Service Selection and Composition in Opportunistic Networks

Marco Conti*, Emanuel Marzini*, Davide Mascitti*, Andrea Passarella*, Laura Ricci†
* IIT-CNR, Via G. Moruzzi 1 56124, Pisa, Italy
† Department of Computer Science, University of Pisa, Italy
email: {marco.conti, emanuel.marzini, davide.mascitti, andrea.passarella}@iit.cnr.it, ricci@di.unipi.it

Abstract—**Opportunistic computing is a new computational paradigm enabling mobile users to access the heterogeneous services present in a pervasive mobile environment. With respect to conventional service-oriented approaches, in opportunistic computing services are provided by the users' mobile devices themselves, and are accessed exploiting opportunistically direct contacts between devices, i.e. without relying exclusively on fixed infrastructures such as the cloud. Pair-wise contacts are exploited to collect information on services and providers available in the network. A proper support may exploit this information to choose the most efficient composition of services satisfying a service request issued either by a user or an application.**

This paper defines a support for service selection and composition in opportunistic environments based on a mathematical model able to describe the different phases of the execution of a service composition. The model enables an estimation of the execution time of a composition and is exploited by the support for choosing the best composition among a set of available alternatives. The paper presents a set of simulations proving the effectiveness of our approach. The experiments show that our approach achieves better query resolution time and better load balancing of the service requests on the providers with respect to reference alternative approaches.

I. Introduction

The increasing popularity of mobile devices such as smart phones and tablet computers, characterized by high computing power and different interfaces for communication with other devices opens new applicative scenarios. In this perspective, opportunistic networks are self-organizing mobile networks where each user that is part of the network exploits its mobility and that of others for transferring messages, disseminating data and accessing/sharing resources [1]. Given the richness of resources and services offered by current mobile devices, the opportunistic paradigm has recently been extended to that of opportunistic computing, where the mobile network users may mutually exploit the services and resources present on their devices [2].

The basic idea of opportunistic computing is to allow the users to take advantage of the resources and services that other users share, by exploiting the direct physical contacts between the users, and the resulting possibility to exchange data through a direct connection between their devices (e.g. through wifi or Bluetooth). Opportunistic computing is complementary to conventional service oriented computing approaches. In opportunistic computing resources available on mobile devices can be directly shared among users in a very dynamic and situated way (i.e., following very closely the dynamic process of users availability and needs), without requiring to go through any pre-existing infrastructure, either at the networking level (e.g., cellular networks) or at the computing/service level (e.g., the cloud). Therefore, opportunistic computing permits to take advantage of typically unused resources available on users' devices, thus augmenting the total "service capacity" of a pervasive networking environment. A challenge to realise the opportunistic computing vision is that in this scenario, the mobility of devices makes it impossible to create and maintain a stable network topology, so that the problem of the instability of the connections has to be considered. Moreover, given the great heterogeneity of the devices, it is important to define a support able to manage the on-line selection and composition of resources and services by exploiting an analysis of the mobility and of the characteristics of the nodes, to satisfy a request submitted by a user or by an application. Thanks to such a support, opportunistic computing can provide, despite intermittent connectivity, a functioning and dynamic distributed computing environment, taking advantage of any resources available in the environment.

This paper proposes a system for the selection and composition of services in opportunistic computing environments, realised as a distributed support layer active on users' devices. This support layer is responsible for sharing resources on the devices in the form of services and takes dynamically information on resources and services from other participants in the network. With this knowledge, our system can process service requests generated either by the user or by the applications that reside on the device. It proceeds to evaluate the possible alternatives which can be exploited to resolve these requests and chooses the most convenient, taking into account both the mobility of the nodes, the heterogeneity of the participating devices, and the expected status of their resource usage (e.g., the computational load on the devices).

Figure 1 shows a possible toy scenario of our system. A user (in the following, the *seeker*, marked as node N_1) asks for a service which consists in the transformation of a raw video file into a compressed video file and a separate audio file. The seeker knows three nodes of the network (in the following, the *providers*, marked as nodes N_2, N_3, N_4) which have published, respectively, the services S_1, S_2, S_3. N_1 recognizes as viable alternatives the service S_1 which offers the required transformations and the sequential composition of the services S_2, S_3. The system we propose is able to choose

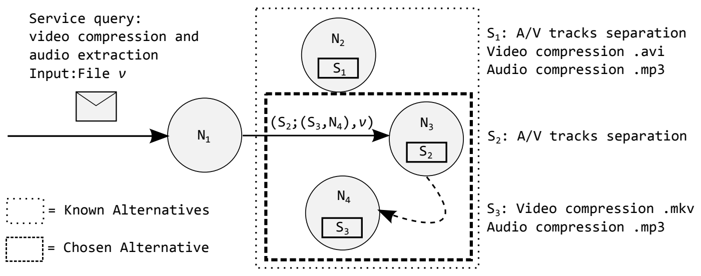


Figure 1. Service composition: an example

the best alternative through the definition of a mathematical model taking into account both the mobility of the devices and their computational capabilities in order to derive statistical measures useful for comparing the alternatives. In particular in this paper we select the alternative that minimises the expected completion time, defined as the time when the seeker receives the service results back when using a chosen alternative.

