Performance Evaluation of Service Execution in Opportunistic Computing*

Andrea Passarella, Marco Conti, Elonora Borgia
IIT-CNR
Via G. Moruzzi, 1
56124 Pisa, Italy
{a.passarella,m.conti,e.borgia}@iit.cnr.it

Mohan Kumar
University of Texas at Arlington
Arlington, TX 76019, USA
mkumar@uta.edu

ABSTRACT

Opportunistic computing has emerged as a new paradigm in computing, leveraging the advances in pervasive computing and opportunistic networking. Nodes in an opportunistic network avail of each others' connectivity and mobility to overcome network partitions. In opportunistic computing, this concept is generalised, as nodes avail of *any* resource available in the environment. Here we focus on computational resources, assuming mobile nodes opportunistically invoke services on each other. Specifically, resources are abstracted as services contributed by providers and invoked by seekers. In this paper, we present an analytical model that depicts the service invocation process between seekers and providers. Specifically, we derive the optimal number of replicas to be spawned on encountered nodes, in order to minimise the execution time and optimise the computational and bandwidth resources used. Performance results show that a policy operating in the optimal configuration largely outperforms policies that do not consider resource constraints.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Communication*; C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Performance, Design, Algorithms

Keywords

Opportunistic networks, service provisioning, performance evaluation, analytical modelling

*This work was partially funded by the European Commission under the SCAMPI (258414) FIRE Project and by the US National Science Foundations award CNS-0834493.

MSWiM'10, October 17–21, 2010, Bodrum, Turkey.

1. INTRODUCTION

Opportunistic computing [5] is a recently proposed mobile computing paradigm, blending the areas of pervasive computing and opportunistic networking. In opportunistic networks, nodes exploit opportunistic contacts to forward messages and distribute content through encountered peers, and contribute and avail of each others' resources in terms of connectivity and temporary buffers. Opportunistic computing generalises this concept, considering the possibility for nodes to opportunistically avail of each others' *general resources*, including, but not limiting to, connectivity and storage. In the context of pervasive computing, resources may 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 schemes for opportunistic computing. The opportunistic computing paradigm enables several interesting applications that are already being investigated in the areas of participatory sensing, pervasive healthcare, intelligent transportation systems, and crisis management, as discussed in [5, 7].

In this view, it is natural to abstract resources as *services* that are contributed by *providers* and invoked by *seekers*. While service-oriented computing has long been investigated in the conventional Internet, in single hop wireless networks (e.g., WLANs or cellular networks) and in conventional MANET (e.g., [6]), it is definitely a novel research area in opportunistic networks. Intermittent connectivity and severely unstable topologies typical of opportunistic networks make conventional service-oriented approaches too cumbersome to be used. Indeed, research on opportunistic networks has mainly focused on routing issues (e.g., [2, 11]), mobility analysis (e.g., [4]) and data-centric architectures (e.g., [3]).

To the best of our knowledge, this paper is one of the first attempts to tackle key research challenges posed by the opportunistic computing concept. Specifically, we develop a detailed analytical model of the service invocation process between seekers and providers. As better described in Section 2, we assume the service invocation and provisioning is carried out as follows: Whenever a seeker has a request for a service execution, it waits to encounter possible providers. When a provider is met, a service execution may be spawned, according to the *replication policy* used by the seeker. Service results are gathered by the seeker from the first provider encountered which has completed the service execution. The

goal of the model is to identify the optimal operating point of the replication policy, in which the expected service time (i.e., the time interval between when a request is generated at the seeker and when the seeker receives the output results) is minimised, subject to the available providers' resources in terms of computation and bandwidth. The model presented in this paper significantly extends an initial model we have described in [10]. Unlike the model in [10] here we consider i) limited bandwidth, ii) heterogeneous mobility processes of different nodes, iii) heterogeneous providers computation capabilities, and iv) varying seekers generation rates. Modelling resource limitations (particularly in terms of bandwidth) is very challenging in opportunistic networks, and indeed it is common practice to assume unlimited bandwidth scenarios, to simplify the analysis (see, e.g., [11]).

The model, described in Section 3, is validated against simulation results to assess its accuracy, and then used to characterise the performance of the optimal, resource aware, replication policy (Section 4). Specifically, comparison of the optimal policy with resource-blind policies clearly shows that the latter leads to saturation or under utilisation of available resources. The performance analysis also characterises the behaviour of the optimal policy as a function of the overall load in terms of service requests. We find that the optimal policy is able to tolerate increasing loads by limiting the corresponding increase of service time, up to the point where the resources become inevitably saturated. Beyond this point, no alternative policy can perform better, as in these cases the only solution would be to drop some service requests without even attempting to serve them. Investigating such "back-pressure" policies is out of the scope of this paper, and subject of future work.

2. SERVICE INVOCATION IN OPPORTUNISTIC COMPUTING ENVIRONMENTS

Whenever a seeker needing a service encounters a service provider, the service may be spawned on the latter. The execution process entails three stages, i.e., uploading the input parameters, executing the service, and downloading the output results. This process may take several contact times. While this process is ongoing, the seeker may encounter a different provider, and therefore an opportunity arises to spawn an additional service execution in parallel. Whether a new parallel execution is spawned or not, when an opportunity arises, depends on the *replication policy* implemented by the seeker. A service is complete when the seeker downloads the output results from any of the providers running in parallel.

