UNIVERSITÀ DI PISA
Scuola di Dottorato in Ingegneria “Leonardo da Vinci”

Corso di Dottorato di Ricerca in
INGEGNERIA DELL’INFORMAZIONE

Tesi di Dottorato di Ricerca

Opportunistic Service Provisioning in Mobile Clouds of Users’ Personal Devices

Davide Mascitti

Anno 2016

UNIVERSITÀ DI PISA

Scuola di Dottorato in Ingegneria "Leonardo da Vinci"

Corso di Dottorato di Ricerca in
INGEGNERIA DELL'INFORMAZIONE

Tesi di Dottorato di Ricerca

Opportunistic Service Provisioning in Mobile Clouds of Users' Personal Devices

Autore:

Davide Mascitti _____

Relatori:

Prof. Enzo Mingozzi _____

Dott. Marco Conti _____

Ing. Andrea Passarella _____

Anno 2016
SSD ING-INF/05

Abstract

Opportunistic computing is the recent application of delay-tolerant networking to the creation of networks of mobile devices that give users the capability to share and access services provided by other (mobile) devices in proximity without using any cellular infrastructures. The importance of this paradigm becomes apparent given the ubiquitous proliferation of personal mobile devices in recent years. Opportunistic computing can also be used to realise service offloading, a recent trend in mobile networking research where resources on the edge of the cellular network are used in synergy with the cloud infrastructure. The importance of this application of opportunistic computing comes from the data traffic generated by mobile devices that, in the last few years, has been steadily increasing. While the development of LTE and LTE-A will boost cellular network capacity, it is unclear whether this would be enough to support the expected exponential increase in traffic demands in the medium term. Opportunistic techniques can contribute to solve this problem by offloading computation and data access to locally available devices, exploiting unused resources and balancing allocation of users requests to obtain an increase in service provisioning performances and avoiding network congestion.

This thesis brings contributions in two different scenarios: the first one is purely opportunistic with the detailing of a distributed system for the establishment and self-organization of mobile service provisioning. The system is established by each device autonomously collecting and using context information to individuate sequential compositions of resources for service provisioning and, thanks to a stochastic model, find the alternative that is expected to result in the lowest service provisioning time. In the second scenario, this thesis presents a solution for the integration of the opportunistic paradigm into a mobile edge system, where

service provisioning is orchestrated between mobile devices and a remote cloud system thanks to the collaboration with network base stations local to the mobile devices. In the first scenario, experiments are presented to validate the decision algorithms and the stochastic model they rely on, while in the second scenario, we evaluate the performance gains obtained by using the opportunistic paradigm for service offloading in respect to traditional remote cloud systems.

Contents

List of Figures

List of Tables

1

Introduction

1.1 Scenario

We have seen in recent years a more and more pervasive diffusion of personal mobile devices like smartphones and wearable sensors. Together with the widespread use of these devices, there has been a striking evolutionary trend about their hardware and software characteristics: what were in the past very limited devices offer now computing capabilities similar to laptops and significant enough storage to host a great variety of services. This, coupled with wireless network interfaces, enables the owner of these devices produce and process significant amounts of data.

All these elements enable a pervasive computing environment where users' devices can form self-organising networks and cooperate to offer each other computing resources, abstracted as services. The services that can be offered by this kind of environment range from raw computing power and storage space to sensor access or software functionalities. Wireless interfaces can be used to establish direct connections between the devices in order to access resources or exchange data of interest.

Implementing a self-organizing network of mobile devices for service provisioning presents various challenges, especially when the devices are enclosed in an area where a common pre-existing network infrastructure is not present or overloaded. In accessing and retrieving the output of services, mobility impacts the ability to transfer data and identify the network topology of the devices to find available services and use them. The devices topology in this case cannot be also assumed to be completely connected, with the presence of groups of devices that may not have a connection path to the other ones in the environment.

