploiting assumption A1, we can say that the average time interval from the end of a contact with a seeker and the point in time when a new seeker is encountered is equal to $EL(p^{(s)}N)$, $p^{(s)}N$ being the average number of seekers. The average time between batches is therefore $EL(p^{(s)}N) + Ec$, and is thus equal to $\frac{Ec+Et}{p^{(s)}}$. As the inter-contact time between any two nodes is assumed to be exponential, the inter-contact time between the tagged provider and any seeker is the minimum over a set of exponential iid random variables, and it thus exponential too. This proves that the batches inter-arrival process is Poisson with rate $\lambda_X = \frac{p^{(s)}}{Ec+Et}$.

Next, we have to prove that the batch size is iid. The batch size is the number of requests a particular seeker generates for the tagged provider between two successive contacts. Requests are generated at the seeker according to a Poisson process with rate λ. By definition, each request is replicated, on average, m^* times. As all providers are encountered by the seeker with the same probability (assumption A1), requests for the tagged provider are generated by the seeker according to a Poisson process with rate $\lambda_g = \frac{\lambda m^*}{M}$ (as m^*/M is the probability that a request results in an execution on the tagged provider), and is thus memoryless. As the inter-contact time between the seeker and the tagged provider is iid, the process describing the number of requests generated during such inter-contact times regenerates at the beginning of each inter-contact time, and thus the batch size is iid. We can thus conclude that it is correct to model the second stage of the pipes as an $M^{[X]}/M/1$ system.

The expected delay of the second stage is the average delay of the $M^{[X]}/M/1$ system. If EW denotes its expected waiting time, ED is then $EW + 1/\mu$. EW can be computed according to Equation 4.13(a) (page 47 of [38]) by deriving the first and second moments of the batch size (EX and EX^2), and the first and second moments of the computation time ($1/\mu$ and $2/\mu^2$, respectively). To compute EX and EX^2 we note that the r.v. $X|\tau$ (the batch size conditioned to an inter-contact time equal to τ)

is Poisson with rate $\lambda_g \tau$. Thus, we obtain:

$$\begin{aligned} EX|\tau &= \lambda_g \tau \\ EX^2|\tau &= (\lambda_g \tau)^2 + \lambda_g \tau \ . \end{aligned}$$

By recalling that the inter-contact time between the tagged provider and a particular seeker is exponential with average value $E\tau = EL(1) + Ec = N(Ec + Et)$

$$\begin{aligned} EX &= \int_0^\infty \lambda_g \tau f(\tau) d\tau = \lambda_g E\tau = \lambda_g N(Ec + Et) = \frac{\lambda m^*}{p^{(p)}}(Ec + Et) \\ EX^2 &= \int_0^\infty \lambda_g^2 \tau^2 f(\tau) d\tau + \int_0^\infty \lambda_g \tau f(\tau) d\tau = \\ &= \lambda_g^2 E\tau^2 + \lambda_g E\tau = 2\left(\lambda_g E\tau\right)^2 + \lambda_g E\tau = \\ &= \lambda_g N(Ec + Et)\left[2\lambda_g N(Ec + Et) + 1\right] = \frac{\lambda m^*}{p^{(p)}}(Ec + Et)\left[2\frac{\lambda m^*}{p^{(p)}}(Ec + Et) + 1\right] \ . \end{aligned}$$

According to [38], the utilization of the providers (ρ) is as follows:

$$\rho = \lambda_X EX \frac{1}{\mu} = \frac{\lambda m^* p^{(s)}}{p^{(p)} \mu},$$

and the stability condition is

$$\frac{\lambda m^* p^{(s)}}{p^{(p)}} < \mu \ .$$

Based on these expressions, and on the expression for EW provided in [38], the expected delay of the second stage becomes:

$$ED = EW + \frac{1}{\mu} = \frac{\lambda_X EX \frac{2}{\mu^2}}{2(1-\rho)} + \frac{EX^2 \frac{1}{\mu}}{2EX(1-\rho)} + \frac{1}{\mu} = \frac{1}{\mu - \frac{\lambda p^{(s)} m^*}{p^{(p)}}} + \frac{2\frac{\lambda m^*}{p^{(p)}}(Ec + Et) + 1}{2\left(\mu - \frac{\lambda p^{(s)} m^*}{p^{(p)}}\right)} \ .$$

□

APPENDIX D

OPTIMAL REPLICATION

In this Appendix we provide the proofs of Lemmas 6 and Theorems 2 and 3, which have been presented in Section 5.5.

D.1 Proof of Lemma 6

Lemma 6: The probability that the tagged seeker receives the results after spawning exactly i executions ($i = 1, \ldots, m$) can be computed as follows:

$$p_1(m^*) = \frac{\delta_1(m^*)}{\delta_1(m^*) + \psi_2}$$

$$p_i(m^*) = \frac{\delta_i(m^*)}{\delta_i(m^*) + \psi_{i+1}} - \frac{\delta_{i-1}(m^*)}{\delta_{i-1}(m^*) + \psi_i}$$

$$p_m(m^*) = \frac{\psi_m}{\delta_{m-1}(m^*) + \psi_m}$$

where $\delta_i(m^*) \triangleq \frac{1}{EH_i(m^*)}$, and $\psi_i \triangleq \frac{1}{EB_i}$.

