Minimum-Delay Service Provisioning in Opportunistic Networks

Andrea Passarella, Mohan Kumar, Marco Conti and Eleonora Borgia

Abstract—Opportunistic networks are created dynamically by exploiting contacts between pairs of mobile devices that come within communication range. While forwarding in opportunistic networking has been explored, investigations into asynchronous service provisioning on top of opportunistic networks 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) their (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. 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

In a network with mobile devices a contact occurs when pairs of mobile devices are within range. An opportunistic network is created when several such opportunistic contacts occur between pairs of devices, distributed in time and space [1]. 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.

Opportunistic networks enable the emerging concept of *opportunistic computing* [2]. The key observation of opportunistic computing is that the environment around (mobile) users, features a steadily increasing set of heterogeneous resources available on fixed and mobile devices with wireless networking capabilities. 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. Essentially, we envision opportunistic computing as an evolution of distributed computing in which resources are accessed through opportunistic contacts, thus complementing solutions based on well-connected networks only. 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. We investigate this novel research area, as researchers in the past have explored i) opportunistic networking mainly as a means for forwarding message packets, and ii) service provisioning in well-connected networks. Service provisioning in the framework of opportunistic computing enables interesting applications such as pervasive healthcare, intelligent transportation systems, crisis management, participatory sensing, as discussed in [2].

We consider a very simple architecture in which nodes can opportunistically request service executions to directly encountered peers, and collect results *the next time* they encounter those peers after they have completed the execution. 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. Main contributions of this paper include: i) development of a model to depict the behavior of the service provisioning system; ii) analysis to determine the optimal number of parallel executions to be spawned; and iii) exhaustive simulation studies to validate the model.

- *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*

2 Related work

Service-oriented architectures have been investigated in the area of ubiquitous and pervasive computing [3], e.g. in [4] 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 that 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 [5], 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. For dynamic service composition (e.g., [6]) the description of the service and the mechanisms employed to identify matches between a requested service and available ones are critical to the successful operation.

Our current work differs from this body of research as the focus here is on opportunistic networking environments. Being the networking environment so 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. [7], [8]), mobility analysis (e.g. [9]) and, more recently, on data-centric architectures for content delivery (e.g., [10]).

The work presented in this paper can be seen as one of the building blocks of the recently proposed concepts of opportunistic computing [2] and people-centric sensing [11]. The latter work already produced significant results, but does not exploit opportunistic contacts for service provisioning.

3 Service provisioning

The reference service provisioning model 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* (node needing a service) and *provider* (node providing the service) 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, asymptotically tends to 1 in the limit $t \to \infty$. We assume that nodes 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 ongoing (e.g., $S_p = [S4]$) and services completed (e.g., $S_c = [S3]$). During a contact time, 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. This algorithm just requires reliable single-hop communications between nodes, that we assume available.

```
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($S_c$) $\to n_a$
    end if
    if $n_b(S_p)$ in $n_a(S_c)$ then
        $n_a$ sends output($S_c$) $\to n_b$
    end if
    if $n_a(S_n)$ in $n_b(S_a)$ and worth replicating on $n_b$ then
        $n_a$ sends input($S_n$) $\to n_b$
    end if
    if $n_b(S_p)$ in $n_a(S_a)$ and worth replicating on $n_a$ then
        $n_b$ sends input($S_n$) $\to n_a$
    end if
end while
```

Fig. 1. Algorithm for service provisioning.

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. 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. The model described hereafter determines the optimal number of replications.

4 Optimal service provisioning

The analytical model presented in this paper focuses on a *tagged* seeker wishing to avail of a given service, assumed to be statistically representative for a generic seeker. We provide a model of the expected service time, 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 service request generated at a seeker.

(Service) Execution: an execution of a request, spawned by a seeker on a provider.

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

Execution Time: the time elapsed at the seeker from a request generation to the reception of the results from a particular provider.

Service Time: minimum over the execution times.

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. This motivates looking for the *optimal number of replicas*.