The paper presents a set of simulation results obtained exploiting the TheOne simulator [3], which has been proposed for evaluating opportunistic environments and includes different mobility models. We have modified the simulator in order to make it suitable for the management of service composition. The simulations highlight the effectiveness of the approach we propose also in presence of a large amount of requests. In particular, we compare the proposed system with different alternatives, including random selection of the alternatives and selection of the first available alternative. Our system provides better performance in terms of completion time and is also able to significantly reduce the load at providers, in particular when this is most needed, i.e. when seekers generate a high load of requests. For instance, with respect to the policy selecting the first available alternative, our system reduces the completion time up to 86%, and the average load on providers up to 40%.

The paper is organized as follows. Section II describes the main approaches to service provisioning and composition in mobile environments. The overall architecture of the system is defined in Section III and the policy for selection of the composition is introduced in Sections IV and V. Section VI discusses the experimental results. Finally, concluding remarks are reported in Section VII.

II. Related work

The goal of opportunistic computing is to allow the users to opportunistically compose the resources in the network. The sharing of resources in a heterogeneous mobile network can be used to compose functionalities not available in a single node of the network thus providing a much more rich functionality set. Such a vision requires new solutions for orchestration and management of resources on different devices [4]. Service composition exploits a mechanism based on graph theory that allows to collect the knowledge of the services offered and to represent this knowledge to evaluate the best alternative to use. Some recent research proposal take into accounts these aspects. [5] describes a system allowing service discovery and composition in networks with stable connectivity. The proposed system includes a mechanism for modelling services representing their semantics through the use of ontologies. They define a taxonomy of services through a list of concepts describing them [6]. A key problem of service provisioning through direct communication between nodes in mobile environments concerns the possibility that possible service providers may not be directly reachable (even through a multi-hop ad hoc path) when the seeker issues a request.

In opportunistic computing, instead, the fact that nodes may not be reachable is considered as the rule, and proposed mechanisms are designed correspondingly. For example, in [7] fault tolerance is achieved through a middleware that exploits information about the devices to create parallel service compositions, in order to increase the probability for successful executions. [8] proposes some heuristics for service composition taking into account the last time of encounter between nodes and the reported load at providers.

In [9], [10] the problem is addressed using a stochastic analysis of the providers in order to find the ones that are most likely reachable from the seeker during the entire period necessary to perform the composition. In these works, the loss of connection is considered as a failure state.

[11] analyses the issue of single service provision in opportunistic networks. The main aim is to improve the efficiency of service provision by replicating requests to a set of different providers. The proper number of requests is computed by considering information on the mobility of the users and on the load of the nodes. The optimal number of requests is computed through an analytical model that minimizes the expected value of the request resolution time, defined as the time required to receive the first response from suppliers.

In this paper we propose a system extending previous techniques whose aim is to minimize the time for the provisioning of a composite service. With respect to [7] and [8] this is based on an analytical model for computing the expected completion time of each alternative, rather than on simple heuristics for selecting the composition (similar in design to the ones we compare against in simulations). With respect to [11], we consider here the possibility of composing different services.

III. System Architecture

The system we propose enables either a user or an application of a mobile network to compose local and remote services to satisfy a user request. Even if a service request may be satisfied either by an atomic service or by a composition of services, in the following we will consider the more general scenario of service composition.

The system is based on a mathematical model which exploits the local knowledge of the network collected by a node through its opportunistic contacts with other nodes. The algorithm to choose, among the alternative compositions, the one providing the best solution according to a given metrics is based on this model.

An important characteristic of our approach is that when evaluating alternative compositions, the system considers also nodes of the network which are not connected to the seeker when the service request is generated. This implies that a node, in its opportunistic contacts, has to collect information related to the mobility of other nodes and to their computational load. The trade-off to be considered for choosing the strategy to spread this information is between the overhead for its diffusion and the richness of the collected statistics. It is worth noticing that our approach is independent of the particular adopted strategy.

Further, the evaluation of the alternatives is dynamic. As a matter of fact, the seeker may establish new opportunistic contacts and collect new information from the network before the node offering the first service in the composition is contacted. Since this information may alter the classification computed previously, the compositions must be re-evaluated whenever fresh information is received from the opportunistic contacts before contacting the first provider of the best path. To describe the system behaviour in more details, we show how a service request is managed and which components are involved in the resolution of the request.