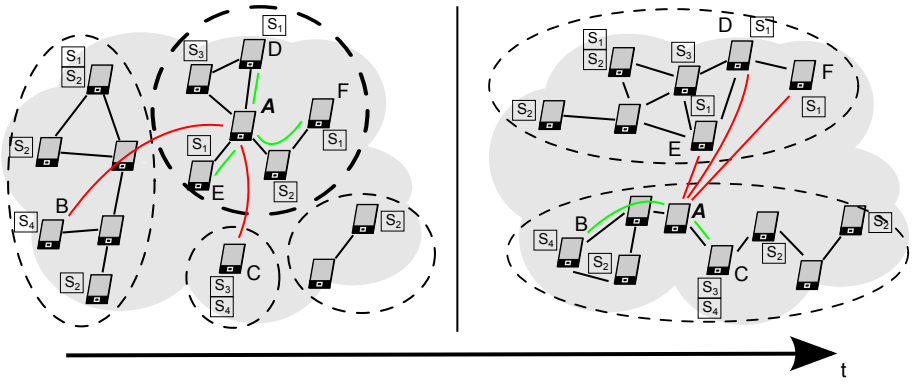


Figure 1.1: Service provisioning scenario without infrastructure

An example for this scenario can be seen in Figure 1.1, where several devices are depicted in an area and some of them share services of different types (indicated as s_1, s_2, s_3, s_4) to other devices. In this case device A may initially (left side of the picture) access a service of type s_1 directly from devices D and E or using a multi-device path to reach device F, but if A wanted to access a service of type s_4 it would not be possible given that only devices B and C offer that service and they do not have a path connecting them to A. Later, devices mobility may bring A in contact B and C, while D, E and F may become unreachable, and it may happen while a service s_1 has been requested and is being executed. In this scenario we would ideally like to be able to have knowledge and reach devices that may not be reachable and also be able to satisfy a service request that is already being served, even in case of disconnection.

To this end, the Opportunistic Networking paradigm can be used. Opportunistic Networking is a recently developed technique that aims in solving the issue of data communications in mobile networks where there are no stable paths between any couple of participants. The main characteristic of Opportunistic Networks is that users' mobility can be exploited to carry messages around the devices network, waiting for new contacts with other devices that may take the messages nearer to their destinations. Extending the Opportunistic Networking paradigm from message transfer to service provisioning, we obtain Opportunistic Computing. This is a paradigm where resources on mobile devices are abstracted into services that can be shared and remotely accessed by other participating devices. A system built using Opportunistic Computing would form a collaborative mobile devices network,

sharing a service pool comprised by all the resources shared by the users that take part in it.

In this work we explore the possibility of building Opportunistic Computing systems where the participant devices may automatically share and access each other's services in a way that would be effective performance-wise for the users and at the same time sustainable for the participants that are willing to share resources. To achieve these objectives, we investigate distributed mechanisms whereby devices are able to individually collect information about the available services, the devices that provide them an the state of the network. This information collection has also to be elaborated to form knowledge, called context information, that can be used to provide estimates of the performance obtainable using different devices as service providers and find the most suitable alternative for the users' needs.

Similarly to traditional cloud architecture, with an Opportunistic Computing system of this kind it would be possible to obtain the creation of "local mobile clouds", that means environments where the devices surrounding the users would provide collaboratively a powerful and rich set of resources that could be used automatically at need, without having to rely on a network infrastructure.

While opportunistic computing addresses a scenario where nodes completely self-organise in absence of any infrastructure, the concept of opportunistic computing can also be used to integrate conventional cloud platforms and local service provisioning from local mobile clouds. This is the concept of Mobile Edge and Fog Computing, which is explored in the second part of thesis.

Mobile Edge Computing is an emerging technology for service provisioning where mobile devices, located in an area where cellular network connectivity is available, can use service providers that are geographically near the users and the network endpoint providing connectivity, specifically on edge gateway at the border between the access network and the Internet, or even on other mobile devices in proximity. An effect of this solution is that, by obtaining services from edge devices (gateways or mobile nodes), users can experience lower latencies when requesting services than using cloud services located remotely on the Internet. A possible issue in using the cellular network for connectivity is that bandwidth to the services is shared between all covered devices, meaning that performances may be impacted by data transfer saturation, which is a likely occurrence given that the cellular network would be used also for general access to the Internet by all the users.

Opportunistic Computing can be useful in supporting Mobile Edge Computing Systems by offloading service requests to the devices in the area, reducing the risk

of overloading the access cellular network. Let's consider the scenario depicted in