Developing this service provisioning model in an efficient manner requires investigations into several challenging research questions. In opportunistic computing environments it is particularly important to optimise the use of the resources, which are typically contributed by mobile users' devices. In a resource unlimited environment the best option would be to replicate service executions onto all encountered providers. Spawning more replicas in parallel on multiple providers may appear to reduce expected service execution time (the time required by the seeker to receive results), as results can be collected from any of the providers. However, when resource constraints are considered, uncontrolled replication may lead to resource saturation and thus

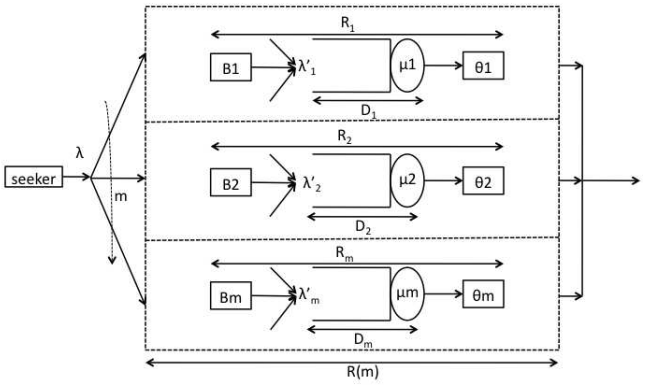

Figure 1: Scheme of the replication process

to exponential increase of the expected service time, as we demonstrate in the remainder of this paper.

The goal of this paper is therefore to characterise through an analytical model the expected service time as a function of the number of spawned replicas, considering two key resource limitations: the computational capabilities of the providers, and the bandwidth limitations between provider-seeker pairs. The model allows us to derive the optimal operating point for the replication policy, i.e., the number of replicas that should be spawned to minimise the expected service execution time. This also corresponds to the optimal use of the computational and bandwidth resources.

3. ANALYTICAL MODEL

In the model we consider a specific service S_j available on M_j providers, and analyse the service execution time as a function of the number of replicas to be spawned by seekers, referred to as m ($\leq M_j$). We focus on a tagged seeker, and derive the expected service time when m executions are spawned on different providers, hereafter referred to as $E[R_j(m)]$. The optimal operating point is the value of m that minimises $E[R_j(m)]$.

The rationale of the model derivation is captured in Figure 1 (to simplify the notation, we omit index j in the figure). We assume that each service S_j's seeker issues requests according to a Poisson process with rate λ_j, and that there are k_j such seekers. The total request rate for service S_j is $\lambda_j k_j$. Each of the m replicas is represented with an horizontal pipe in Figure 1.

Each pipe consists of three stages. The first stage represents the time required to complete the i-th upload of the input parameters, and is referred to as B_{ji}. The second stage represents the time required to execute the request after it is spawned, and is referred to as D_{ji}. The third stage represents the time required by the seeker to complete the download of the output results, and is referred to as θ_{ji}. The delay on the i-th pipe is thus $R_{ji} = B_{ji} + D_{ji} + \theta_{ji}$, and the service time experienced by a seeker is $R_j(m) = \min_{i=1,\ldots,m}\{R_{ji}\}$.

The offered load on the second stage of pipe i (λ'_{ji}), can be evaluated as follows. The total offered load for service S_j is $\lambda_j k_j m$. This rate is split among M_j providers. By denoting with f_{ji} the percentage of executions of service S_j spawned on provider i, the offered load on the second stage of pipe i is $\lambda_j k_j m f_{ji}$.

Bandwidth limitations affect the delays of the first and third stage, while computational limitations affect the delay of the second stage. A contact event might not be long

enough to upload all service S_j's input parameters and/or download all output results. In such cases, uploads/downloads are resumed at the next contact opportunity with the same provider.

Hereafter, we separately analyse the delays of the three stages, and then derive the service time $R_j(m)$. In general, the distributions of the random variables (r.v.) R_{ji} are fairly complex, and therefore it is possible to provide a closed form expression for their minimum $R_j(m)$ only under particular assumptions. Specifically, we will fully describe the case in which the r.v. R_{ji} are assumed to be exponential. Although in the following we are able to provide a more precise characterisation of some R_{ji} components, note that, under the assumption that the r.v. R_{ji} are exponential, it is sufficient to derive the *average* values of the delays of the stages in order to fully characterise R_{ji}.

First, we present a general concept used extensively in the analysis. Whenever a new request for service S_j is issued by the seeker, the seeker has to spawn m replicas. The execution of a replica on a provider actually starts when the corresponding input parameters are *completely* uploaded on the provider. Therefore, the m replicas are spawned *on the first m providers (out of M_j) on which input parameters are completely received*. As uploads may require several contacts to complete, the providers on which the replicas are executed are not necessarily the first m providers encountered by the seeker after the request is generated. Therefore, pipe i corresponds to the provider on which the i-th upload completes, starting from the point in time when the request is issued at the seeker. In other words, if the set of r.v. $\{\beta_l\}_{l=1,\ldots,M}$ denotes the time intervals needed by the seeker to upload the input parameters on providers starting from the point in time when the request is generated, pipe i corresponds to provider $\hat{l}$ iff $\beta_{\hat{l}}$ is the i-th shorter time in the set $\{\beta_l\}$.

For ease of notation in the analysis, without loss of generality, we omit the subscript when representing a service. As a final remark, due to space reasons, we provide hereafter a concise description of the analytical details, preferring to focus more on the rationale behind, and the meaning of the analytical derivations. The detailed proofs of the lemmas and theorems are omitted, and can be found in [9].