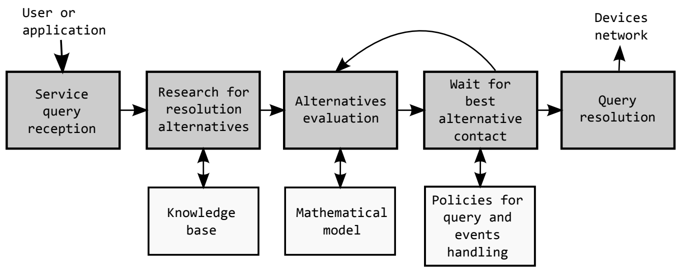


Figure 2. Service request resolution

Consider Figure 2. When a node generates a query, it searches for alternative service compositions satisfying it. To this end, it builds a graph representing all these compositions. The formal definition of the graph is given in Section V. The graph is built by exploiting the information present in the knowledge base recording the services available in the network (to the best of the node's knowledge) and the providers offering them. The graph is weighted by a set of values required to evaluate each alternative. While the structure of the graph is not modified after its construction, the weights paired with the nodes/arcs of the graph are dynamically updated when fresh information is collected by new contacts.

The modelling of the different phases to execute a service composition, i.e. waiting the next provider of the composition, transfer of data for running the service component, queue waiting and waiting time at the provider before the execution an start, is described in Section IV. The evaluation of the alternatives through the mathematical model can be described as the application of a set of parameters to a function where the value of the parameters is taken from the knowledge base of the node. The result of the evaluation is a set of values that enable ordering the alternatives by the defined metrics.

When the best alternative is detected, if the chosen provider cannot be directly contacted, the node waits for a contact with it. While waiting for this, it may encounter further nodes and collect new information from them. The task of the events handling component is to keep monitoring both the network and the node itself to catch events that may affect either the resolution of a pending query or, in general, the local knowledge base. The kind of events to handle depend on the policies to handle queries and by the information required to weight the evaluation graph. The event handling component triggers either the transmission of fresh information on the network as a response to an event generated by the local node or the refreshing of the local knowledge base as a response to the reception of new information received from the network.

IV. Modelling service execution

This section introduces the stochastic model exploited to estimate the execution time of a single service (component). Modelling the execution time of a single service is the building block to evaluate the completion time of a composition. Starting from this, we will model service composition in Section V.

We can divide the execution time of a single service (R) into the following phases:

- *Contact of the service provider* (W). The time to contact the node providing the service is determined by the intercontact time between the seeker and the provider. This value depends on the their relative mobility.
- *Data transfer* (Input Time B, Output Time θ). Input/output data for the service execution must be transferred from the seeker to the provider and back. Note that in an opportunistic network data transfer between two nodes may be affected by connection disruptions due to the nodes mobility. This implies that the duration of contacts affects the number of contact events required to complete the transfer, while both contact and intercontact duration affect the time to transfer data.
- *Queue waiting time* (DQ). Once transferred, requests may be delayed at the provider due to previous pending executions (we model this as a FIFO queue at the provider). The duration of this delay depends both on the frequency of the request arrivals to the provider and on the time to process them.
- *Service execution time* (DS). The time to execute a service on the provider depends both on its computational capabilities and on the type of the service.

Since the execution of previous phases is sequential, we obtain:

$$R = W + B + DQ + DS + \theta$$

We will define R as a random variable whose expected value will be exploited to choose the best alternative. The random variables corresponding to the different phases will be introduced in the following sections.

A. Contacting the service provider

We introduce the random variables $T_{C_{n_h,n_j}}$ and $T_{IC_{n_h,n_j}}$ modelling, respectively, the contact and intercontact times between two nodes n_h and n_j. For each pair of nodes, we assume that contact and intercontact times between those nodes are independent and identically distributed (i.i.d.). We also assume that contact and intercontact times of different pairs of nods are independent of each other. Finally, we assume that the variables $T_{C_{n_h,n_j}}$ and $T_{IC_{n_h,n_j}}$ follow exponential probability distributions with rates δ_{n_h,n_j} and δ'_{n_h,n_j}. As shown by real trace analysis presented, for example, in [12], [13], although controversial, exponential contact and intercontact times is one of the possibilities, and is a common assumption in the literature on opportunistic networking and computing (e.g. [14], [11]). Since a node cannot know beforehand the values of δ_{n_h,n_j} and δ'_{n_h,n_j}, each node computes an estimate of these values by averaging the values of contact and intercontact time with other nodes collected by opportunistic contacts.

The time for node n_h to contact the generic service provider n_j, is denoted by the random variable W_{n_h,n_j}. This is equal to 0 if, at the time when the evaluation is done, n_h and n_j are in contact, while it is equal to the residual intercontact time $T_{IC_{n_h,n_j}}$ otherwise (and, under our assumption, its expected value is equal to $E[T_{IC_{n_h,n_j}}]$ due to the memoryless property of the distribution).

The expected value of W_{n_h,n_j} is recomputed at each connection/disconnection of the two nodes, changing from 0 to $E[T_{IC_{n_h,n_j}}]$ when a disconnection occurs or from $E[T_{IC_{n_h,n_j}}]$ to 0 in case of a connection establishment.

B. Data transfer time

The estimation of the time to transfer data between two nodes requires to take into account the network dinamicity, since disconnections may occur during the transfer process. If a transfer between two nodes starts at the beginning of a contact period, it may be interrupted at the end of each contact, to be resumed when a new contact (between the same nodes) is established. This means that the total time for the transfer has to be computed by considering a sequence of time intervals.