4.1 Replication

The (tagged) seeker manages each request as described below. The seeker uploads the input parameters to each encountered provider opportunistically, until either of the following event occurs: i) it completes m uploads, where m is the maximum number of parallel executions a seeker can generate; or ii) it downloads the results from a provider before generating all the m replicas. The model provides the optimal value (m_{opt}) of the m parameter, i.e. the optimal number of maximum allowed replications. 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 and has been completed. 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. 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 several output results, related to different requests served by the same provider. 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 2. We assume there are N nodes in the network, and that the tagged seeker issues requests according to a Poisson process with rate λ. 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 is a seeker. The case in which the requests rate is different for each seeker is investigated in [12]. The average number of seekers is thus $k \triangleq p^{(s)}N$. 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 2 represents a different execution (on a different provider) for a request issued by the tagged seeker[1]. Denoting by p_i the probability that only i replicas are spawned by the seeker before receiving the results back, the average number of spawned replicas (also referred to as effective replication level) is $m^* = \sum_{i=1}^{m} ip_i$.

Let us now consider the individual execution times. For each request, pipe i corresponds to the i-th provider

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

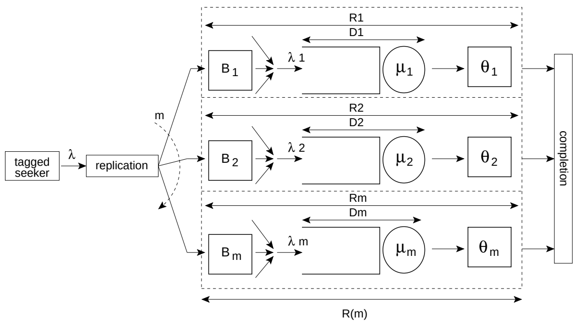


Fig. 2. General scheme of the replication process.

starting the computations, i.e., pipes are ordered in increasing temporal order of computation starting times. Each pipe consists of three stages. The first stage represents the time B_i required to complete the i-th upload of the input parameters. The second stage represents the time required to execute the request after it is spawned. Note that more seekers can spawn requests on each provider. Thus, the providers' offered load (λ_i in Figure 2) is the result of the joint request pattern of all the seekers. 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, and is denoted by θ_i. The delay on the i-th pipe is thus

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

The model firstly provides closed form expressions for the expected execution time at each provider (ER_i), by deriving the expressions for the expected delays of the three stages, EB_i, ED_i, $E\theta_i$, respectively. Then the expected service time $ER(m)$ is derived, accounting for the possibility that less than m executions be generated. The optimal number of allowed replicas is thus $m_{opt} = \arg\min_{1 \le m \le M}\{ER(m)\}$.

5 Analysis

Before presenting the detailed analytical results, here are the assumptions under which they are derived.

A1. The tagged seeker (provider) encounters any *specific* provider (seeker) at the same rate at which it encounters any given node in the network. Furthermore, the encounter process between the seeker (provider) and any specific provider (seeker) is memoryless with respect to the sequence of previously encountered nodes, contact and inter-contact times.

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.

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 can actually be relaxed at the cost of a drastic increase of the model's complexity (see [12]). Assumption A5 is reported here for avoiding even simple inaccuracies in the mathematical derivations. If this assumption is dropped, the resulting inaccuracy is very limited, as discussed in [13][2]. 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. However, to have tractable analytical expressions, we compute $R(m)$ as the minimum over a set of *exponentially* distributed, independent r.v., with average ER_i. We assess the approximation level of this choice through validation with simulation results in Section 6.1. Detailed discussion and derivations of the following results are available in [13].

As a pre-requisite for the expressions of the average delays, it is necessary to derive the average time required by a tagged seeker to encounter a provider in a generic set $\mathcal{A}$ out of all providers, either starting from a random point in time or from the end of a contact time between the seeker and any node. These figures are denoted by $ET(\mathcal{A})$ and $EL(\mathcal{A})$, respectively, and their expressions are provided in [13]. Exploiting assumption A1, it is easy to show that they just depend on the cardinality of $\mathcal{A}$, and therefore are represented by $ET(|\mathcal{A}|)$ and $EL(|\mathcal{A}|)$.