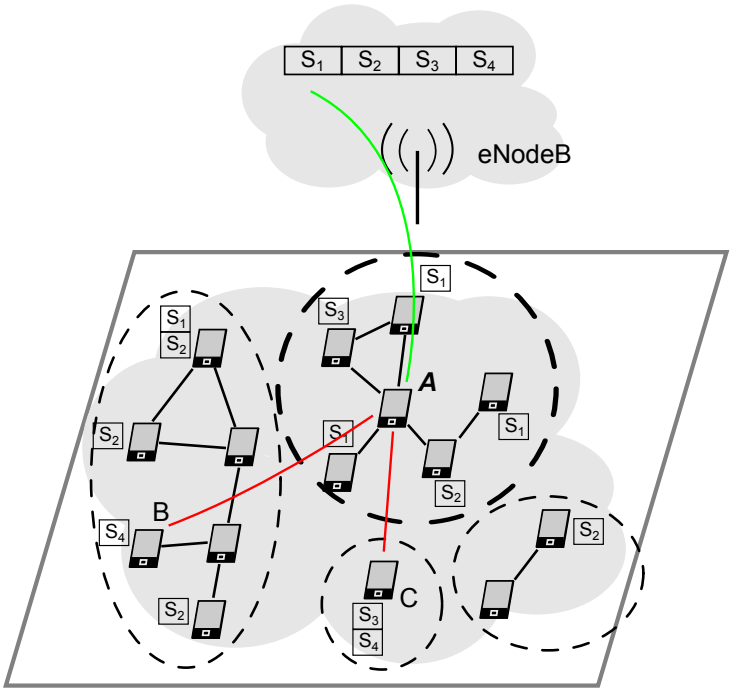


Figure 1.2: Edge Computing scenario example

Figure 1.2: in this case, the same mobile devices we had in the first scenario (in Figure 1.1) are in an area covered by LTE (4th generation cellular network) connectivity, provided by an LTE base station (eNodeB) that connects the devices to services s_1, s_2, s_3, s_4 provided by some remote cloud provider. Thanks to the cellular connectivity of mobile devices, those services are always available, so if device A needs service s_4, it can be immediately accessed. But, in the case congestion, it may be useful for A to obtain the same service from nodes B or C (assuming they can provide it), which are located nearby.

This scenario can be implemented by using an hybrid system that exploits Opportunistic Computing, where mobile devices can autonomously decide whether to solve a request accessing the cellular network or by offloading the request to another mobile device offering the same service depending on what solution would take the shortest time. Through this approach, it is possible to obtain better performances for the users and to avoid that the cellular network may incur in bandwidth

saturation, penalising also all other users that are using the connection for other purposes.

1.1.1 Our Approach

In this thesis solutions are presented, which demonstrate how Opportunistic Computing is an effective paradigm for the implementation of service provisioning systems for mobile devices.

First we propose a solution for environments where network infrastructure is absent. We designed a distributed support layer where users' mobile devices are self-organizing in an Opportunistic network and are capable to autonomously share resources in the form of services and access them remotely through direct connections. In this system, the devices use each contact opportunities with other devices to exchange information about the services that are available in the network, the devices that provide them and statistics about the relative mobility between each couple of devices, the load experienced by providers, and the performance in transferring data between each other. With this knowledge (context information), a device that needs to solve a service request uses a stochastic model able to produce an estimate of the expected service provisioning times of each alternative present in the network, including the impact of disconnections periods on the process duration itself, without considering it a failure state. In order to expand the possibility to solve service requests even in sparse network topologies and to have more alternative solutions to choose from, we implemented an algorithm to compose sequentially the services in order to obtain functionalities that are equivalent to the users requests. As an example, in the scenario shown in Figure 1.1, service s_4 is provided by devices B and C and we assume that the sequential composition of s_1 and s_2 is equivalent to s_4. Device A, when in need of service s_4, may decide to use a composition s_1, s_2 involving devices that are already connected with it in order to avoid waiting to meed devices B and C. These compositions are also evaluated with the stochastic model together with single service solutions in order to find, among a larger set of alternatives, the best one. To prove the effectiveness of this solution, we tested the system performances against other service provisioning decision algorithms that do not use context information to make decisions or service composition.

In the second step in the study of opportunistic service provisioning we focused on the analysis of the impact of the duration of connections between a service requester and a provider on the service provisioning process. We modified the model of the previous system in order to achieve precise estimates of the

duration of provisioning processes that involve a single provider by differentiating its definition depending on the distribution of the duration of the contact used to start the process itself. We also extensively evaluated the system performances in scenarios where devices follow real mobility traces. Finally we analysed how precise are the model estimates for both compositions of services and single services in scenarios differing by load of requests and size of the input and output data of the services.

Finally, in the second part of the thesis, we have integrated the Opportunistic Computing architecture into a Hybrid Mobile Edge Computing support system for service offloading in scenarios with LTE (4th generation cellular network) congestion. We construct a cooperation system between the mobile devices in an area (self-organized through Opportunistic Computing) and the LTE base station (eNodeB) that provides them with connectivity to access the remote service providers in the cloud. In this system the LTE eNodeB collects its own set of context information about the state of the infrastructure and about the performances of the remote cloud, while the mobile devices collect context information on the state of the Opportunistic Network. The mobile devices use their own context information and the information about the infrastructure, after requesting it to the eNodeB, to estimate the service provisioning times of all available service alternatives in the devices network and compare them to an estimate of the remote cloud service provisioning time to choose the best service provider alternative. We evaluate the performances of this solution, focusing specifically on the reduction of service provisioning time with respect to the case where only the remote cloud is used. Moreover, we highlight the efficiency of the system in reacting to spikes in infrastructure load without impacting on the performances the users' experience. We also test our system in scenarios where user requests are not uniform and tend on preferring certain subsets of offered services.

1.2 Thesis contribution

The main contribution of this thesis consists in the design, validation and evaluation of a system for service provisioning using Opportunistic Computing both as a mean to create service provisioning opportunities when a network infrastructure is unavailable, and as a solution to implement service provisioning offloading. This validation has been developed in three steps.