Proof: Let us first consider p_1, i.e., the probability that the seeker receives the results by spawning a single execution only. This event occurs if the execution time on the first provider (i.e., the time required to get results from the first provider) is less than the time to upload the input parameters on the second provider, or, in other words, if the seeker receives the results from the first provider before uploading the input parameters to the second provider. Therefore, $p_1 = P[R_1 < B_2]$, as R_1 is the execution time on the first provider, and B_2 is the upload time on the second provider. By assuming that R_i and B_j are independent exponential r.v. with rates equal to $\delta_1(m^*) \triangleq \frac{1}{ER_1(m^*)} = \frac{1}{EH_1(m^*)}$, and $\psi_2 \triangleq \frac{1}{EB_2}$, respectively, we obtain (see, e.g., [39])

$$p_1(m^*) = \frac{\delta_1(m^*)}{\delta_1(m^*) + \psi_2} \ .$$

To derive the expression for $p_i, i > 1$ we make the following observations. The seeker receives the results after spawning exactly i executions if, for each $j < i$, the expected service time of the first j providers is higher than the time to upload the input parameters on the $(j+1)$-th provider, i.e., if $\forall j \, s.t. \, 1 \leq j < i \, , H_j > B_{j+1}$. In other words, if the seeker does *not* receives the results from any of the first j providers before uploading the input parameters to the $(j+1)$-th provider. In addition, the seeker must also receive the results from any of the first i providers *before* uploading the input parameters to the $(i+1)$-th provider, i.e., $H_i < B_{i+1}$ must hold true. Therefore, we can write the probability of spawning exactly i executions as follows:

$$p_i = P\left[H_1 > B_2, \ldots, H_{i-1} > B_i, H_i < B_{i+1}\right] \ . \tag{14}$$

Note that, by definition, the sequence of r.v. H_i, $i \geq 1$ is non-increasing. For each $i > 1$, H_i is the minimum over the first i execution times. Therefore, we can write

$$H_i = \min_{j=1,\ldots,i} \{R_j, j = 1, \ldots, i\} = \min\{R_i, H_{i-1}\} \leq H_{i-1} \ .$$

On the other hand, the sequence of r.v. B_i is non-decreasing as, by definition, B_i is the time to upload the parameters to the i-th provider, and, according to our assumptions, is thus the time to meet the i-th provider after uploading the input parameters to the $(i-1)$-th provider. Thus, $B_i \geq B_{i-1}$ holds true for each $i > 1$. Based on these observations, it follows that $H_{i-1} > B_i \Rightarrow H_{j-1} > B_j$ holds true for all $j < i$. Therefore, we can rewrite equation (14) as follows:

$$p_i = P\left[H_1 > B_2, \ldots, H_{i-1} > B_i, H_i < B_{i+1}\right] = P[H_{i-1} > B_i, H_i < B_{i+1}] \ .$$

Applying the same observation, and the law of total probability, we can also write

$$\begin{aligned} P[H_{i-1} > B_i] &= P[H_{i-1} > B_i, H_i < B_{i+1}] + P[H_{i-1} > B_i, H_i > B_{i+1}] \\ &= P[H_{i-1} > B_i, H_i < B_{i+1}] + P[H_i > B_{i+1}] \ . \end{aligned}$$

and thus

$$P[H_{i-1} > B_i, H_i < B_{i+1}] = P[H_{i-1} > B_i] - P[H_i > B_{i+1}] = P[H_i < B_{i+1}] - P[H_{i-1} < B_i] \ .$$

By assuming, again, that, for any i, H_i and B_{i+1} are exponential independent r.v. with rates $\delta_i(m^*) \triangleq \frac{1}{EH_i(m^*)}$, and $\psi_{i+1} \triangleq \frac{1}{EB_{i+1}}$, we obtain

$$p_i(m^*) = \frac{\delta_i(m^*)}{\delta_i(m^*) + \psi_{i+1}} - \frac{\delta_{i-1}(m^*)}{\delta_{i-1}(m^*) + \psi_i} \ .$$

The expression for p_m can be derived following a similar line of reasoning:

$$p_m(m^*) = P[H_1 > B_2, \ldots, H_{m-1} > B_m] = P[H_{m-1} > B_m] = \frac{\psi_m}{\delta_{m-1}(m^*) + \psi_m} \ .$$

Finally, we show that the expressions of $p_i(m^*)$, $i = 1, \ldots, m$ are a well-defined probability distribution. To this end we show that for each $i = 1, \ldots, m$, $0 \leq p_i(m^*) \leq 1$ holds true, and that $\sum_{i=1}^{m} p_i(m^*) = 1$ also holds true.