3.1 Model of the contact process between nodes

Before analysing the delay of the three stages of the pipes we provide a general result, which is instrumental to the following parts of the analysis. We focus on a tagged node, and analyse the time it requires to encounter any node in a given subset of the network, starting from a random point in time. We denote as success the event by which the tagged node finds any node belonging to the subset, and as T the time for success, starting from a random point in time. This general result is used to model the time required: i) by a seeker to meet possible providers; ii) by a tagged provider to meet one of the seekers; and iii) by a tagged provider to meet a particular seeker after one of its services has been executed.

We assume that inter-contact times (contact times) between the tagged node and any other nodes in the subset can be divided in two classes: The first class includes inter-contact (contact) times after which a success occurs, the second class inter-contact (contact) times after which a failure occurs. We assume that inter-contact (contact) times within each class are independent and identically distributed (iid),

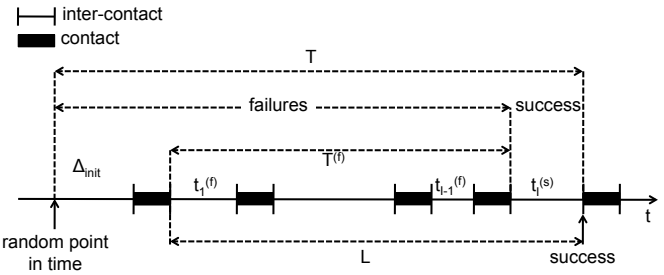

Figure 2: Scheme of the general contact process among nodes.

that contact and inter-contact times are mutually independent, and that the next contact of the tagged node is independent of the previous contacts. Inter-contact (contact) times of different classes may have different, arbitrary distributions. This is a more general setting with respect to typical opportunistic networks modelling assumptions, where inter-contact (contact) times are assumed iid and following an exponential distribution, at least in the tail (e.g., [11]).

In general, we can represent the process as in Figure 2. To derive $E[T]$ we condition on the specific point in the mobility process of the tagged node from where we start measuring T (remember that T starts at a random point in time), and apply the law of total probability. The starting point of T may fall i) during a contact time with one of the sought nodes; ii) during a contact time with a node not in the sought subset; iii) during an inter-contact time after which a success occurs; iv) during an inter-contact time after which a failure occurs. These events will be denoted by $C^{(s)}$, $C^{(f)}$, $IC^{(s)}$ and $IC^{(f)}$, and their probabilities by $p_c^{(s)}$, $p_c^{(f)}$, $p_{ic}^{(s)}$ and $p_{ic}^{(f)}$, respectively. Those probabilities can be derived with routine analysis as shown in [9], and are functions of the average inter-contact and contact times of the two classes (success, failure), throughout referred to as $E[c^{(s)}]$, $E[c^{(f)}]$, $E[t^{(s)}]$, and $E[t^{(f)}]$, respectively, and of the probability that an inter-contact time results in (a contact time is) a success, p_s. As in case i) T is clearly 0, $E[T]$ can be written as

$$E[T] = p_c^{(f)} E\left[T|C^{(f)}\right] + p_{ic}^{(s)} E\left[T|IC^{(s)}\right] + p_{ic}^{(f)} E\left[T|IC^{(f)}\right]. \quad (1)$$

The expression of $E\left[T|IC^{(s)}\right]$ can be derived by noting that in case iii), T is the residual inter-contact time after which a success occurs (denoted as $t_+^{(s)}$):

$$E\left[T|IC^{(s)}\right] = E\left[t_+^{(s)}\right] = \frac{E\left[t^{(s)}\right]}{2} + \frac{Var\left[t^{(s)}\right]}{2E\left[t^{(s)}\right]}.$$

In the remaining cases ii) and iv), we have to account for an initial component $\Delta_{init,k}$, representing the time between the start of T and the end of the following contact time. After $\Delta_{init,k}$ there is a random number $I-1$ (possibly equal to 0) of inter-contact and contact times after which a failure occurs, and a final inter-contact time after which success occurs. As proved in [9], we obtain:

$$\begin{aligned} E[T|IC^{(f)}] &= E[t_+^{(f)}] + \frac{E[t^{(f)}] + E[c^{(f)}]}{p_s} - E[t^{(f)}] + E[t^{(s)}] \\ E[T|C^{(f)}] &= E[c_+^{(f)}] + \left(\frac{1}{p_s} - 1\right)\left(E[t^{(f)}] + E[c^{(f)}]\right) + E[t^{(s)}]. \end{aligned}$$

In the following analysis, in addition to T, we will also need the time to success starting from the end of a contact, referred to as L. It is easy to see that $E[L]$ can be written

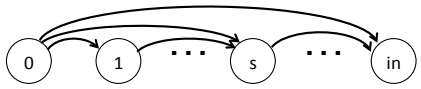

Figure 3: Embedded process for input parameters upload.

as follows:

$$E[L] = \left(\frac{1}{p_s} - 1\right)\left(E[t^{(f)}] + E[c^{(f)}]\right) + E[t^{(s)}]. \quad (2)$$

Note that $E[T]$ and $E[L]$ depend only on the average inter-contact and contact times, and on the probability of an inter-contact time resulting in (a contact time being) a success (p_s), i.e., on the properties of the mobility process and on the probability of encountering one of the sought nodes.