First of all we show that our service provisioning system performs particularly well in the analysed scenarios. In particular, thanks to The algorithm that nodes use to chose the estimated best service composition, and the prediction model

on which the algorithm is based, the designed system is able to avoid saturating the resources of the devices by distributing the load much better than other service provisioning solutions used as benchmarks. The service provisioning performances are far better than that obtained by the benchmarks in all conditions, and particularly when either the network starts becoming congested because of the traffic associated to the input/output data of the services, or when nodes become congested due to high numbers of generated requests by the users. We show that this holds true in all service composition cases, i.e. either when services are entirely available on individual devices (no service composition), or when composition is the only possibility to provide the required functionality, or in mixed cases. Improvement of our algorithm with respect to the alternative solutions can be as high as 50, 40, 75% in these three cases, respectively. These results were originally presented in [8].

To validate the model accuracy we analysed, via real mobility trace simulations and synthetic mobility traces the precision of the estimates that can be produced by a stochastic model using context information in an Opportunistic Computing system, that estimates service provisioning times based exclusively on local information at mobile devices, i.e., without centralised information. In scenarios varying in service request loads, sizes of service input and output parameters and length of the compositions used we also highlight the behaviour of the solution as a function of service availability. Results show that the model used to estimate service provisioning time is accurate, as the maximum estimation error is in the order of 15%. Then we compare the performances of the system against other alternatives using real mobility traces. We show that it can achieve up to 43% shorter average service provisioning times with respect to the closest benchmark. Therefore, the proposed model is a viable practical tool to implement efficient service provisioning between mobile nodes. These results were originally presented in [23].

At last we evaluated the efficacy of using Opportunistic Computing in a Mobile Edge Computing scenario. We present simulation results showing that our solution is capable of offering better service provisioning time than a system where service offloading is unavailable. We also explore the effects on service provisioning performances of user service preferences and cellular network congestion. We show that the proposed system is able to autonomously adapt to the level of congestion of the cellular network, avoiding to contribute to its saturation, and still preserving low service provisioning times to the users, even in cases where the cellular network is highly congested or users have strong preferences on certain services. These results were originally presented in [9] and [5].

With this thesis we show how, in pervasive environments, the use of the Opportunistic Computing paradigm can bring several advantages in creating service provisioning systems, both in absence or in presence of a supporting infrastructure.

1.3 Thesis Layout

The remainder of this thesis is organized as it follows. In Chapter 2 we present works in the literature that are related to ours in the themes of service composition, Opportunistic Computing solutions and Mobile Edge Computing. Then we present our proposals for using Opportunistic Computing in pervasive environments. In Chapter 3 we delineate the Opportunistic Computing system for service composition and provisioning in pervasive environments and show the results of the simulation experiments comparing it with different opportunistic computing systems that do not use context information. Then, in Chapter 4 we illustrate the system containing a refined stochastic model for service provisioning time estimates for compositions and single services (Section 4.2), and show (Section 4.4) through simulations its accuracy and effectiveness in a varied range of scenarios, also using real mobility traces. In the following chapter (Chapter 5) we present our hybrid system for Mobile Edge Computing and the results of our experiments to show the performance contribution in using Opportunistic Computing for Mobile Edge computing (Section 5.3). Finally, in Chapter 6 we sum up the results obtained in our work and how they relate to the future research directions.

2

Related Work

2.1 Service Composition

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 [7]. Service composition exploits a mechanism based on graph theory that allows to collect the knowledge of the services offered to evaluate the best alternative to use. Some recent research proposal take into accounts these aspects. [18] 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 [3]. 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.

2.2 Service Provisioning and Opportunistic Computing

In opportunistic computing the fact that nodes may not be reachable is considered as the rule, and proposed mechanisms are designed correspondingly. For example, in [38] 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. [29] proposes some heuristics for service

composition taking into account the last time of encounter between nodes and the reported load at providers.

In [11], [40] the problem is addressed using a stochastic analysis in order to find providers 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, and therefore they are not suitable to dynamic mobile scenarios where nodes disconnections are normal. A system that exploits composition of services coupled with a stochastic model to determine the system behaviour is MobIoT, presented in [15]. MobIoT is a service-oriented middleware for mobile participatory sensing, in which the mobility paths of the devices are estimated to decide whether their services should be shared in order to avoid redundancy of services in an area and waste of resources. In MobIoT this is achieved also composing already available services to obtain functionalities that would have to be covered by services from other unregistered devices in the area. MobIoT allows a new device to register its services only if it increases the sensing coverage of a physical attribute. To do so the model is used to elaborate the probability that another node with coverage for that attribute will cross devices' path. The middleware uses a registry connected to the infrastructure that maintains the information on the registered services in the area. With respect to MobIoT, we use a system policy that optimises service provisioning time and only uses information gathered by the devices themselves through opportunistic contacts.

Unlike these proposals, opportunistic computing does not assume that mobile nodes are well connected. The fact that nodes may not be reachable is considered as the rule, and proposed mechanisms are designed correspondingly. For example, in [38] fault tolerance is achieved through a middleware that creates parallel service compositions, in order to increase the probability for successful executions. [29] proposes some heuristics for service composition taking into account the last time of encounter between nodes and the reported load at providers. With respect to [38] and [29] our proposal is based on an analytical model for computing the expected completion time of each alternative, rather than on simple heuristics for selecting the composition. [26] 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 optimal 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 service provisioning time. With respect to [26], the solution we present, considers the possibility of composing different services.