It is easy to note from their definitions that $\psi_i > 0 \ \forall i$, and $\delta_i(m^*) \geq 0 \ \forall i$. Specifically, as $\psi_i = \frac{1}{EB_i}$, $\psi_i \geq 0$ holds true as long as the average contact and inter-contact times

are finite. Furthermore, $\delta_i(m^*)$ is equal to $\frac{1}{EH_i(m^*)}$, is thus greater or equal to 0, and is actually equal to 0 only when $EH_i(m^*)$ is infinite. This can occur only when the system saturates and $ED(m^*)$ diverges. Based on this remark, we can immediately conclude that $0 \leq p_1(m^*) < 1$ and $0 < p_m(m^*) \leq 1$ hold true, and that $p_1(m^*)$ is 0 ($p_m(m^*)$ is 1) only when the system saturates.

Let us now analyze the case of $p_i(m^*)$ for $1 < i < m$. By the definitions of EH_i and EB_i, it is easy to show that, when the system is not saturated, $EH_{i+1} < EH_i \; \forall i$ (when the system is saturated $EH_i = \infty \; \forall i$), and that $EB_{i+1} > EB_i \; \forall i$. Thus, the following relations also hold true:

$$\begin{cases} \delta_{i+1} > \delta_i \\ \psi_{i+1} < \psi_i \end{cases} \tag{15}$$

Based on equation (15), we derive that

$$\delta_i(m^*)\psi_i > \delta_{i-1}(m^*)\psi_{i+1}$$

and we can thus show that $p_i(m^*) > 0$ holds true, when the system is not saturated, for all $i = 2, \ldots, m-1$:

$$p_i(m^*) = \frac{\delta_i(m^*)}{\delta_i(m^*) + \psi_{i+1}} - \frac{\delta_{i-1}(m^*)}{\delta_{i-1}(m^*) + \psi_i} = \frac{\delta_i(m^*)\psi_i - \delta_{i-1}(m^*)\psi_{i+1}}{(\delta_i(m^*) + \psi_{i+1})(\delta_{i-1} + \psi_i)} > 0$$

Furthermore, note that $p_i(m^*)$ is equal to 0 for all $i = 2, \ldots, m-1$ when the system is saturated (as $\delta_i(m^*$ is equal to 0 for all i). Finally, also note that, as $\frac{\delta_i(m^*)}{\delta_i(m^*)+\psi_{i+1}}$ is always less than 1, also $p_i(m^*) < 1$ holds true.

Finally, it can also be shown that $\sum_{i=1}^{m} p_i(m^*) = 1$, as follows (we omit the dependence on m^* for simplicity):

$$\sum_{i=1}^{m} p_i = \frac{\delta_1}{\delta_1 + \psi_2} + \left(\frac{\delta_2}{\delta_2 + \psi_3} - \frac{\delta_1}{\delta_1 + \psi_2}\right) + \ldots + \left(\frac{\delta_{m-1}}{\delta_{m-1} + \psi_m} - \frac{\delta_{m-2}}{\delta_{m-2} + \psi_{m-1}}\right) + \frac{\psi_m}{\delta_{m-1} + \psi_m} =$$
$$= \frac{\delta_{m-1}}{\delta_{m-1} + \psi_m} + \frac{\psi_m}{\delta_{m-1} + \psi_m} = 1$$

□

D.2 Proofs of Theorem 2

Theorem 2: The average value of the number of executions actually spawned when m replicas are allowed (m^*) is the solution of the following fixed point equation:

$$m^* = \sum_{i=1}^{m} i \cdot p_i(m^*)$$

where p_i are as in equations (11). Specifically, equation (12) admits either one or three solutions. In the former case, the unique solution is stable. If it falls in the saturated region, then $m^* = m$ (and is greater than the saturation threshold $m_c = \frac{\mu p^{(p)}}{\lambda p^{(s)}}$). Or, it can fall in the non-saturated region (i.e., $m^* < m_c$). When equation (12) admits three solutions, one on them falls in the saturated region, and is stable, while the other two fall in the non-saturated region, and one of them is stable.

Proof: The expression of m^* follows by the definition of m^*, which is the average number of spawned replicas. By definition, it is thus equal to $\sum_{i=1}^{m} i \cdot p_i$. However, as p_i are also functions of m^* as shown by Lemma 6, m^* becomes the solution of the fixed point equation $m^* = \sum_{i=1}^{m} i \cdot p_i(m^*)$.

We now prove the properties of the solutions of equation (12) highlighted in the theorem. First of all, we show that at least one solution always exists. We define $f(x)$ as the right-hand side of equation (12) by considering a generic variable x:

$$f(x) = \sum_{i=1}^{m} i \cdot p_i(x) \ .$$

Solving the fixed point equation (12) means finding the points where $f(x)$ intersects the line $y(x) = x$. The expression of $f(x)$ can be expanded as follows:

$$\begin{aligned} f(x) &= \frac{\delta_1(x)}{\delta_1(x)+\psi_2} + 2\left(\frac{\delta_2(x)}{\delta_2(x)+\psi_3} - \frac{\delta_1(x)}{\delta_1(x)+\psi_2}\right) + \ldots + \\ &+ (m-1)\left(\frac{\delta_{m-1}(x)}{\delta_{m-1}(x)+\psi_m} - \frac{\delta_{m-2}(x)}{\delta_{m-2}(x)+\psi_{m-1}}\right) + m\frac{\psi_m}{\delta_{m-1}(x)+\psi_m} \\ &= m - \frac{\delta_{m-1}(x)}{\delta_{m-1}(x)+\psi_m} - \ldots - \frac{\delta_1(x)}{\delta_1(x)+\psi_2} = \\ &= m - \sum_{i=1}^{m-1} \frac{\delta_i(x)}{\delta_i(x)+\psi_{i+1}} \end{aligned} \tag{16}$$