The closed form expression of the delay of the first stage of pipe i (B_i) can be derived as follows. 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 $ET(M)$ seconds, as M is the total number of providers. After spawning each replica, the seeker has to meet a *new* provider on which a replica has not already been spawned. This occurs, on average, after $EL(M-k)$ seconds where k is the number of replicas already spawned. The execution on the i-th provider starts after a time interval with average $ET(M) + iEc + \sum_{k=1}^{i-1} EL(M-k)$. The closed form expression follows after routine manipulations:

$$\begin{aligned} EB_i &\simeq \tfrac{Ec+Et}{p^{(p)}} - \tfrac{EcEt}{Ec+Et} + \\ &+ \; N\,(Ec+Et)\ln\tfrac{M-1}{M-i},\; i<M \end{aligned} \quad (2)$$

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

where γ is the Euler constant, Ec and Et are the average contact and inter-contact times.

As shown in Figure 2 we model the second stage of the pipes 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 $1/\mu$. Therefore, it is possible to model the second stage of the pipes as an $\mathrm{M}^{[X]}/\mathrm{M}/1$ batch arrivals system, whose average delay is as follows:

$$ED = \frac{1}{\mu - \frac{\lambda p^{(s)} m^*}{p^{(p)}}} + \frac{2\frac{\lambda m^*}{p^{(p)}}(Ec+Et)+1}{2\left(\mu - \frac{\lambda p^{(s)} m^*}{p^{(p)}}\right)}. \quad (3)$$

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 $\frac{\lambda p^{(s)} m^*}{p^{(p)}} < \mu$ holds. This condition confirms that replicating too aggressively saturates the providers. 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.

The average delay of the third stage can be derived as follows. Under assumptions A2 and A3, the time required to download a request is the time required to meet the provider, plus a contact time. As i) a tagged seeker meets any provider with the same probability (according to assumption A1), and ii) the delay of the third stage starts when the execution of the request has been finished by the provider, the time to meet the sought provider is $ET(1)$, and the average delay of the third stage is:

$$E\theta = ET(1) + Ec. \quad (4)$$

The above results allow us to derive the optimal value of m, i.e. m_{opt}. We denote with $p_i(m^*)$ the probability that the seeker spawns only i replicas, $i = 1, \dots, m$, and with $EH_i(m^*)$ the expected service time under the condition that just i replicas are spawned. The expected service time when a maximum of m replicas are allowed can thus be computed as $ER(m) = \sum_{i=1}^{m} p_i(m^*) \cdot EH_i(m^*)$ (the expressions for $p_i(m^*)$ and $EH_i(m^*)$ are derived in [13]). The optimal value for the maximum allowed number of replications is thus:

$$m_{opt} = \arg \min_{m=1,\dots,M} \left\{ \sum_{i=1}^{m} p_i(m^*) \cdot EH_i(m^*) \right\} . \quad (5)$$

For any value of m, m^* can be found by solving the fixed point equation $m^* = \sum_{i=1}^{m} i \cdot p_i(m^*)$. From a practical standpoint, the only complexity of the model is solving a fixed point equation, which turns out much more efficient than using alternative evaluation tools (such as, for example, simulations), for a large range of relevant system's parameters.

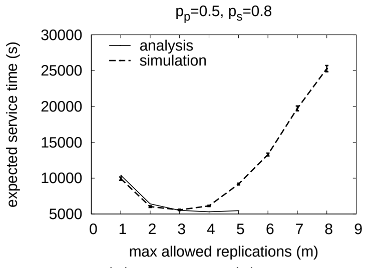
6 Performance evaluation

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 is computed according to the model.

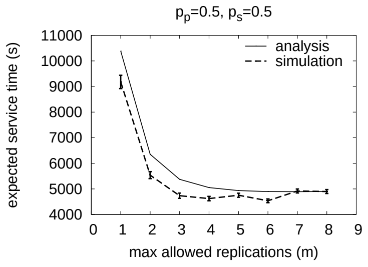
In order to validate the analytical model, we also developed a simulation model based on the OMNeT++

2. Reference [13] is also provided as supplemental material.

simulator. In the simulated environment the nodes move in a square, according to the RWP mobility model, with the modifications described in [14] to guarantee stationarity (the nodes' average speed is 1.5 m/s, representative of walking speeds). 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. Finally, computation times at providers are exponential with a configurable rate. We repeated the simulation runs with increasing values of m. At the end of each run, we computed the corresponding average service time (with 95% confidence intervals), and we identify the optimal case as the value of m achieving the minimum average service time. Our studies show a very good agreement between the simulation and the analysis, which allows us to conclude that the analytical model is accurate. Note 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) makes it practically impossible to explore the system's behavior through simulation over a large range of key parameters, as highlighted in the following.