Other approaches have been proposed in the literature where mobile nodes collaborate to provide services to each other. With respect to the solution proposed in [14], in this paper we put much more emphasis on the conditions of the mobile networking environment, and how they can be taken into account to select the most suitable service composition. We consider this focus particularly important, since we target a very dynamic mobile networking environment. In [17], authors propose a system where mobile nodes form a mobile cloud (similar to the opportunistic computing approach), but service between nodes is supported only for those nodes that are guaranteed to stay in contact during service execution. Therefore, our approach is much more general. Finally, [33] also assumes a network environment similar to ours. However, service provisioning times are not modelled, and therefore the proposed system is based on very simple heuristics, such as minimising execution times during contacts between nodes. Thanks to a more accurate modelling of providers availability, our solution can allocate service requests to available providers in a more efficient way.

2.3 Edge Computing

Research on Edge Computing spurs from the advancements in the branch of Mobile Cloud Computing showing a strong effort towards finding ways to relieve mobile devices from the execution of services, in favour of using the cloud infrastructure. In [28] authors present many different propositions for doing so, but in this approach the possibility of offloading computation or services execution to other mobile devices is rarely discussed. In [12] authors present cloud solutions where mobile devices provide services to other users, but such proposals consider mobility only as an exception. Depending on the application and objectives, mobile cloud computing solutions may differ in their architectures and in how the service behaves in case of service requests. Four main types of system architecture may be individuated: remote cloud solutions, local mobile clouds, edge computing solutions (or cloudlets), and hybrid solutions [1]. Systems for remote cloud computing offload functionalities (computation, storage, coordination) on the remote cloud. [21] describes numerous proposals for transferring computation functionalities from mobile devices to the cloud to improve performances or with the objective of saving energy. Local mobile clouds are systems where mobile devices collaborate in an area in order to provide functionalities to other participants, without using the infrastructure. MobiCloud [16] is a cloud framework in which mobile devices in a MANET are virtualized to service nodes or service broker, linked through a MANET routing protocol.

In Edge Computing, services and resources are also located dynamically on static devices connected to the wireless infrastructure in the vicinity of the mobile devices. For example [31] describes how to dynamically instantiate Virtual Machines for mobile users that can be accessed through wireless LAN networks. Hybrid solutions unite remote, local clouds and cloudlets to create systems where functionalities can be provided on different sites. Some initial proposals going into this direction have been proposed recently. For example SAMI [30] and MOCHA [35] are two examples of systems where computation activities needed by mobile devices are divided and distributed to sites of different nature. SAMI has the objective of minimizing the energy and monetary cost of computation when deciding to execute code on other mobile devices, a local cloudlet or on the remote cloud, while MOCHA uses information on latency and response times for all available remote cloud sites and the local cloudlet to decide where to execute code. However, none of these solutions exploit collaborative service provisioning among mobile devices, which is the key element of our approach, given that Edge Computing solutions can be supported by the Opportunistic paradigm by handling mobility as a functional part of the system and using the available resources offered by mobile devices in the area in the proximity of the edge system.

3

Service Selection and Composition in Opportunistic Networks

3.1 Overview

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. Another exploitable capability of opportunistic computing is to compose the services provided in the devices network, giving two main advantages: to add or substitute features that are momentarily or definitively absent due to unavailability of the devices that provide the service and to widen the choice of providers that can be used. This gives to the system the means to exploit policies to balance the load on the providers and network or obtain a possibility to achieve better performances.

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 impor-

tant 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.

In this chapter we propose 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 dynamically accumulates knowledge on resources and services from other participants in the network, finding also suitable composition of services to be used in alternative to the default offered services. 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). The system we propose is able to choose 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 chapter we select the alternative that minimises the expected service provisioning time, defined as the period between the choice of an alternative and the reception of the service results.

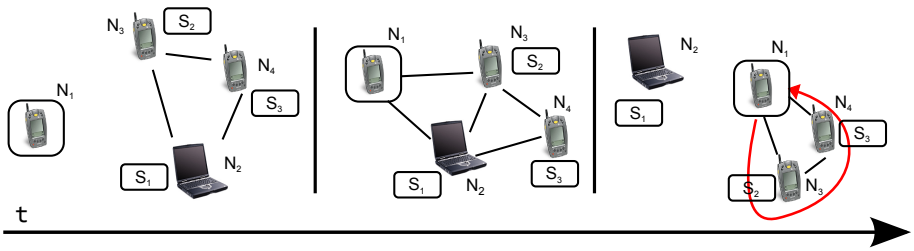


Figure 3.1: Service composition: an example

Figure 3.1 shows a possible scenario for 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.