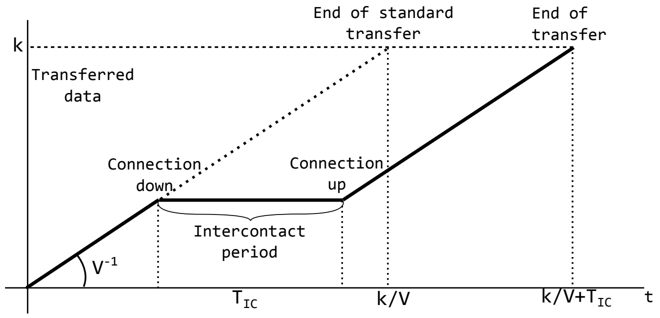


Figure 3. Modelling Data Transfer

We assume that the data throughput is a constant $V > 0$ computed by averaging its values measured during the contact periods.

We denote the random variable modelling the time needed to transfer the input data from the seeker n_h to the provider n_j as B_{n_h,n_j}. The input data transfer is characterized by starting while a contact is established with the provider and, hereafter, we refer as B_{n_h,n_j} for each data transfer time starting during a contact period. B_{n_h,n_j} depends on k, the size of data to transfer, on V and on the number N of contacts required to transfer data. Figure 3 illustrates through an example the general data transfer process. If no disconnections would occur during the whole process, the data transfer time would be k/V. Otherwise, additional intercontact times must be added, depending on the encounter pattern between the nodes. Specifically, the number of contact events necessary to complete the data transfer is equal to n with a probability that the sum of n contact times between the two nodes is less than k/V, and the sum of $n+1$ contact times is greater than k/V.

Denoting with $T_{C_{n_h,n_j}}(i)$ the length of the i-th contact event between n_i and n_j, the probability that N is equal to n is thus:

$$P\{N = n\} = P\{\sum_{i=1}^{n} T_{C_{n_h,n_j}}(i) < \frac{k}{V} \leq \sum_{i=1}^{n+1} T_{C_{n_h,n_j}}(i)\}$$

If we condition to the number of contact events required to complete the data transfer, B_{n_h,n_j} can be computed as k/V (the sum of the contact times during which the data transfer occur) plus the sum of the n intercontact times occurring between the required contacts.

$$B_{n_h,n_j|N=n} = \frac{k}{V} + \sum_{i=0}^{n} T_{IC_{n_h,n_j}}(i)$$

Removing the conditioning on *n*, by exploiting the previous formulation of $P\{N = n\}$, we obtain the formula of the expected value of B_{n_h,n_j}(the complete derivation is available in [15] and is omitted here for space reasons):

$$E[B_{n_h,n_j}] = \frac{k}{V} * \left(1 + \frac{\delta_{n_h,n_j}}{\delta'_{n_h,n_j}}\right)$$

Finally, the random variable θ_{n_h,n_j} modelling the output data transfer time, can be expressed by taking into account that the seeker is not capable of forecasting whether the service execution will end during a contact period. We can exploit the definition of B_{n_h,n_j} to define θ_{n_h,n_j} as:

$$\theta_{n_h,n_j} = \begin{cases} B_{n_h,n_j} & \text{if } n_h \text{ and } n_j \text{ are in contact} \\ B_{n_h,n_j} + T_{IC_{n_h,n_j}} & \text{otherwise.} \end{cases}$$

and its expected value is [15]:

$$E[\theta_{n_h,n_j}] = \frac{k}{V} * \left(1 + \frac{\delta_{n_h,n_j}}{\delta'_{n_h,n_j}}\right) + \frac{\delta_{n_h,n_j}}{\delta'_{n_h,n_j}} * \frac{1}{\delta'_{n_h,n_j} + \delta_{n_h,n_j}}$$

C. Queue waiting time

A provider offering a set of services receives a stream of requests from the network and enqueues them, waiting for the execution.

We consider the $M^{[X]}/G/1$ [16] queueing model, where nodes generate queries according to a Poisson distribution with rate λ and send batches of requests to the provider (upon encountering). Consider the random variable DQ_{n_j} modelling the waiting time for a service request in the queue of provider j in a $M^{[X]}/G/1$ queue system and the random variable G modelling the number of queries in a batch received by the provider. The expected value of the random variable DQ_{n_j} can be computed [16] if we assume that the first two moments of the general distribution of the service execution time DS_{s_i,n_j} (with expected value d and $d^{(2)}$ its second moment) and of the random variable G (with expected value g and $g^{(2)}$ its second moment) do exist. These values can be estimated by monitoring the batches arriving to the provider and the executed services.

To complete the characterization of DQ_{n_j} we extract the average rate λ of the query batches arrivals to the provider and compute the average load ρ of the provider as $\lambda * g * d$. Starting from these values, the expected value of DQ_{n_j} can be computed, as shown in [16], as:

$$E[DQ_{n_j}] = \frac{\lambda g d^{(2)}}{2(1-\rho)} + \frac{g^{(2)} d}{2g(1-\rho)}$$

D. Service execution time

The random variable DS_{s_i,n_j} for the execution of the service s_i on a provider n_j is influenced both by the device computational power and by the implementation of the requested service. Each provider estimates the expected value of the DS_{s_i,n_j} by collecting the execution times of that service and transmits this value in each opportunistic contact.