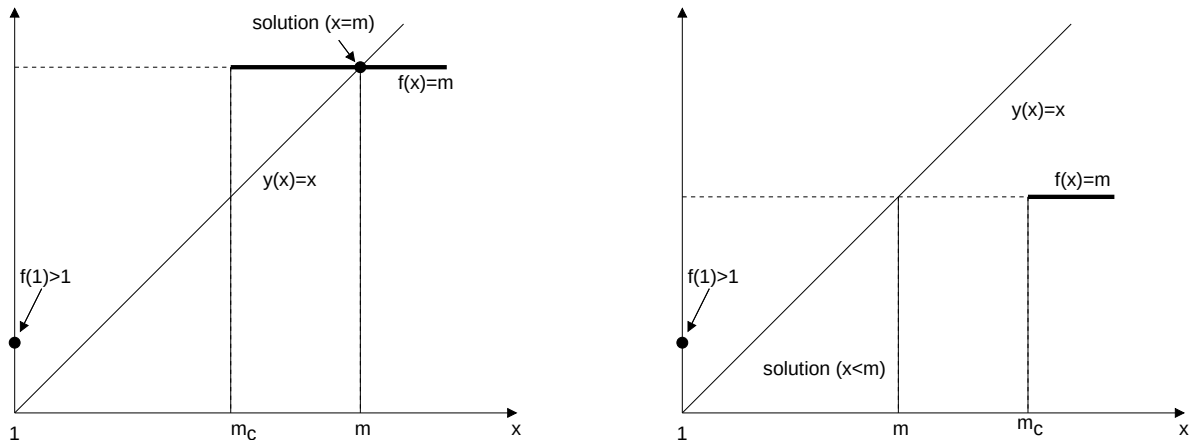


Fig. 12. Existence of solutions for equation (12).

Equation (16) allows us to show several interesting properties, and will be used extensively throughout the proof. For now, note that, as $\frac{\delta_i}{\delta_i+\psi_{i+1}} < 1$, we can write

$$f(x) = m - \sum_{i=1}^{m-1} \frac{\delta_i(x)}{\delta_i(x) + \psi_{i+1}} > m - (m-1) = 1 \; \forall x \; .$$

Furthermore, we can also say that $f(x) < m$ in the non-saturated region (as $\delta_i(x) > 0 \; \forall i$), and $f(x) = m$ in the saturated region, i.e., for all x greater than $m_c \triangleq \frac{\mu p^{(p)}}{\lambda p^{(s)}}$. As $f(x)$ is a continuous function, these two properties are sufficient to conclude that at least one solution always exists for the fixed point equation (12). To understand this, let us focus on Figure 12 (note that, being the average value of a r.v. distributed on $[1, m]$, $f(x)$ has a physical meaning just in the interval $[1, m]$). When $m_c < m$, then the point $x = m$ is necessarily a solution, as $f(x) = m \; \forall x > m_c$. When $m_c > m$ then there is necessarily a solution within the interval $[1, m)$, as $f(1) > 1$ and $f(x) < m \; \forall x < m_c$.

Next, we analyze how many solutions can exist for the fixed point equation (12). The key part of this analysis is showing that $f(x)$ is always monotonically increasing and convex in $[1, m)$. This requires some elaboration, that we provide at the end of the proof. Before this, by exploiting this finding, it is easy to analyse the solutions of the fixed point equation (12). Again, we can separately consider the cases $m_c < m$ and $m_c > m$, as shown in Figure 13.

When $m_c < m$, we should distinguish two cases, denoted by case (i) and case (ii) in Figure 13(a). In case (i) $f(x)$ is greater than x in $[1, m_c]$, and thus the fixed

point equation (12) admits only one solution $m^* = m$. Note that this solution is stable, as $f(m-\epsilon) > m-\epsilon$ and $f(m+\epsilon) < m+\epsilon \; \forall \epsilon > 0$ and ϵ is sufficiently small. From a systems standpoint, case (i) corresponds to a situation in which the system, as seekers are allowed to spawn up to m replicas, can only work in saturation. In case (ii), besides the stable (saturated) solution $m^* = m$, only two other solutions exist, both falling in the interval $(1, m_c)$. The smallest solution (x_1) is stable, while the largest one (x_2) is not. The fact that the fixed point equation admits one stable solution in the non-saturated region is very important. In the average case, seekers receive results for requests after spawning x_1 executions only, and experience a finite service time. Therefore, they will never replicate executions so aggressively to make the system operate in the saturated condition corresponding to the solution $m^* = m$.

The case when $m_c > m$ is simpler to analyse (see Figure 13(b)). As $f(x)$ is convex in $[1, m_c)$, there can only be one unique solution (x_1) for the fixed point equation, this solution falls in the non-saturated region ($x_1 < m_c$), and is stable. From a systems standpoint this means that the system will always operate in this stable (non-saturated) configuration.