In this chapter, we also present a set of simulation results, obtained using the TheOne simulator [20], which has been proposed for evaluating opportunistic environments and includes different mobility models. The simulator has been modified in order to make it suitable for the management of service composition. Thanks to the simulations, we compare the performance of our system against a set of alternative solutions. The rationale is considering lightweight solutions, that do not estimate service execution times as a function of the state of the devices network. We also evaluate the impact of service compositions on the system performance and how the knowledge collected by the devices influences them.

The comparison is done in a variety of scenario, differing by the network traffic caused by the service requests and the amount generated by the devices in the system.

In our tests, we show that our system performances particularly well in the analysed scenarios. In particular, our algorithm avoids saturating the resources of the network by distributing the load much better than other solutions. The resulting service provisioning time is far better than that obtained by the benchmarks in all conditions, and particularly when either the network starts becoming congested because of the traffic associated to the input/output data of service execution, or nodes becomes congested due to increased number of requests. We show that this holds true in all service composition cases, i.e. either when services are entirely available on individual nodes (no service composition), or when composition is the only possibility to provide the required function, or in mixed cases. Improvement of our algorithm with respect to the alternative solutions can be as high as 50, 40, 75% in these three cases, respectively, when network and provider saturation don't become unmanageable for the benchmark algorithms.

The simulations also highlight the effectiveness of the approach we propose, especially in presence of a large amount of requests. Our system provides better performance in terms of provisioning time in particular when this is most needed, i.e. when a high load of requests is generated. For instance, with respect to the policy selecting the first available alternative, our system reduces the provisioning time up to 86%, and the average load on providers up to 40%.

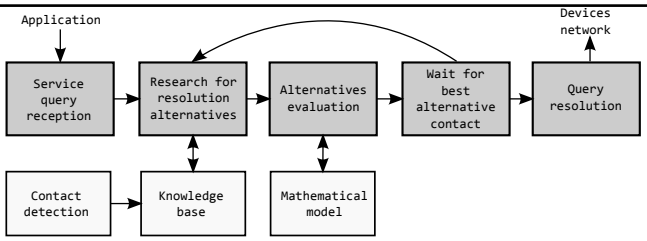


Figure 3.2: Service request resolution algorithm

This chapter is organized as follows. The overall architecture of the system is defined in Section 3.2 and the policy for selection of the composition is introduced in Section 3.3 and 3.4. Section 3.5.2 discusses the experimental results. Finally, concluding remarks are reported in Section 3.6.

3.2 System Architecture

To describe the system behaviour we show how a service request is managed and what is the logical decision process to select the components that form the composition involved in the resolution of the request. Fig. 3.2 shows a graphical representation of the algorithm used by seekers. Let us consider a chosen seeker running an application that at some point generates a request. The system running at the seeker first finds, through the knowledge base (i.e., a local storage with information about available services, managed as described in the following of the section), all possible compositions that would satisfy the request. Then, the service provisioning time of each composition is estimated, and the one providing the minimum provisioning time is selected. If the first provider in the composition is in contact with the seeker, the execution of the first component starts. Otherwise, the seeker waits to encounter this provider. In the meanwhile, it may encounter other nodes, which, as explained in the following, could result in updating the local knowledge at the node. In this case, the selected composition is re-evaluated, and possibly modified according to more refined knowledge acquired by the seeker. Eventually, the service composition starts and proceeds until the application request is satisfied. Between two consecutive components, output parameters of the former are passed to the next provider as input parameters.

Let us discuss more in detail how the key components of Fig. 3.2 are realised. Let us first consider the block indicated as "research for resolution alternatives": once a service request is generated at a seeker, the system searches for reso-