V. Service Composition

A service request may be satisfied by a composition of the services offered by the nodes of the network. In our solution, the composition is defined by the seeker which exploits its local knowledge of the services present in the network and of the providers offering them.

At an abstract level, we define a directed *Service Graph* showing the execution dependencies inside a composition. Each path connecting two vertices of the graph shows a valid sequence of service executions. Each service s_j is identified in our system as a pair (I_j, O_j) where I_j is the input type and O_j is the output type of s_j and these types are atomic. For the sake of simplicity, in the following we assume that these types are codified by integer values, furthermore we will consider acyclic compositions since they represent the most frequent situation in real scenarios. For instance, Figure 4 shows a set of services $\{S_1, S_2, S_3, S_4\}$ linked by their type dependencies together with two special services *start* and *end* representing the start and the end points of the composition to which are assigned the input, respectively the output type of the service request. Each other node is paired with a pair of integers that represents the input, respectively the output type of the corresponding service.

To evaluate alternative compositions, each service in this graph has to be paired with the nodes (providers) offering it. The *Composition Graph*, shown in Figure 5 is defined by

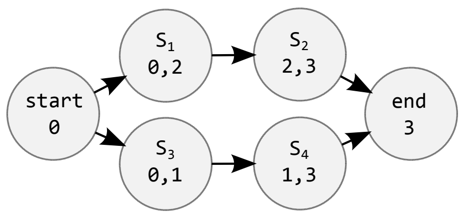


Figure 4. The Service Graph

replacing each vertex of the service graph corresponding to a service S_i by a set of vertexes (S_i, N_j) such that node N_j provides service S_i, and the edges of the modified graph link the same services pairs of the abstract graph. Note that node N_0 corresponds to the seeker, while services *start* and *end* represent, respectively, the service request issued by the seeker N_0 and the reception of the request output by N_0.

The *Composition Graph* can be used to determine, based on the seeker's local knowledge, all the compositions which satisfy the service request, just by identifying the set of paths from the *start* to the *end* service. Each path corresponding to a sequential composition of services is shown in Figure 5.

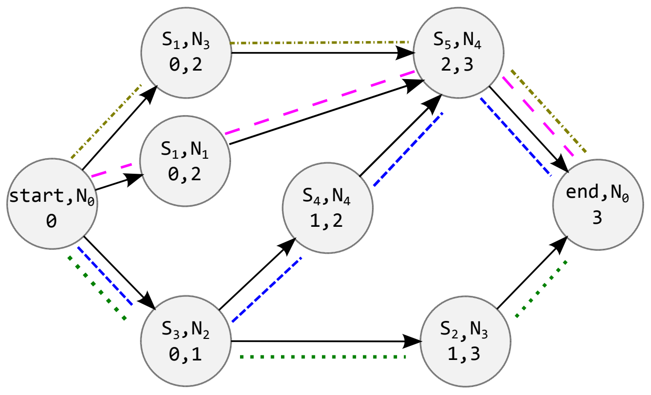


Figure 5. The Composition Graph

To estimate the execution time of different compositions and choose the best one, the graph is weighted by considering the local knowledge of the node. Each vertex of the graph is paired with the average queue waiting time at the corresponding provider and with the average service execution time of the corresponding service at that provider, while the edges of the graph are labelled by considering the average time required for contacting the next node and for transferring data to it.

The time to transfer intermediate results between providers may be computed in different ways according to the knowledge of the network collected by the seeker. In the solution requiring the minimal overhead in terms of amount of exchanged information between nodes, each provider transfers the results back to the seeker which forwards them to the next provider of the composition. In the solution that maximises the amount of knowledge that the seeker can exploit to estimate the best composition, the intermediate results of the composition are directly transferred between the providers. In this case the seeker needs to know the contact and intercontact rates of any pair of providers involved in the composition. These values may be epidemically exchanged when nodes come into contact. Let n_i and n_j be a pair of nodes establishing a contact and exchanging their knowledge of the network. n_i needs to get each contact rates δ_{n_j,n_h} and intercontact rates δ'_{n_j,n_h} of each other node n_h that is known by n_j (so that its services are also known). This solution may require to exchange up to n^2 elements, where n is the number of nodes in the network. The trade-off is between the data exchanged and the possibility to exploit direct data transfer between nodes, which may result in more efficient solutions, because data has not to be transferred back to the seeker after each invocation. As we discuss next, the estimation of the time required for a specific service composition also depends on which solution is used.