The properties of the solutions highlighted so far can be also seen from a complementary standpoint. When lots of replications are allowed in comparison with the saturation point m_c (i.e., case (i) of Figure 13(a)), the system inevitably works in saturation. Solutions x_1 and x_2 in Figure 13(a)) appear as m is reduced. Further reducing m results in the case in Figure 13(b), in which the replication level is so low that saturated solutions are not possible at all.

The last part of the proof is devoted to show that $f(x)$ is always monotonically increasing and convex in $[1, m_c)$. To this end, we prove that $\frac{\partial f(x)}{\partial x} > 0$ and $\frac{\partial^2 f(x)}{\partial x^2} > 0$ in $[1, m_c)$. Let us analyze $\frac{\partial f(x)}{\partial x}$ first. From equation (16) we derive:

$$\frac{\partial f(x)}{\partial x} = -\sum_{i=1}^{m} \frac{\frac{\partial \delta_i(x)}{\partial x}\psi_{i+1}}{(\delta_i(x)+\psi_{i+1})^2} \; . \tag{17}$$

By recalling that $\delta_i(x)$ is defined as $\sum_{j=1}^{i}\beta_j(x) = \sum_{j=1}^{i}\frac{1}{ER_j(x)} = \sum_{j=1}^{i}\frac{1}{EB_j+ED(x)+E\theta}$ (see Lemma 5 and 6), we obtain:

$$\frac{\partial \delta_i(x)}{\partial x} = \sum_{j=1}^{i}\frac{\partial \beta_j(x)}{\partial x} = -\sum_{j=1}^{i}\frac{\frac{\partial ER_j(x)}{\partial x}}{(ER_j(x))^2} = -\sum_{j=1}^{i}\frac{\frac{\partial ED(x)}{\partial x}}{(ER_j(x))^2} \tag{18}$$

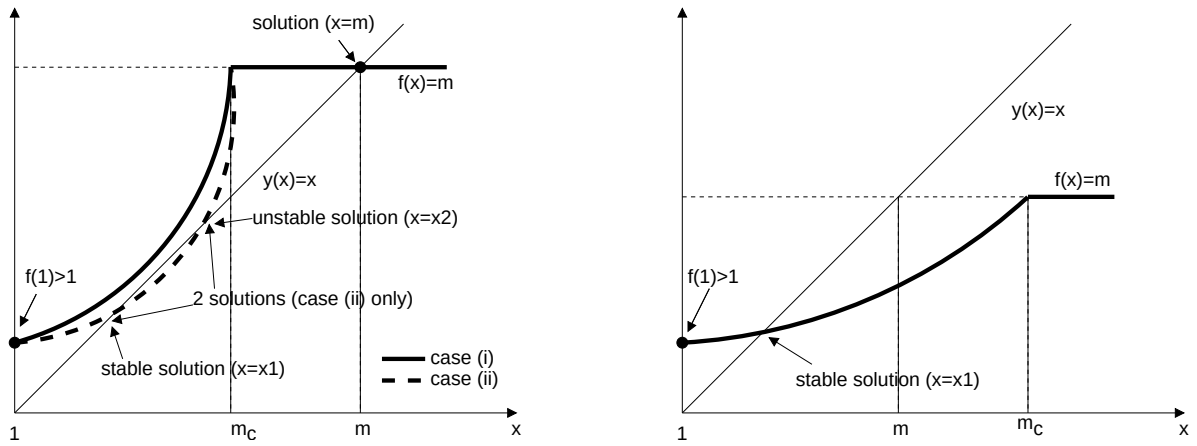


Fig. 13. Properties of the solutions of equation (12).

By looking at equation (6), and by defining $a \triangleq \frac{\lambda p^{(s)}}{p^{(p)}}$ and $b \triangleq \frac{2\lambda(Ec+Et)}{p^{(p)}}$, it is possible to compute $\frac{\partial ED(x)}{\partial x}$ as follows:

$$\frac{\partial ED(x)}{\partial x} = \frac{\partial \left[\frac{bx+3}{2(\mu-ax)}\right]}{\partial x} = \frac{2b\mu + 6a}{[2(\mu-ax)]^2} > 0 \; . \tag{19}$$

Equation (17) shows that $\frac{\partial ED(x)}{\partial x}$ is finite and strictly positive in the non-saturated region $[1, m_c)$ (note that in that region $\mu - ax > 0$, and the saturation threshold m_c is equal to $\frac{\mu}{a}$, see equation (7)). By looking at equation (18), it follows that $\frac{\partial \delta_i(x)}{\partial x} < 0 \; \forall i$, and, by looking at equation (17), that $\frac{\partial f(x)}{\partial x} > 0$ (recall that $\psi_i > 0 \; \forall i$ by definition). This proves that $f(x)$ is finite and monotonically increasing in $[1, m_c)$, and that it diverges to $+\infty$ in the saturation threshold $m_c = \frac{\mu}{a}$.