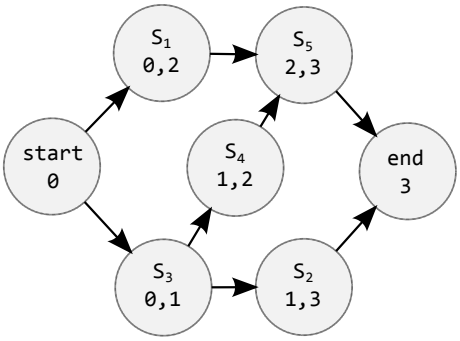


Figure 3.3: Service Graph

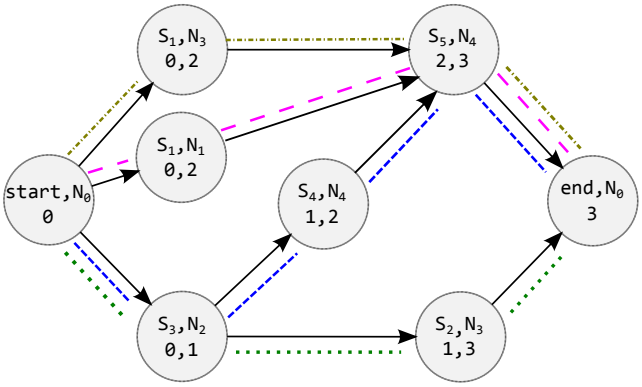


Figure 3.4: Composition graph

lution alternatives to satisfy the request. To this end, the system searches in the local knowledge base, in order to collect all the known service components that might be used and the statistics needed for the evaluation of the alternatives. The knowledge base is updated by each node upon encountering other nodes. In particular, for each encountered node it stores (i) the set of provided services, (ii) an estimate of the computation time for each provided service, (iii) an estimate of the number of service requests generated by the node upon encounter, and the respective size of exchanged parameters, (iv) an estimate of contact and inter-contact times, and (v) an estimate of the average throughput available when the two nodes encounter. Specifically, for the reason explained in Section 3.3, upon encountering, nodes exchange the first two moments of service computation time, as well as the list of provided services. The number and size of requests, contact

and inter-contact times and average throughout estimates are simply monitored by each node without requiring exchange of information. This information is sufficient for the chosen node to estimate the service provisioning time.

After collecting the service components from the knowledge base, the system builds a Service Graph out of them (an example is shown in Fig. 3.3) where vertices are components, and edges represent the fact that two components can be executed sequentially. Note that in the Service Graph there is not yet information about which nodes provide components, which is added in the following logical step (the Composition Graph). Each path connecting two vertices of the graph shows a possible composition to satisfy the application request. Each service component 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. For the sake of simplicity, in the following we assume that these types are codified by integer values, furthermore we will consider acyclic compositions, i.e. compositions where the same components cannot appear twice. To ensure this, any Service Graph we consider will contain only services s_j such that $I_j < O_j$. For instance, Fig. 3.3 shows a set of service components $\{s_1, s_2, s_3, s_4, s_5\}$ linked by their type dependencies together with two special components $Start$ and End, representing the start and the end points of the service composition corresponding to the considered request.

Given that each component may be offered by different providers known by the seeker, there can be different composition alternatives depending on the chosen providers. To identify these alternatives, a Composition Graph is created (as in Fig. 3.4), where, for each component, vertices are created for all known providers offering the component. On the resulting graph, each path from component $Start$ to component End is a suitable composition. The graph is weighed, and weights are the key elements provided by our analytical model to estimate the service provisioning time, as explained in Section 3.3. Note that the graph may change from node to node, as we assume it is built based on information available locally and collected through direct pairwise contacts, and not on global information.

In the "alternatives evaluation" block, the list of alternatives taken from the graph is then evaluated by the system through a mathematical model that takes each composition to estimate its expected service provisioning time. How the model does these estimates is described in Section 3.3. After the evaluation, the system ranks the alternatives, choosing the one with the least expected service provisioning time.

In the "wait for best alternative contact" block, the seeker, if it is not currently in contact with the chosen provider for the first component, waits for a contact with it. Otherwise the decision process is over and the service request is queued to be

sent to the provider. If the seeker has to wait for the first provider, it continues to monitor the state of the network upon new contacts. Any new contact triggers a new exchange of information between the nodes. This information may alter the classification computed previously, so, in this case, the system goes back to the alternatives evaluation phase to update the ranking of the compositions.

3.3 Modelling service execution

This section introduces the stochastic model exploited to estimate the execution time of a single component service. Modelling the execution time of a single component service is the building block to evaluate the completion time of a composition. Starting from this, we will model service composition in Section 3.4.

Table 3.1: System Elements

N	Nodes Set
S	Service components Set
$n_i \in N$	i-th node of the network
$s_i \in S$	i-th service
I_i	Input type of the i-th service
O_i	Output type of the i-th service

Table 3.2: Random variables

R	Completion time of service i on provider j, for a service required by node h
W	Time h spent by h to establish a contact with j
B	Time required to upload data of size k from node h to node j
θ	Time necessary to download data of size k from node h to node j
DS	Execution time of service i in provider j
DQ	Queue waiting time for a service request in provider j
L	Number of queries in a batch received by a provider
T_C	Contact duration of node h and node j
T_{IC}	inter contact period of node h and node j

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

Table 3.3: Network Statistics

δ	Contact rate between node h and node j
δ'	Intercontact rate between node h and node j
ρ	Average load on provider j
μ	Average service time for service i on provider j
λ	Query arrival rate on provider j
V	Average throughput between nodes h and j

- *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.

3.3.1 Assumptions

This section defines some basic assumptions we introduce to reduce the complexity of the mathematical analysis. We will exploit some notations used also for the definition of the model which are summarized in tables 3.1,3.2,3.3.

Assumption. *For each service request q generated by the seeker $n_h \in N$, it is possible to satisfy the request by a composition of services allocated on the provider nodes $n_j, ..., n_m \in N$ which have been previously contacted by n_h*

This assumption guarantees that the nodes satisfying the user query may be detected at the node where the query is generated, with the knowledge it has previously gained from the network.