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



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

Fig. 3. Representative cases of the expected service time as a function of m. Both analytical (solid) and simulation (dashed) curves are shown.

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 $\lambda = 0.005$ 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 3(a) allows us to highlight two important aspects. First of all, the existence of a trade-off that manifests itself as the number of replicas increases, and of a corresponding optimal operating point. The analytical curve stops when the system enters in the saturated region, as the expected service time is infinite beyond this point. In the simulations, after this point the service time is clearly not stationary anymore, and therefore we plot the average over the service requests actually completed within the simulation time, to give an indication of the trend of the service time. Secondly, our results show that the analytical model is able to capture the trend of the simulation results up to the saturation point. 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 3(b) shows a case in which the system never saturates. The optimal and the greedy policies outperform the single policy, and produce almost equivalent expected service times. However the optimal policy is more efficient than the greedy policy, as it generates less executions, and thus avoids useless resource consumption. In cases such as that represented in Figure 3(b), the plot of the expected service time flattens out after a certain value of m, implying that 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 of the flat shape of the expected service time curves. The analytical model predicts with good accuracy the simulation results with respect to the expected service time.

In general, results presented hereafter show that the optimal policy adapts to the system's configuration, converging to the greedy or the single policy when

TABLE 1
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

appropriate, and achieving lower expected service times when neither of them is the best policy.

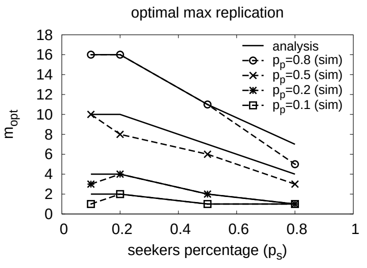
6.2 Performance in sparse scenarios

In this section a sparse network with inter-contact time of 10 minutes and contact time of 16s is considered. The values used to obtain this scenario are shown in Table 1.

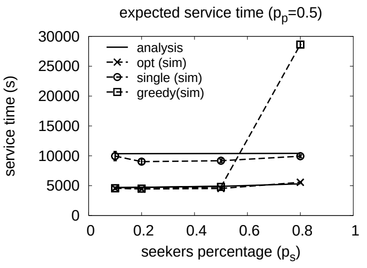
Figure 4(a) shows the optimal value of the maximum allowed number of replicas for different values of seeker and provider percentages. When the percentage of seekers is small, the optimal policy is the greedy one, i.e. m is equal to $M = p^{(p)}N$. This is because the load generated by such small number 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. In this scenario, the single policy is optimal just when the number of seekers is high (beyond 50%), and the number of providers is low (below 20%), as the number of providers is so low that even replicating requests more than once results in significant congestion. Finally, 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. Figure 4(a) confirms that the optimal policy autonomically switches either to the greedy or the single policy when appropriate, or works in between these two extremes.

It is indeed counter-intuitive that, in the simulation results, m_{opt} increases when moving from $p^{(s)} = 0.1$ to $p^{(s)} = 0.2$ both for $p^{(p)} = 0.1$ 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 replicas 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 by the optimal and greedy policies 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 4(b) shows the expected service time for the three policies in the case of $p^{(p)} = 0.5$ (similar remarks



(a) m_{opt}



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

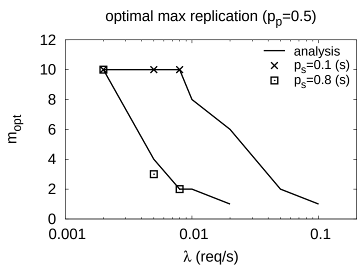
Fig. 4. Performance of the three policies in sparse scenarios. For each configuration, both analytical (solid) and simulation (dashed) curves are presented. Analytical points are not plotted for saturated conditions.

can be done with respect to the other values of $p^{(p)}$, as well). 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. 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. For $p^{(s)} \leq 0.5$, 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. On the contrary, for $p^{(s)} > 0.5$ the optimal policy significantly outperforms both the single and the greedy policies. The optimal policy results in just a slight increase of the expected service time as the number of seekers increases. By jointly looking at plots in Figure 4(a) and (b) it is clear 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.