3.2 Analysis of the first stage

In order to model the delay of the first stages $\{B_i\}_{i=1,\ldots,m}$, we first analyse the time required by the tagged seeker to upload the input parameters to any individual provider from the point in time when the request is generated. Recall that these time intervals are denoted as $\{\beta_l\}_{l=1,\ldots,M}$. As discussed earlier, B_i is the i-th shortest element in $\{\beta_l\}$.

The figure β_l can be modelled as an M/G/1 queue. Requests are generated at the seeker according to a Poisson process with rate λ. The service time of the queue ($\hat{\beta}_l$) includes the time to meet provider l enough times to upload the input parameters of the service. Therefore, it accounts both for the inter-contact process between the seeker and the provider, and the bandwidth constraints. Let us denote by T_l and L_l the time required by the seeker to meet provider l, respectively, starting from an arbitrary point in time or from the end of a contact (as shown in Section 3.1), and by q_l the number of contact times needed to complete the input parameters upload. Then, $E[\beta_l]$ is provided in Lemma 1.

LEMMA 1. *The average delay to complete the upload of the input parameters on provider l is*

$$E[\beta_l] = \frac{E\left[\hat{\beta}_l\right]}{1 - \lambda E\left[\hat{\beta}_l\right]}. \quad (3)$$

where $E[\hat{\beta}_l]$ is

$$E\left[\hat{\beta}_l\right] = \frac{E[T_l] - E[L_l] + E[q_l]\left(E[L_l] + E\left[c^{(s)}\right]\right)}{\frac{M}{m} - \lambda\left(E[L_l] - E[T_l]\right)}. \quad (4)$$

The figure q_l accounts for the bandwidth limitations, and is derived as follows. Let us denote with *in* the r.v. representing the size of the input parameters to be uploaded. We consider an embedded Markov process, whose state is the number of bytes already uploaded to provider l, and whose transitions occur upon each contact between the seeker and the provider (see Figure 3).

Assuming that uploads always start at the beginning of a contact, the chain always starts in state 0 at the beginning of the first contact. Upon a contact, the chain moves from the state s to the state $s+h$ if h bytes can be uploaded, i.e., with a probability equal to $p\left(h \le b \cdot c^{(s)} < h+1\right)$ where b is the bandwidth available during a contact. The number of contacts required to complete the upload process is thus the hitting time on state *in*, starting from state 0.

The line of reasoning for completing the analysis of B_i is as follows. Conditioning on the fact that the i-th ordered shortest times in the set $\{\beta_l\}$ is known, say $\{\beta_{k1}, \ldots \beta_{ki}\}$, then B_{i+1} is the minimum over the rest of the variables in $\{\beta_l\}$. Assuming that the random variables (r.v.) β_l are exponential and independent, we can provide closed form expressions both for the probability of the set $\{\beta_{k1}, \ldots \beta_{ki}\}$ being the i-th shortest r.v. in $\{\beta_l\}$, and for the distribution (and average value) of B_{i+1}. This is derived in Lemma 2 and Theorem 1. As a matter of notation, in the following $\phi_l = 1/E[\beta_l]$.

LEMMA 2. *The probability of the set $\{\beta_{k1}, \ldots \beta_{ki}\}$ being the i-th ordered shortest variables in the set $\{\beta_l\}_{l=1,\ldots,M}$ is*

$$P(\beta_{k1}, \ldots, \beta_{ki}) = \frac{\prod_{p=1}^{i} \phi_{kp}}{\sum_{p=1}^{M} \phi_p \sum_{\substack{p=1 \\ p\neq k1}}^{M} \phi_p \cdots \sum_{\substack{p=1 \\ p\neq k1,\ldots,k(i-1)}}^{M} \phi_p} \quad (5)$$

THEOREM 1. *The time required by a seeker to complete the upload of the input parameters on provider $i+1$ ($i = 0, \ldots, m-1$) is distributed as follows:*

$$B_{i+1} \sim \sum_{k1=1}^{M} \cdots \sum_{\substack{ki=1 \\ ki\neq k1,\ldots,k(i-1)}}^{M} P(\beta_{k1}, \ldots, \beta_{ki}) \exp\left(\sum_{\substack{p=1 \\ i\neq k1,\ldots,ki}}^{M} \phi_p\right)$$

and its average value is:

$$E[B_{i+1}] = \sum_{k1=1}^{M} \cdots \sum_{\substack{ki=1 \\ ki\neq ki,\ldots,k(i-1)}}^{M} \frac{P(\beta_{k1}, \ldots, \beta_{ki})}{\sum_{\substack{p=1 \\ p\neq k1,\ldots,ki}}^{M} \phi_p}. \quad (6)$$

Note that the above expression greatly simplifies when the r.v. β_l are also identically distributed, with rate ϕ. In this case, B_{i+1} is exponential with rate $(M-i)\phi$. This is a very intuitive result, as B_{i+1} becomes the minimum over $M-i$ iid exponential random variables with rate ϕ.

3.3 Analysis of the second stage

As shown in Figure 1 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. We assume that the computation time at providers is exponentially distributed with rate $\mu_l, l = 1, \ldots, M$. Considering a random computation time allows us to account for variations in providers capabilities, and variations in computation times of different requests for the service. We assume to know that the generic provider l corresponds to pipe i, and derive the conditioned average delay. By computing the probability of this event, we can apply the law of total probability, and derive the average value of the second stages.