A. Modelling Service Compositions

Let us consider a vertex (s_j, n_i) of the Composition Graph: it represents the execution of the service s_j on the provider n_i, while an edge $((s_j, n_i), (s_t, n_k))$ shows that n_k waits from n_i the results produced by service s_j to start execution of service s_t. In the following, without loss of generality, we will assume that the node corresponding to the seeker is n_0. Let us consider a path p from the vertex $(start, n_0)$ to the vertex (s_j, n_i) of the composition graph: our goal is to define a random variable R_{p,s_j,n_i} modelling the time to execute the sequence of services on the path p, starting from the seeker n_0, up to the end of the execution of service s_j inside the provider n_i. The expected value of R_{p,end,n_0} is an estimation of the time to execute the composition corresponding to a specific alternative, available through path p.

The form taken by the random variable R_{p,s_j,n_i} depends on the solution adopted to transfer data between the providers of the composition. Let us introduce the variable $\overline{\theta}_{s_j,n_i,s_t,n_h}$ modelling the time required to transfer data from the service s_j of the provider n_i to the service s_t of the provider n_h. In the first scenario, the intermediate results generated are transmitted to the seeker which, in turn, propagates them to the next provider of the composition. In this case, we define the variable $\overline{\theta}_{s_j,n_i,s_t,n_h}$, by distinguishing the first step of the composition from the other ones. As discussed in section IV, when considering the first data transfer, the seeker is able to know whether the contact with the first provider is available, while this is not possible in the following transfers. Therefore, if an edge of the graph connects the start service with another service, we sum up the initial contact waiting time and the data transfer starting during a contact period. In the other cases, we sum up the data transfer time between each provider and the seeker.

The definition of $\overline{\theta}_{s_j,n_i,s_t,n_h}$ is therefore the following one:

$$\overline{\theta}_{s_j,n_i,s_t,n_h} = \begin{cases} W_{n_i,n_h} + B_{n_i,n_h} & \text{if } s_j = start. \\ \theta_{n_i,n_0} + \theta_{n_0,n_h} & \text{otherwise.} \end{cases}$$

where W_{n_t,n_k},B_{n_t,n_k},θ_{n_t,n_k} have value 0 if $t = k$.

In the second scenario the seeker estimates if the lower execution time is obtained by transmitting the intermediate results directly between the providers or by returning them to itself. In this case, the random variable $\overline{\theta}_{s_j,n_i,s_t,n_h}$ is computed as in the previous case, by considering that each transfer between two providers may be realized by relying on the seeker or by a direct transfer.

$$\overline{\theta}_{s_j,n_i,s_t,n_h} = \begin{cases} W_{n_i,n_h} + B_{n_i,n_h} & \text{if } s_j = start \\ min(\theta_{n_i,n_h}, \theta_{n_i,n_0} + \theta_{n_0,n_h}) & \text{if } s_j \neq start \end{cases}$$

Let us now compute the random variable R_{p,end,n_0} expressing the execution time of the composition corresponding to the path p.

Consider a path p in the Composition Graph defined by m providers $n_1, ..., n_m$ and m service invocations $s_{n_1}, ..., s_{n_m}$, where s_{n_i} is the service invoked on the node n_i of p. and consider a query submitted by the seeker n_0. The random variable defining the execution time of the path p, is defined by adding a component for the first composition step, one for the sequence of service executions up to end of the composition and a component for the final data transfer to the seeker. Recalling the notation for the waiting time in the providers' queues and the execution times at the providers (defined in Section IV), and the formulas presented in this Section, it is easy to derive the following expression:

$$R_{p,n_0,end} = T_{first} + T_p + T_{last}$$

where:

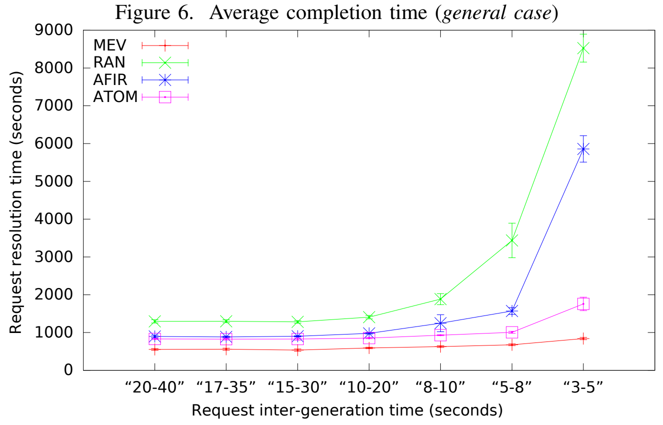
$$T_{first} = W_{n_0,n_1} + B_{n_0,n_1} + DQ_{n_1} + DS_{s_{n_1},n_1}$$

$$T_p = \sum_{i=2}^{m} (\overline{\theta}_{s_{n_{i-1}},n_{i-1},s_{n_i},n_i} + DQ_{n_i} + DS_{s_{n_i},n_i})$$

$$T_{last} = \overline{\theta}_{s_{n_m},n_m,n_0,end}.$$

The expected value of the execution time may be computed by assigning a weight to each edge, according to the previous formula, and then using the shortest path algorithm [17] to find the best alternative.