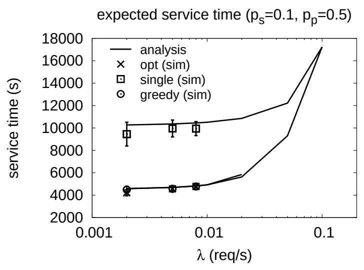
6.3 Performance with varying request loads

In this section we study the system's sensitiveness with respect to the request load (λ)[3], i.e. for λ=0.002, 0.005,

3. Qualitatively similar results are obtained also for varying computation loads, as shown in [13].



(a) m_{opt}



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

Fig. 5. Performance for varying request loads (λ).

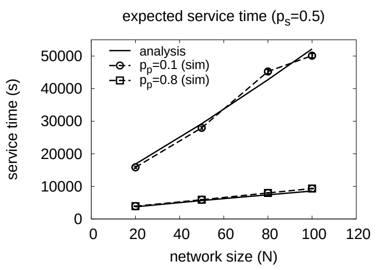


Fig. 6. Optimal service time as a function of N.

0.01, 0.02, 0.05, 0.1 req/s. Figure 5(a) shows the optimal number of replicas 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 5(b) shows the expected service time of the optimal, single and greedy policies for $p^{(s)} = 0.1$ and $p^{(p)} = 0.5$. All the other parameters are as in Table 1. Simulation results are presented for small values of λ only (up to $\lambda = 0.008$), as beyond this point they are practically unmanageable. 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 5(a) and Figure 5(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 5(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. Finally, the optimal policy converges to the single policy for very high load (at $\lambda = 0.1$). 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 5(a)) there is basically no policy that can avoid saturation.

6.4 Performance with varying number of nodes

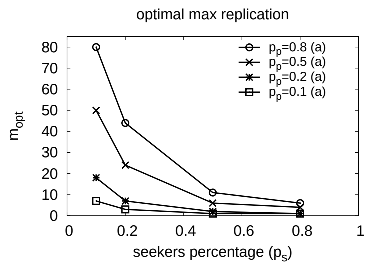
In this section we study the behavior of the system for an increasing number of nodes N, at constant density. We specifically focus on the case of $p^{(s)} = 0.5$ (Figure 6), which highlights general properties of the problem. A more complete set of results is available in [13].

It is quite clear that there is a linear increase of the expected optimal service time with N (while m_{opt} is found to be basically independent of N, see [13]). This result can be justified by looking at the analytical model presented earlier. Recall that the delay on each pipe R_i is made up of the delay of three stages. The delay of the first stage (EB_i, Equation 2) is composed by a part that is independent of N, and a part that is linear with N. The delay of the second stage (ED, Equation 3) can be shown to be slightly dependent on N, as the only component which depends on N (m^*) shows just a sublinear increase with N. Finally, the delay of the third stage ($E\theta$, Equation 4) depends on the expected time to meet a *specific* node (the provider corresponding to the pipe), $ET(1)$. By inspecting this expression it appears that it linearly depends on N. We can thus write ER_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 show that EY_i can 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 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 results that $ER(m_{opt}) \approx E\theta \frac{N}{m_{opt}}$ holds true. This confirms the linear dependence of $ER(m_{opt})$ with N found in Figure 6(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 $p^{(p)}$).