Focusing on a particular provider l, it is possible to model the second stage as a batch arrival system (see, e.g., [12]): A batch of requests arrive at the provider when it meets a seeker, and the size of the batch is the number of (sets of) input parameters for requests queued at the seeker whose upload completes during the contact. As a matter of notation, we describe the second stage as an $M^{[X]}/M/1$ system, where X is the r.v. denoting the size of the batch. Lemma 3 provides the average delay of the second stage of pipe i, assuming provider l corresponds to it, denoted by $E[D_i|l]$. In the formula we use the component $E[L_l^{(P)}]$, denoting the average time for the provider l to meet any seeker starting from

the end of a contact. $E[L_l^{(P)}]$ can be computed as shown in Section 3.1.

LEMMA 3. *The average delay of the second stage of pipe i, assuming provider l corresponds to it is*

$$E[D_i|l] = \frac{E[X_l^2] + 2E[X_l]}{2\mu_l E[X_l](1-\rho_l)}, \quad (7)$$

where X_l is the size of the batches arriving at provider l. Furthermore, the utilisation of the providers is

$$\rho_l = \frac{\lambda_{X,l} E[X_l]}{\mu_l} = \frac{E[X_l]}{\mu_l (E[L_l^{(P)}] + E[c^{(s)}])}, \quad (8)$$

where $\lambda_{X,l}$ is the rate of batch arrivals at provider l.

The result in Lemma 3 confirms our initial intuition about the possibility of saturation of the service provisioning system. The utilisation of providers increases with the batch size and with the number of seekers "using" the provider (which clearly means an increase of $\lambda_{X,l}$). When the utilisation approaches 1, the average delay on the providers tends to infinity.

The probability that provider l corresponds to pipe i is the probability that the time to complete the upload of the input parameters on provider l is the i-th shortest time in the set $\{\beta_l\}_l$, i.e., $P(B_i = \beta_l)$. The analysis of this figure is similar to that related to Lemma 2. Specifically, by exploiting the result of Lemma 2 and recalling that ϕ_l is equal to $1/E[\beta_l]$, we obtain the result in the following Lemma.

LEMMA 4. *The probability of provider l corresponding to pipe $i+1$ $(i = 0, \ldots, m-1)$ is*

$$P(B_{i+1} = \beta_l) = \sum_{\substack{k1=1 \\ k1 \neq l}}^{M} \cdots \sum_{\substack{ki=1 \\ ki \neq k1, \ldots, k(i-1), l}}^{M} \frac{P(\beta_{k1}, \ldots, \beta_{ki})\phi_l}{\sum_{\substack{s=1 \\ s \neq k1, \ldots, ki}}^{M} \phi_s}. \quad (9)$$

Based on Lemmas 3 and 4 it is straightforward to derive the expression of $E[D_i]$.

THEOREM 2. *The average delay to complete the service computation of the i-th replica is*

$$E[D_i] = \sum_{l=1}^{M} E[D_i|l] \, P(B_i = \beta_l), \quad (10)$$

where $E[D_i|l]$ and $P(B_i = \beta_l)$ are as in Equations 7 and 9.

3.4 Analysis of the third stage

The delay of the third stage is the time interval from the end of the execution of a request at the provider, until the point in time when the output results are completely downloaded to the seeker. In general, when a request execution for a particular seeker completes at the provider, output results from previously completed requests for the same seeker might still be waiting to be downloaded. Therefore, we must model the third stage with a queue. The analysis of the third stage is conceptually similar to that of the first stage, albeit for the fact that the roles of the provider and seeker are switched. We model the third stage with an M/G/1 queue, and the service time of the queue is the time for the provider to download the output parameters to the seeker.

We derive θ_i first conditioning on a particular provider l corresponding to pipe i, and then applying the law of total probability. As a matter of notation, we denote with θ_i the delay of the third stage of pipe i, with $\hat{\theta}_i$ the service time of the queue representing the stage, and with $\theta_i|l$ and $\hat{\theta}_i|l$ the same figures conditioned to the fact that the pipe corresponds to provider l. Recalling that we denote by λ_i' the total offered load on pipe i, and by k the number of seekers, Lemma 5 provides the closed form expression for the average delay conditioned to provider l. In the Lemma we denote by T_l and L_l the time required by provider l to meet the seeker (as can be derived by exploiting the analysis in Section 3.1), and by q_l the number of contact events required to download the output results.

LEMMA 5. *The average delay of the third stage of pipe i conditioned to the fact that provider l corresponds to the pipe is*

$$E[\theta_i|l] = \frac{E\left[\hat{\theta}_i|l\right]}{1 - \frac{\lambda_i'}{k} E\left[\hat{\theta}_i|l\right]}, \quad (11)$$

where $E\left[\hat{\theta}_i|l\right]$ is:

$$E\left[\hat{\theta}_i|l\right] = \frac{E[T_l] - E[L_l] + E[q_l]\left(E[L_l] + E\left[c^{(s)}\right]\right)}{m - \frac{\lambda_i'}{k}(E[L_l] - E[T_l])}.$$

Based on Lemmas 5 and 4 it is straightforward to derive the expression of $E[\theta_i]$.

THEOREM 3. *The average delay for the seeker to download the output results of the i-th replica $(i = 1, \ldots, m)$ is*

$$E[\theta_i] = \sum_{l=1}^{M} E[\theta_i|l] \, P(B_i = \beta_l), \quad (12)$$

where $E[\theta_i|l]$ and $P(B_i = \beta_l)$ are as in Equations 11 and 9, respectively.