To prove that $f(x)$ is also convex in $[1, m_c)$ we proceed as follows. From equation (17) we can compute $\frac{\partial^2 f(x)}{\partial x^2}$ as follows:

$$\frac{\partial^2 f(x)}{\partial x^2} = -\sum_{i=1}^{m} \psi_{i+1} \frac{\frac{\partial^2 \delta_i(x)}{\partial x^2}[\delta_i(x) + \psi_{i+1}] - 2\left[\frac{\partial \delta_i(x)}{\partial x}\right]^2}{[\delta_i(x) + \psi_{i+1}]^3} \; . \tag{20}$$

In the following we prove that $\frac{\partial^2 \delta_i(x)}{\partial x^2} < 0 \; \forall i$ in the non-saturated region. As $\delta_i(x)$ and ψ_i are strictly positive for all i, this is sufficient to prove that $f(x)$ is convex in $[1, m_c)$. Actually, we prove hereafter that $\frac{\partial^2 \beta_j(x)}{\partial x^2} < 0 \; \forall j$. As $\delta_i(x) = \sum_{j=1}^{i} \beta_j(x)$, this is sufficient to our purpose.

By looking at equation (18), we can write $\frac{\partial^2 \beta_j(x)}{\partial x^2}$ as follows:

$$\frac{\partial^2 \beta_j(x)}{\partial x^2} = \frac{2\left[\frac{\partial ED(x)}{\partial x}\right]^2}{[ER_j(x)]^3} - \frac{\frac{\partial^2 ED(x)}{\partial x^2}}{[ER_j(x)]^2} \tag{21}$$

Therefore, $\frac{\partial^2 \beta_j(x)}{\partial x^2}$ is strictly negative iff

$$\left[\frac{\partial ED(x)}{\partial x}\right]^2 < \frac{ER_j(x)}{2} \cdot \frac{\partial^2 ED(x)}{\partial x^2} \tag{22}$$

From equation (19) we can derive $\frac{\partial^2 ED(x)}{\partial x^2}$:

$$\frac{\partial^2 ED(x)}{\partial x^2} = \frac{4a(2b\mu + 6a)}{[2(\mu - ax)]^2} \ . \tag{23}$$

Furthermore, we can write $ER_j(x)$ in the following form:

$$ER_j(x) = EB_j + ED(x) + E\theta = ED(x) + Q_j = \frac{A_j + B_j x}{2(\mu - ax)} \tag{24}$$

where $Q_j \triangleq EB_j + E\theta$, $A_j \triangleq 3 + 2\mu Q_j$, and $B_j \triangleq b - 2aQ_j$. By substituting equations (19), (23) and (24) in equation (22) we obtain the following condition:

$$\frac{\partial^2 \beta_j(x)}{\partial x^2} < 0 \Leftrightarrow B_j \cdot x > \frac{\mu}{a} \cdot B_j \ . \tag{25}$$

The last step to solve equation (25) is studying the sign of B_j in the non-saturated region. It is easy to show that the condition $B_j < 0$ can be substituted as follows:

$$B_j < 0 \Leftrightarrow \frac{Ec + Et}{p^{(s)}} < EB_j + E\theta \ . \tag{26}$$

We consider the approximate expression of EB_j provided in equation (5). It can be shown that this expression is actually *lower* than the real value of EB_j. Replacing the approximate expression of equation (5) in (26) thus results in a more tight condition to check. It will be clear in the following that, with respect to our analysis, it is sufficient to check this tighter (but easily to compute) condition. Specifically, from equations (5) and (8), the condition in equation (26) becomes:

$$B_j < 0 \Leftarrow j > \frac{p^{(p)}}{p^{(s)}} - p^{(p)}\left[N - \frac{Ec(Ec + 3Et)}{(Ec + Et)^2}\right] \ . \tag{27}$$

It is easy to show that the relation $0 \leq \frac{Ec(Ec+3Et)}{(Ec+Et)^2} \leq 1.125$ holds true. We can thus approximate the condition in equation (27) as follows:

$$B_j < 0 \Leftarrow j > \frac{p^{(p)}}{p^{(s)}} - p^{(p)}N = \frac{p^{(p)}}{p^{(s)}}(1 - Np^{(s)}) \ . \tag{28}$$

As j is always greater than or equal to 1, the condition in (28) is always verified if $Np^{(s)} > 1$. This is clearly always true as $Np^{(s)}$ is the average number of seekers, which has to be greater than 1 for the system to have physical meaning. This shows therefore that, for any physically meaningful instance of the system, a tighter condition, with respect to that in equation (26), is verified, and thus also that condition is satisfied. We can thus conclude that B_j is always negative, and therefore that the following relation holds true:

$$\frac{\partial^2 \beta_j(x)}{\partial x^2} < 0 \Leftrightarrow x < \frac{\mu}{a} \ . \tag{29}$$

As $\frac{\mu}{a}$ is equal to the saturation threshold m_c, equation (29) tells that, in the non-saturated region, $\frac{\partial^2 \beta_j}{\partial x^2}$ is always negative for all j. From this it follows that, in the non-saturated region ($x \in [1, m_c)$), $f(x)$ is convex. This concludes the proof.

□

APPENDIX E

PERFORMANCE FOR DIFFERENT NETWORK SIZES

In this appendix we show additional results with respect to those presented in Section 6.4, regarding the dependence of the optimal service time on the network size.