Figure 6. Average completion time (*general case*)



VI. System Evaluation

In order to validate the effectiveness of a system where the choice of the service composition exploits the model presented above, we developed a set of simulations through TheOne [3]. The experiments exploit a set of mobility traces

Figure 7. Average completion time (*case08*)

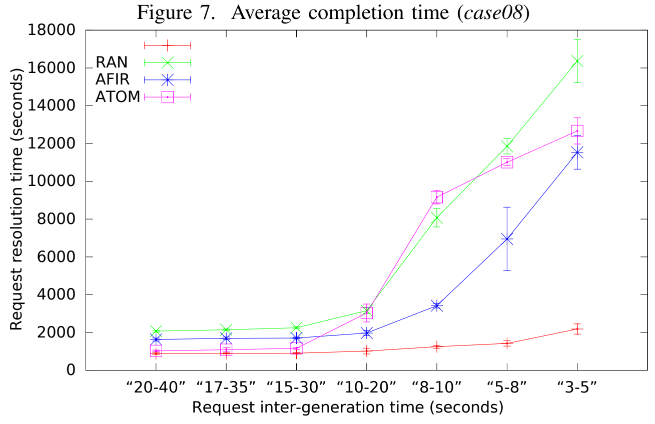
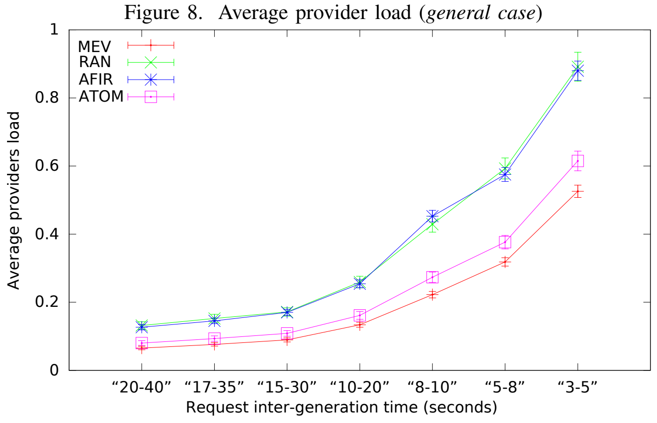


Figure 8. Average provider load (*general case*)



generated according to the RandomWayPoint model, modified as discussed in [18] in order to avoid problems related to the initial transient phase of the mobility model.[1]

The main parameters of the simulations are described in Table I.

Table I
SIMULATION PARAMETERS

Simulation runs	5
Number of nodes	$500m \times 500m$
Total simulation time	$70000s$
Warm-up time	$10000s$
Request generation phase	$30000s$
Connectivity range	$90m$
Transmission speed	$2Mbps$
Input/output data size	$20KB/2MB$
Density of each service	25%
Input type range	$i \in [0, 7]$
Requests output type range	$o \in [1, 8], o > i$

As previously described, each service is identified by the type of its input/output which is codified by an integer. In our simulation an input type i is selected in the integer range [0,7], while output type o in the range [1,8], with the constraint that i is less than o to avoid cyclic compositions. Each service is randomly assigned to 25% of nodes.

[1]Note that, although in general other mobility models are considered more realistic, RWP is still a valid option when users form a unique social community moving in a common area [19]

We consider two different scenarios for service requests. In the first one, the service requests are generated so that both the input and the output type of each service is randomly selected. In the second scenario, referred as *(case08)*, all requests have input type 0 and output type 8. This represents the case with the longest possible composition in our scenario.

We compare the following service selection policies:

- *Minimum Expected Value (MEV).* The choice of the service composition is selected according to our model.
- *Random(RAN).* For each request of service, a random path in the Composition Graph is selected.
- *Always First (AFIR).* The path on the Service-Node Graph is selected by the seeker by considering, at each composition step, if possible, a suitable provider that is already in contact with it. If no such provider exists, the seeker waits to encounter such a provider.
- *Atomic (ATOM).* The seeker waits for a provider that offers a single service satisfying the request. In this case service composition is not taken into account at all.

Our experiments measure both the average completion time of the service requests and the average load on the providers. Our results are the average of 5 independent simulation runs, shown with 95% confidence intervals.

TheOne allows us to configure the frequency of requests creation and their assignment to the seekers. When a new request is created, it is assigned to a seeker selected at random. We examine the behaviour of the system in different query load scenarios, by increasing the number of requests generated by the system starting with a generation time between requests uniformly distributed in the range 20-40 seconds, up to a generation time between requests uniformly distributed in the range 3-5 seconds.

As we will see in the following section, the effectiveness of the policy *MEV* with respect to the *AFIR*, *RAN* and *ATOM* policy is more remarkable for higher load values.

A. Results

In this section we consider the more general scenario where the seeker exploits also the information received by the encountered nodes, like the intercontact times of these nodes with other ones.

The results presented in Figure 6 and in Figure 7 show how the performance of *MEV* remains fairly stable when varying the number of requests, as opposed to the *AFIR* and *RAN* policies, which have the worst performance and present a massive degradation in the case of a high frequency of service requests. *ATOM* follows the trend of *MEV* with average times slightly larger than *MEV* in the general case, despite it requires a single service invocation to satisfy the user request. In case *case08* its behaviour is similar to that of other policies different from *MEV*.