3.5 Optimal replication

The expected service execution time experienced by the seeker ($R(m)$) is the minimum delay over the m pipes. As analytical expressions for the minimum of a generic set of r.v. are not available, we approximate $R(m)$ assuming that the r.v. R_i are exponentially distributed with rate $\gamma_i = (E[B_i] + E[D_i] + E[\theta_i])^{-1}$. $R(m)$ is then also exponentially distributed, with rate $\Gamma(m) = \sum_{i=1}^{m} \gamma_i$, and average value $E[R(m)] = \Gamma(m)^{-1}$.

In the general case, $E[R(m)]$ can be computed numerically, while closed formulas can be found under additional assumptions. In the following of the section we derive a closed formula of $E[R(m)]$ under the assumption that the r.v. $\hat{\beta}_l$, $D_i|_l$ and $\theta_i|_l$ do not depend on the particular provider. Under these assumptions, the expressions of the average delay of the three stages in Equations 6, 10, 12 become simpler. As far as the second stage is concerned, with respect to Equation 10 all the dependencies on the particular provider disappear. As we discussed in Section 3.2, the average delay of the first stage on pipe i becomes the minimum over $M-i+1$ exponentially distributed r.v. with rate $\phi = 1/E[\beta]$. Recalling the expression of $E[\beta_l]$ in Equation 3 we obtain

$$E[B_i] = \frac{E[\beta]}{M-i+1} = \frac{1}{M-i+1} \cdot \frac{mK_1}{M - m\lambda(K_1+K_2)},$$

where K_1 is $E[T] - E[L] + E[q](E[L] + E[c^{(s)}])$, and K_2 is $E[L] - E[T]$. Note that K_1 and K_2 do not depend either on the particular pipe i nor on the number of replicas m. Finally, after similar routine manipulations, noting that the offered load λ' becomes equal to $\frac{\lambda km}{M}$, the average delay of the third stage becomes

$$E[\theta] = \frac{K_1}{m - \frac{\lambda m}{M}(K_1 + K_2)}.$$

The only term that depends on the particular pipe i is the delay of the first stage, due to the fact that the pipe corresponds to the i-th provider on which the seeker completes the upload of the input parameters.

Finally, we are in a position to derive a closed form expression for the expected service time $E[R(m)] = 1/\Gamma(m)$, as in Theorem 4.

THEOREM 4. *The average time for the seeker to avail of service S_j is*

$$E[R(m)] = \frac{(E[D] + E[\theta])^2}{m(E[D] + E[\theta]) - E[\beta] \ln \frac{Q-1}{Q-m-1}}, \qquad (13)$$

where Q is equal to $\frac{E[\beta]}{ED+E\theta} + M + 1$.

The optimal replication level m_{opt} can be found by minimising Equation 13. This requires specifying the specific dependence of ED on m. Once this is done, m_{opt} can be either found analytically or numerically. The result provided by Theorem 4 is therefore general enough to be customised to the features of any specific scenario.

4. PERFORMANCE RESULTS

In this section we analyse the expected service (execution) time of a replication policy which exploits the analytical model of Section 3 (*optimal* policy). Specifically, various facets of such a policy are analysed. First, it is compared with two resource-unaware policies, which replicate requests on the first encountered node only (*single* policy), and on all encountered nodes (*greedy* policy), respectively. Note that the performance in terms of expected service time of these two policies can also be found by using the analysis presented before, as they correspond to $E[R(1)]$ and $E[R(M)]$, respectively. Comparing the three policies highlights the advantage of considering resource limitations. Then, a sensitiveness analysis of the optimal policy is presented. We consider the impact of i) the probability of nodes being providers ($p^{(p)}$) or seekers ($p^{(s)}$); ii) the number of nodes in the network (N); iii) the request generation rate at seekers (λ); and iv) the average service computation time at providers ($1/\mu$). Comparison of these policies in terms of performance figures other than the service time are available in [9, 10].

In order to validate the analytical model, we also developed a simulation model based on the OMNeT++ simulator (`http://www.omnetpp.org/`). In the simulated environment the nodes move in a square, according to the Random Way-Point (RWP) mobility model, with the modifications described in [8] to guarantee stationarity (the nodes' average speed is 1.5 m/s, representative of walking speeds). 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 for a given value of m, we computed the corresponding average service time (with 95% confidence intervals). We identify the optimal case as the value of m achieving the minimum average service time.

Table 1: Default analysis parameters

$1/\mu$	30s
λ	0.005 req/s
N	100
tx_range	20m
bandwidth	5.5Mbps
input param	100KB
output param	1MB
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

Results presented hereafter show a very good agreement between simulation and analysis. The analytical model is thus able to well predict the trends of behaviour of the various policies under investigation. The analytical model provides a much more flexible tool than the simulation model. The inherent complexity of the simulation model (mainly, the number of events that are generated) makes it practically impossible to explore the policies behaviour over a large range of key parameters, while this becomes possible analytically. This is the case, for example, of the sensitiveness analysis with respect to the request generation rate λ and service computation time $1/\mu$, for which the simulation model becomes too complex (in terms of execution time and memory space), while the analytical model allows us to completely investigate the optimal policy's behavior.

Unless otherwise stated, the key parameters' values are set as in Table 1. These settings represent typical opportunistic networking environments, in which the network is sparse (average inter-contact times of about 10 minutes and contact times of about 15s). Note that the bandwidth value is a conservative estimate, as it is the typical throughput measured at the application level with 802.11b technologies operating at a nominal rate of 11Mbps [1]. For the purpose of our study, the bandwidth parameter is also able to hide all the specificities of the physical communications (interference, channel model, ...), which are therefore not explicitly taken into account.