Specifically, Figure 14 shows the expected service time of the optimal, greedy and single policies for $p^{(p)} = 0.5$ and $N = 50$. This confirms the agreement of the analytical and simulation models, as it was the case of $N = 20$. Furthermore, Figure 15 show the optimal expected service time according to the analytical and simulation models for the different values of $p^{(s)}$ and $p^{(p)}$ for $N = 50$, confirming again the agreement between the two models. Similar remarks can be drawn also when considering the cases of $N = 80$ (Figures 16 and 17) and $N = 100$ (Figures 18 and 19).

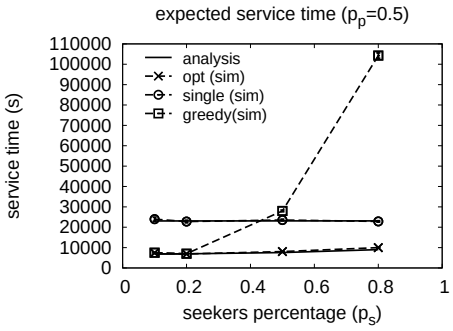
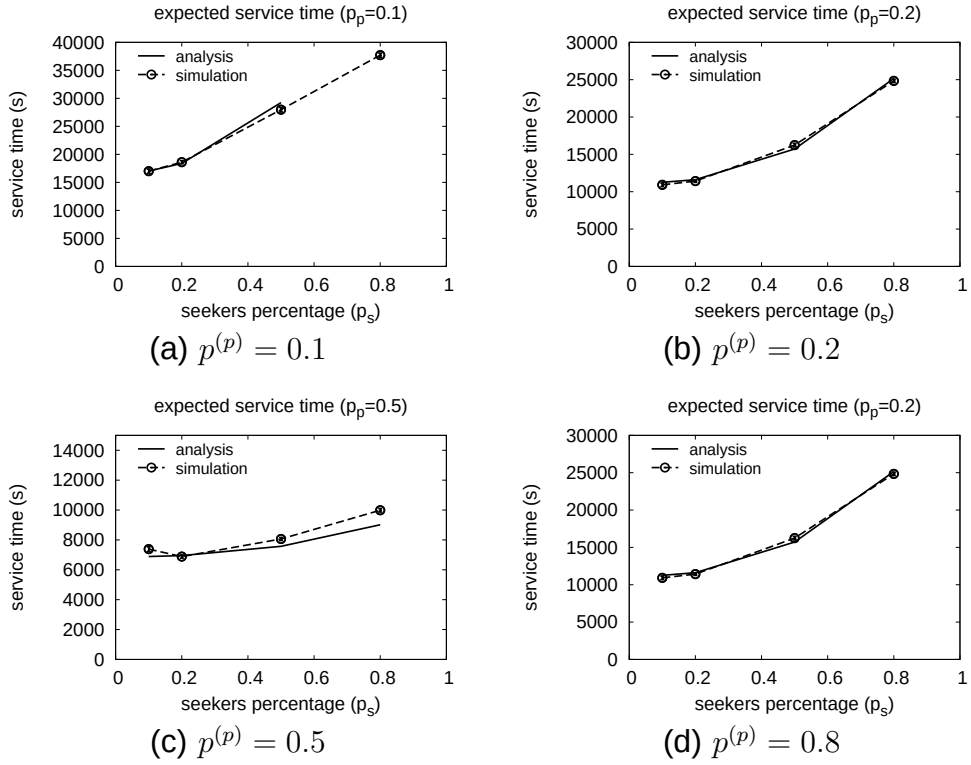


Fig. 14. Expected service time in the optimal, single and greedy policies ($N = 50, p^{(p)} = 0.5$).



(a) $p^{(p)} = 0.1$ (b) $p^{(p)} = 0.2$

(c) $p^{(p)} = 0.5$ (d) $p^{(p)} = 0.8$

Fig. 15. Expected service time for varying values of $p^{(s)}$ and $p^{(p)}$ ($N = 50$).

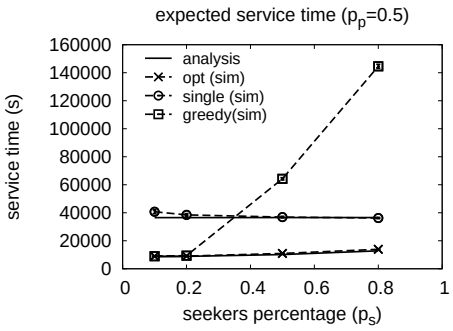


Fig. 16. Expected service time in the optimal, single and greedy policies ($N = 80, p^{(p)} = 0.5$).