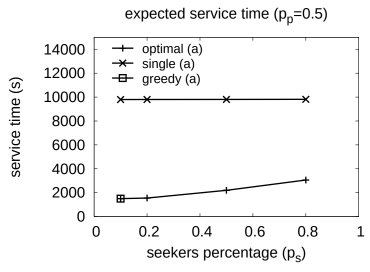
6.5 Performance in dense scenarios

Although the main reference scenario of this work are sparse opportunistic networks, we explore in this section how the system behaves when the network becomes dense, by increasing the number of nodes to 200. This results in decreasing the average inter-contact time to about 35s, while the average contact time is still in the order of 15s.

Figures 7 show the optimal number of allowed replicas and the expected service time (for $p^{(p)} = 0.5$). When the



(a) m_{opt}



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

Fig. 7. Performance in a dense scenario ($N = 200$).

number of nodes increases, 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 $p^{(p)} = 0.5$). For the rest of the configurations reported in Figure 7, the greedy policy brings the system in the saturated region. By recalling that the load on each provider is defined by $\frac{\lambda p^{(s)} m^*}{p^{(p)}}$, 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.

7 Conclusions

In this paper, we investigate service provisioning in an opportunistic networking environment, where multi-hop communication is accomplished through a series of opportunistic contacts, rather than through continuous multi-hop paths. 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 results assessing the performance of a system that replicates executions according to the model, in comparison with other reference policies. The developed analytical model is validated through simulations, and used to characterize the system performance with respect to a number of parameters - number of nodes, number of seekers, number of providers, and request load. In all investigated cases, 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.

Acknowledgments

This work was partially funded by the European Commission under the FP7 SCAMPI (258414) Project, and by the US National Science Foundation under the CSR 0834493 Project. Authors would like to acknowledge the anonymous reviewers for their constructive comments.

References

[1] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic Networking: data forwarding in disconnected mobile ad hoc networks," *IEEE Comm. Mag.*, vol. 44, no. 11, 2006.
[2] M. Conti and M. Kumar, "Opportunities in opportunistic computing," *IEEE Computer*, vol. 43, no. 1, pp. 42–50, Jan. 2010.
[3] M. Papazoglou, "Service Oriented Computing: Concepts, characteristics and directions," in *IEEE WISE*, 2003.
[4] M. Ravi, P. Stern, N. Desai, and L. Iftode, "Accessing Ubiquitous Services using Smart Phones," in *IEEE PerCom*, 2005.
[5] S. Helal, N. Desai, V. Verma, and C. Lee, "Konark - a service discovery and delivery protocol for ad-hoc network," in *IEEE WCNC*, 2003.
[6] S. Kalasapur, M. Kumar, and B. A. Shirazi, "Dynamic Service Composition in Pervasive Computing," *IEEE TPDS*, vol. 18, no. 7, pp. 907 – 918, 2007.
[7] 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.
[8] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Efficient Routing in Intermittently Connected Mobile Networks: The Multiple-copy Case," *IEEE Trans. on Net.*, 2008.
[9] 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.
[10] 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. 54, no. 5, pp. 589–604, March 2009.
[11] 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.
[12] A. Passarella, M. Kumar, M. Conti, and E. Borgia, "Exploiting opportunistic contacts for service provisioning in bandwidth limited opportunistic networks," in *IIT TR-18/2010, available at http://bruno1.iit.cnr.it/~andrea/tr/services-tr-bw.pdf*, 2010.
[13] ——, "Exploiting opportunistic contacts for service provisioning in pervasive environments," in *IIT TR-19/2010, available at http://bruno1.iit.cnr.it/~andrea/tr/services-tr.pdf*, 2010.
[14] W. Navidi and T. Camp, "Stationary distributions for the random waypoint mobility model," *IEEE Trans. Mob. Comp.*, vol. 3, no. 1, pp. 99–108, 2004.