4.1 Resource-aware vs naïve policies

In this section, we compare the performance of the optimal (resource aware) policy against that of the single and greedy policies, which do not take resource constraints into account. Table 2 compares the average service time for an increasing percentage of seekers, and a representative percentage of providers ($p^{(p)} = 0.5$, similar results are obtained for the other values of $p^{(p)}$, as well). Results are shown, for each policy, both for the analytical and the simulation model.

For a small number of seekers (i.e., $p^{(s)} \leq 0.2$), the optimal and greedy policies basically coincide, as for such a small percentage of seekers the "overall computational capacity" of the system is large enough to afford greedy replication. However, when the number of seekers increases beyond this point, the greedy policy saturates the system. In these cases, the expected service time is infinite. Simulation results are the average over completed executions, which gives an indication of the exponential increase of the simulated delay. On the contrary, beyond $p^{(s)} = 0.2$ the optimal policy progressively reduces the number of spawned replicas, and significantly outperforms both the single and the greedy policy.

Table 2: Policies comparison, $p^{(p)}$=0.5

$p^{(s)}$	optimal(an)	optimal(sim)	single(an)	single(sim)	greedy(an)	greedy(sim)
0.1	9886.32	8899.77±135.025	45914.6	44933±1313.36	9888.7	8904.68±137.857
0.2	10030.6	9567.35±108.429	45920.6	46236.7±1007.69	10072.9	9931.05±102.3
0.5	11883.7	12431.5±127.545	45940.8	44559.5±621.437	∞	69154.4±801.452
0.8	15381.7	16945.1±133.773	45965.8	45031.6±509.473	∞	167508±1501.26

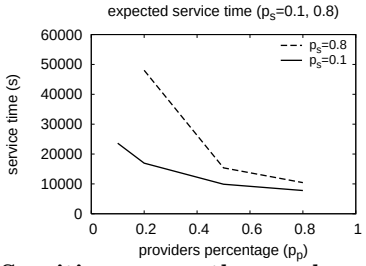

Figure 4: Sensitiveness on the number of providers

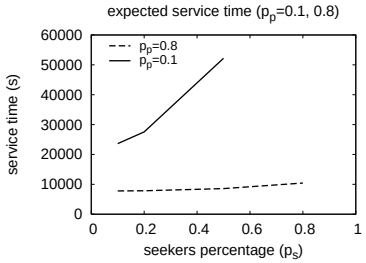

Figure 5: Sensitiveness on the number of seekers

Also note that the optimal policy results in just a slight increase of the expected service time as the number of seekers increases.

These results clearly indicate the significant performance gain that a resource-aware policy achieves with respect to resource-unaware policies. From now on, we therefore focus on assessing the performance of the optimal policy with respect to a number of parameters. Furthermore, as the analytical model shows to predict very well the trend of simulation results, in the rest of the analysis we use the analytical model only. Additional validation results are available in [9, 10].

4.2 Sensitiveness on the seekers and providers population

Figures 4 and 5 show the expected service time as a function of the percentage of providers and seekers, respectively. In each plot, two curves are shown, for the minimum and maximum values of the other parameter ($p^{(s)}$ and $p^{(p)}$, respectively).

The expected service time increases when either less providers are available, or more seekers are present. In some configurations, this may result in saturation (e.g., $p^{(s)} = 0.8, p^{(p)} = 0.1$). For a given percentage of seekers (providers) the service time decreases (increases) as the percentage of providers (seekers) increases. Unless for particularly congested settings (very low percentage of providers or very high percentage of seekers), using the optimal policy results in a graceful degradation of the performance, as the expected service time gracefully increases (e.g., see the curve for a varying number of providers and $p^{(s)} = 0.1$, or the curve for a varying number of seekers and $p^{(p)} = 0.8$).

4.3 Sensitiveness on the network size

To keep a scenario representative of opportunistic networking environments, we scale up the size of the simulation area with N to keep the node density constant (this indeed results in invariant average contact and inter-contact times). Specifically we consider the cases $N = 20, 50, 80, 100$. Figure 6 shows the expected service time for the two extreme values of $p^{(s)}$ and a representative percentage of providers ($p^{(p)} = 0.5$).

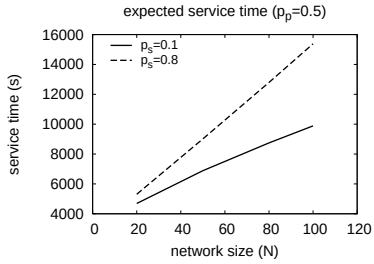

Figure 6: Impact of the network size

The expected service time linearly increases with the network size. The main reason behind this behaviour is due to the delay for downloading the output results to the seeker (i.e., the delay of the third stage). Intuitively, after a provider has completed the execution, it must encounter a single tagged seeker to download the output parameters. The probability of this event can be shown to be proportional to $1/N$, which results in a linear increase of this part of the service delay with N. More in detail, it is possible to 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., $ER_i = EW_i + NEY_i$. 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, the optimal service time $R(m_{opt})$ is also an exponential r.v. with rate $\gamma_R = \sum_{j=1}^{m_{opt}} \frac{1}{EW_i + NE\theta}$. 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 EW_i. After simple manipulations, we find 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 6(b). It also justifies the fact that the slope of the line increases with $p^{(s)}$, because, for a given value of $p^{(p)}$, m_{opt} decreases with the percentage of seekers.