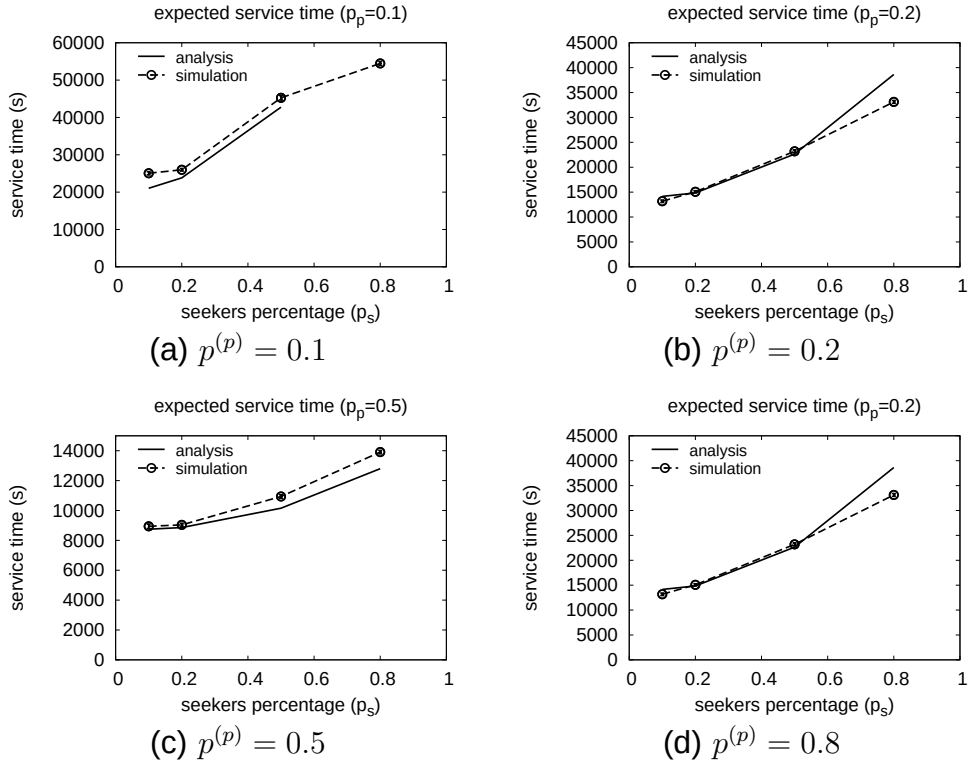


Fig. 17. Expected service time for varying values of $p^{(s)}$ and $p^{(p)}$ ($N = 80$).

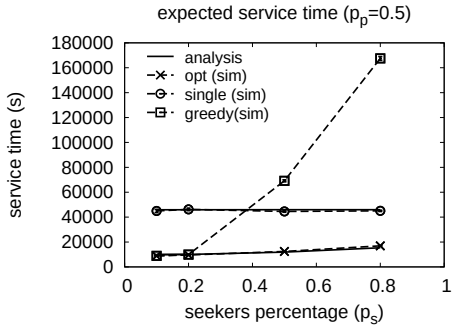


Fig. 18. Expected service time in the optimal, single and greedy policies ($N = 100, p^{(p)} = 0.5$).

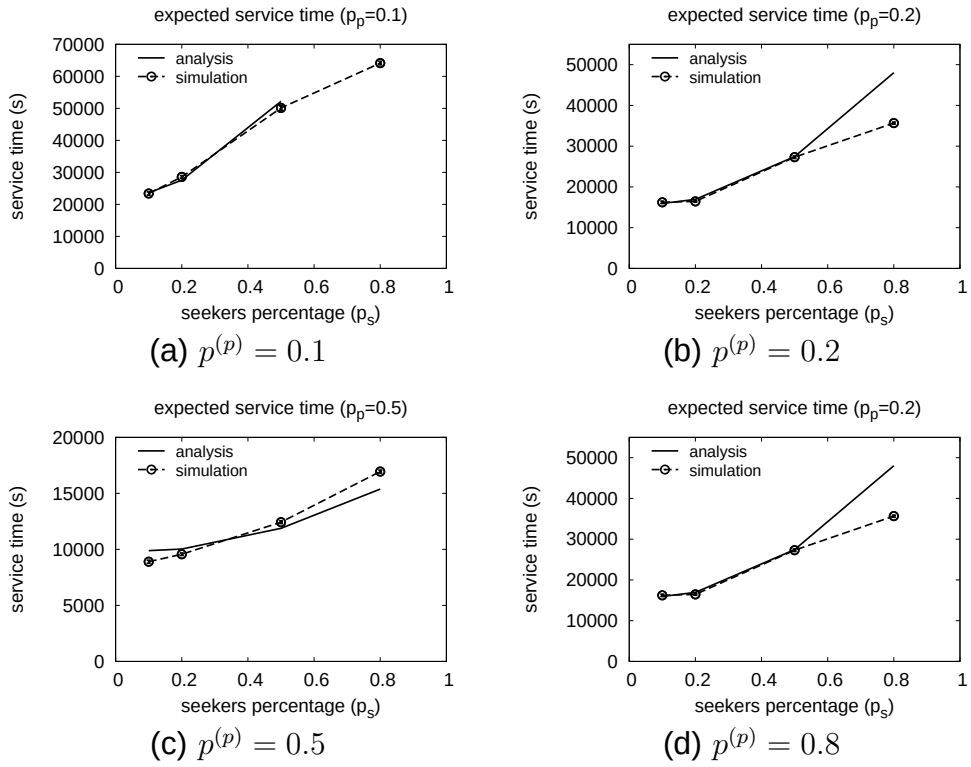


Fig. 19. Expected service time for varying values of $p^{(s)}$ and $p^{(p)}$ ($N = 100$).