For what concerns the average load (Figure 8), *RAN* and *AFIR* bring most of the providers to saturation (as the *average* load approaches 1). *ATOM* and *MEV* result in a significantly lower average load. Specifically neither of them exceeds 0.6, and *MEV* achieves a lower average load. This is remarkable, as - in general - the total number of generated requests is higher in *MEV*, because *MEV* exploits service composition (while *ATOM* does not) and thus generates a number of requests larger or equal to 1 for *each* request issued by the applications. Nevertheless, this result in an average load even lower than in *ATOM*, showing that *MEV* is able to efficiently distribute the additional load. Simulations in the *case08* provide similar results and are omitted for space reasons (they are available in [15]).

Overall, by comparing the four policies based on the two performance figures, we can conclude that *MEV* largely outperforms *AFIR* and *RAN*. *MEV* outperforms also *ATOM*, though to a lesser extent in the average case. However, as soon as some specific service becomes very popular (like in the *case08*), *MEV* largely outperforms also *ATOM*, because it exploits composition to avoid constantly using only the set of providers that provide exactly that service (as *ATOM* does), thus overloading them.

VII. Conclusions

In this paper we have presented a system able to select and compose service requests in an opportunistic environment. We have defined a mathematical model to evaluate alternatives known by the seeker where the request is generated. The simulation results show the effectiveness of our approach. We plan to extend our proposal in several directions. For instance, we plan to consider parallel composition of services and multi-hop service invocation, where also nodes not directly contacted are considered.

References

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

[2] M. Conti, M. Kumar, *Opportunities in opportunistic computing*. Computer, IEEE, vol. 43, no. 1, 2010.

[3] A. Keränen, J. Ott, T. Kärkkäinen, *The ONE Simulator for DTN Protocol Evaluation*. SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, 2009.

[4] M.Conti, S.Giordano, M.May, A.Passarella, *From opportunistic networks to opportunistic computing*. Communications Magazine, IEEE, vol. 48, no. 9, 2010.

[5] S. Kalasapur, M. Kumar, B. A. Shirazi, *Dynamic Service Composition in Pervasive Computing*. IEEE TPDS, Vol.18, Issue 7, 2007.

[6] D. Bianchini, V. D. Antonellis, M. Melchiori, *An Ontology-Based Architecture for Service Discovery and Advice System*. Proceedings 16th International Workshop on Database and Expert Systems Applications, 2005.

[7] S. A. Tamhane, M. Kumar, A. Passarella, M. Conti, *Service Composition in Opportunistic Networks*, The IEEE International Conference on Cyber, Physical and Social Computing, 2012.

[8] U. Sadiq, M. Kumar, A. Passarella, M. Conti, *Modeling and Simulation of Service Composition in Opportunistic Networks*. ACM MSWiM, 2011.

[9] L. Del Prete, L. Capra, *Reliable discovery and selection of composite services in mobile environments*. In Proceedings of IEEE EDOC, 2008.

[10] J. Wang, *Exploiting mobility prediction for dependable service composition in wireless mobile ad hoc networks*. IEEE Transactions on Services Computing, 2011.

[11] A. Passarella, M. Kumar, M. Conti, E. Borgia. *Minimum-Delay Service Provisioning in Opportunistic Networks* IEEE Transactions on Parallel and Distributed Systems, Vol. 22, Issue 8, 2011.

[12] W. Gao, Q. Li, B. Zhao, G. Cao, *Multicasting in delay tolerant networks: a social network perspective*. ACM MobiHoc, 2009.

[13] P.U. Tournoux, J. Leguay, F. Benbadis, J. Whitbeck, V. Conan, M. de Amorim, *Density-aware routing in highly dynamic dtns: The rollernet case*. IEEE Transactions on Mobile Computing, vol. 10, no. 12, 2011.

[14] T. Spyropoulos, T. Turletti, K. Obraczka, *Routing in delay tolerant networks comprising heterogeneous node populations*. IEEE Transactions on Mobile Computing, vol. 8, no. 8, 2009.

[15] M. Conti, E. Marzini, D. Mascitti, A. Passarella, L. Ricci, *Services selection and composition in Opportunistic Networks*. Technical report available on: http://compass2.di.unipi.it/TR/Files/TR-12-10.pdf.gz, 2012.

[16] H. Takagi, *Vacation and Priority Systems, Part 1*. Elsevier Science Publishers B.V., 1991

[17] S. Dasgupta, C. Papadimitriou, U. Vazirani, *Algorithms*. McGraw-Hill, 2007

[18] W. Navidi, T. Camp, *Stationary Distributions for the Random Waypoint Mobility Model*. IEEE Transactions on Mobile Computing, vol. 3, no. 1, 2004

[19] C. Boldrini, A. Passarella, *HCMM: modelling spatial and temporal properties of human mobility driven by users social relationships*, Elsevier Computer Communications, Vol. 33, Issue 9, 2010.