Andrea Passarella (PhD in Computer Engineering ’05) is a Researcher of the Institute for Informatics and Telematics (IIT) of the National Research Council (CNR), Italy. Prior to joining IIT, he was a Research Associate at the Computer Laboratory, Cambridge, UK. His current research focuses on opportunistic and mobile social networks, with emphasis on content-centric architectures, services, routing protocols, mobility models. He published more than 60 research papers in the areas of opportunistic, ad hoc and sensor networks. He is Program Co-Chair for IEEE WoWMoM 2011, and in the PC, among others, of IEEE MASS and PerCom. He was Co-Chair of IEEE AOC 2009, TPC Co-Chair of ACM MobiOpp 2007, Vice-Chair for ACM REALMAN (2005/06), and IEEE MDC (2006). He was Workshops Co-Chair for IEEE PerCom and WoWMom 2010. He is in the Editorial Board of Elsevier Pervasive and Mobile Computing and Inderscience IJAACS Journals. He served as Guest Co-Editor for special sections on ACM Mobile Computing and Communications Review, Elsevier Pervasive and Mobile Computing Journal, Elsevier Computer Communications Journal, Adhoc & Sensor Wireless Networks.

Mohan Kumar received the BE (1982) degree from Bangalore University in India and the MTech (1985) and PhD (1992) degrees from the Indian Institute of Science. He is a professor in computer science and engineering at the University of Texas at Arlington. Prior to joining the University of Texas at Arlington in 2001, he held faculty positions at the Curtin University of Technology, Perth, Australia (1992-2000), the Indian Institute of Science (1986-1992), and Bangalore University (1985-1986). His current research interests are in pervasive and mobile computing, opportunistic networking and computing, sensor networks and distributed computing. He has published more than 150 articles in refereed journals and conference proceedings and supervised several doctoral dissertations and masters theses in the above areas. He is one of the founding editors of the Pervasive and Mobile Computing Journal and is one of the area editors of computer communications. He has guest coedited special issues of several leading international journals. He is a senior member of the IEEE. He is a cofounder of the IEEE International Conference on Pervasive Computing and Communications (PerCom), where he served as program chair (2003) and general chair (2005). He has also served on the technical program committees of numerous international conferences/workshops. Recently, he has developed or codeveloped algorithms/methods for service composition in pervasive environments, information acquisition, dissemination and fusion in pervasive and sensor systems, and caching and prefetching in mobile, distributed, pervasive, and P2P systems.

Marco Conti is a research director at the Institute of Informatics and Telematics (IIT), an institute of the Italian National Research Council (CNR). He published in journals and conference proceedings more than 250 research papers related to design, modelling, and performance evaluation of computer-network architectures and protocols. He co-authored the book “Metropolitan Area Networks” (1997) and is co-editor of the books “Mobile Ad Hoc Networking” (2004) and “Mobile Ad Hoc Networks: From Theory to Reality” (2007). He is the chair of the IFIP WG 6.3 “Performance of Communication Systems”. He is the Editor-in-chief of the Computer Communications Journal and Associate Editor-in-chief of Pervasive and Mobile Computing Journal; he is on the editorial board of IEEE Transactions on Mobile Computing, Ad Hoc Networks, Journal of Communications Systems, and Wireless Ad Hoc and Sensor Networks. He served as general chair of ACM REALMAN 2006 and IEEE MASS 2007, and as general Co-chair of IEEE WoWMoM 2006, ACM MobiOpp 2007, and IEEE PerCom 2010. He served as TPC chair of IEEE PerCom 2006, and of the IFIP-TC6 conferences Networking 2002 and PWC 2003, and as TPC Co-chair of ACM WoWMoM 2002, WiOpt 2004, IEEE WoWMoM 2005, ACM MobiHoc 2006 and ACM MobiOpp 2010.

Eleonora Borgia received the Laurea and the Ph.D degree in Computer Engineering both from the University of Pisa, Italy, in 2002 and 2006, respectively. She is currently a Researcher in the Ubiquitous Group at Institute for Informatics and Telematics (IIT-CNR, Italy). Her research activities are focused on wireless and pervasive computing addressing issues related to service provisioning for opportunistic networks and data dissemination for sensor networks. Before, she also worked on routing algorithms and MAC protocols for ad hoc networks. She served in the technical program committee of several international conferences such as ACM/IEEE ICPS 2010, IEEE CCNC (since 2009), and for several international workshops such as IEEE AOC(since 2009), IEEE PerSeNS 2010 and 2011, ACM PM2HW2N (since 2009) and ACM/SIGMOBILE REALMAN 2006. She was also in the organizing committee of international conferences and workshops serving as Publicity Chair and Publication Co-Chair.