4.4 Sensitiveness on the load on providers

The final aspect we analyse is the sensitiveness of the optimal policy with the load on providers. This can be shown by varying either the seekers' request rate λ (Figure 7), or the average providers' computation time $1/\mu$ (Figure 8). Specifically, we scale the value of λ logarithmically from 0.001 req/s up to 0.1 req/s, while we scale the value of $1/\mu$ by doubling it from 15s up to 240s. We show results for the lowest percentage of seekers ($p^{(s)} = 0.1$), i.e., we do not further congest the system with a high number of seekers, to better isolate the dependence on λ and μ.

Both plots essentially highlight the same feature. The service time increases with the load of the servers, be it due to an increase of the seekers' request rate, or an increase

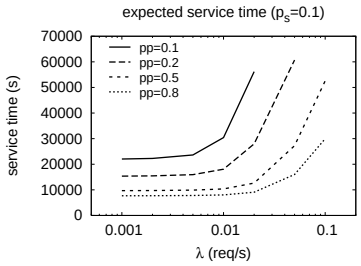

Figure 7: Sensitiveness on λ

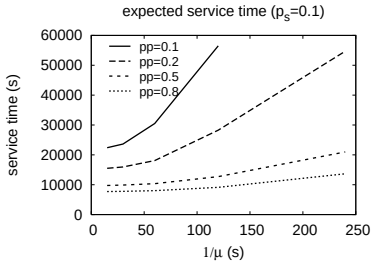

Figure 8: Sensitiveness on μ

of the computation time for generating the service results. The service time increase is moderate when the percentage of providers is high enough (e.g., for $p^{(p)} = 0.8$), meaning that the total "service capacity" of the system is high enough to tolerate increases of the offered load. For lower numbers of providers, the increase of the service time becomes more evident. For a very low percentage of providers ($p^{(p)} = 0.1$) even the optimal policy can work only up to a certain load, while the system inevitably becomes saturated beyond this limit. This is shown in the plots by the fact that the curves stop for $\lambda > 0.02$ req/s and $1/\mu > 120$s, respectively.

5. CONCLUSIONS

In this paper we have derived a complete analytical model for service invocation and provisioning in opportunistic computing environments. The model takes into consideration a very general scenario, featuring different mobility patterns, and, very importantly, resource constraints both in terms of computation and bandwidth capabilities.

Employing the proposed model, it is possible to analyse the performance of optimal service invocation, in terms of expected service time (the time required by seekers to receive output results). In this paper, we have highlighted several features. First of all, we have shown that considering resource constraints in service invocation policies is a must to optimise the performance. Naïve, resource-unaware policies either easily saturate the available resources, or significantly under-utilise them, while the optimal, resource-aware policy always optimises their use. Second, we have shown that the optimal policy is able to automatically counteract increased demand on resources, which may arise due a reduced number of providers, an increased number of seekers, an increased load in terms of request rate or service computation time. In all of these cases, the optimal policy results in a graceful degradation of the system performance, until a point where the overall service capacity of the system is too low to cope with the total offered load. Finally, we have highlighted an intrinsic increase of the service time with the network size, which is an inevitable side effect of finding a particular user in larger populations.

To the best of our knowledge, this paper is one of the first considering the challenging problem of service provisioning in opportunistic networks. Despite the significant contributions of this paper, there are several avenues for future research. Among the most interesting ones, we mention the design of distributed algorithms that implement when possible, or approximate otherwise, the optimal policy found by analysis. Characterising the impact of opportunistic multi-hop forwarding or requests upload and results download is another interesting extension. Finally, it will also be important to understand how the opportunistic computing paradigm can work when services available on different nodes can be composed together in order to provide richer functionality.

6. REFERENCES

[1] G. Anastasi, E. Borgia, M. Conti, E. Gregori, and A. Passarella. Understanding the real behavior of Mote and 802.11 ad hoc networks: an experimental approach. *Pervasive and Mobile Computing*, 1(2):237–256, 2005.

[2] C. Boldrini, M. Conti, and A. Passarella. Exploiting users' social relations to forward data in opportunistic networks: The hibop solution. *Pervasive and Mobile Computing*, 4(5):633 – 657, 2008.

[3] C. Boldrini, M. Conti, and A. Passarella. Design and performance evaluation of contentplace, a social-aware data dissemination system for opportunistic networks. *Computer Networks*, 54(4):589 – 604, 2010. Advances in Wireless and Mobile Networks.

[4] A. Chaintreau, P. Hui, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Trans. Mob. Comp.*, 6(6):606–620, 2007.

[5] M. Conti and M. Kumar. Opportunities in opportunistic computing. *Computer*, 43(1):42–50, Jan. 2010.

[6] S. Kalasapur, M. Kumar, and B. A. Shirazi. Dynamic Service Composition in Pervasive Computing. *IEEE TPDS*, 18(7):907 – 918, 2007.

[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*, 6(1):58 – 71, 2010.

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

[9] A. Passarella, M. Kumar, M. Conti, and E. Borgia. Exploiting opportunistic contacts for service provisioning in bandwidth limited opportunistic networks. In *IIT-CNR TR 18-2010, available at* `http://bruno1.iit.cnr.it/~andrea/tr/services-tr-bw.pdf`, 2010.

[10] A. Passarella, M. Kumar, M. Conti, and E. Borgia. Minimum-Delay Service Provisioning in Opportunistic Networks. In *IIT-CNR TR 19-2010, available at* `http://bruno1.iit.cnr.it/~andrea/tr/services-tr.pdf`, 2010.

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

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