Assumption. *Each seeker n_h and each provider n_j have the same knowledge about the service s requested by n_h e and provided by n_j.*

This property is needed to model symmetric service knowledge between seekers and providers so that the the types of input/output parameters known by seekers and providers match

3.3.2 Contacting the service provider W

We introduce the random variables T_C and T_{IC} 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 and T_{IC} follow exponential probability distributions with rates δ and δ'. As shown by real trace analysis presented, for example, in [13, 39], 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. [36, 26]). Since a node cannot know beforehand the values of δ and δ', 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. 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} otherwise (and, under our assumption, its expected value is equal to $E[T_{IC}]$ due to the memoryless property of the distribution).

The expected value of W is recomputed at each connection/disconnection of the two nodes, changing from 0 to $E[T_{IC}]$ when a disconnection occurs or from

$E[T_{IC}]$ to 0 in case of a connection establishment.

3.3.3 Data transfer times B and θ

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.

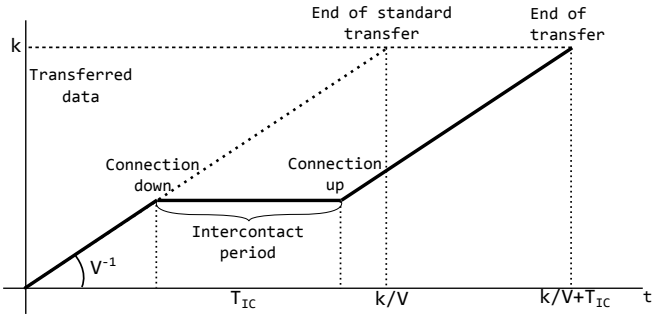


Figure 3.5: Modelling Data Transfer

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. The input data transfer is characterized by starting while a contact is established with the provider and, hereafter, we refer as B for each data transfer time starting during a contact period. B depends on k, the size of data to transfer, on V and on the number N of contacts required to transfer data. Figure 3.5 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 $T_C(i)$ the length of the i-th contact event between n_i and n_j, the probability that N is equal to n is thus:

$$P\{N = n\} = P\{\sum_{i=1}^{n} T_C(i) < \frac{k}{V} \leq \sum_{i=1}^{n+1} T_C(i)\}$$

If we condition to the number of contact events required to complete the data transfer, B 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.

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

To associate the distribution of N to B, we can note that $\sum_{i=1}^{n} T_{IC}(i)$ and $\sum_{i=1}^{n} T_C(i)$ are Erlang random variables with average rates respectively δ' and δ and we will call them $S_{IC,n}$ and $S_{C,n}$.
Given that we are interested in getting a formulation of the expected value of B, we can use the Laplace transform of B to ease the analisys. Also, we can formulate the expected value of N using its Z-transform $\Pi_N(z)$.
The transform of B is $L_B(s) = L_{k/V+S_{IC,N}}(s) = e^{-sk/V} L_{S_{IC,N}}(s)$ and thanks to a property of the Laplace transform of an Erlang variable which has a random variable as the number of components, we can write $e^{-sk/V} L_{S_{IC,N}}(s) = e^{-sk/V} \Pi_N(L_{T_{IC}}(s))$, successfully separating the number of intercontacts from the intercontacts distribution.
This composite function can be changed thanks to the definition of Z-transforms into $e^{-sk/V} \sum_{n=0}^{\infty} P\{N = n\} * (L_{T_{IC}}(s)^N) = e^{-sk/V} \sum_{n=0}^{\infty} P\{N = n\} * \left(\frac{\delta'}{\delta'+s}\right)^n$
The probability of $N = n$ depends on the probability that n contact periods won't be enough to finish transfering the data, but that with another contact the transfer will end:

$$P\{N = n\} = P\{\sum_{i=1}^{n} T_C(i) < \frac{k}{V} \leq \sum_{i=1}^{n+1} T_C(i)\}$$

We considerate the sum of n contact periods as an Erlang with average rate δ that we call $S_{C,n}$:

$$P\{N = n\} = P\{S_{C,n} < \frac{k}{V} \leq S_{C,n} + T_{C,n+1}\} =$$

$$= P\{S_{C,n} + T_{C,n+1} \geq \frac{k}{V} \wedge S_{C,n} < \frac{k}{V}\} =$$

$$= \int_0^{k/V} P\{x + T_{C,n+1} \geq \frac{k}{V} \wedge S_{C,n} = x\} dx =$$

$$= \int_0^{k/V} P\{T_{C,n+1} \geq \frac{k}{V} - x | S_{C,n} = x\} * P\{S_{C,n} = x\} dx =$$

We substitute the probability values with the cumulative probability distribution of the intercontact times $F_{T_{IC}}$ and the density function of $S_{IC,n}$:

$$= \int_0^{k/V} (1 - F_{T_C}(\frac{k}{V} - x)) * f_{S_{C,n}}(x) dx =$$

$$= \int_0^{k/V} (1 - (1 - e^{-\delta(\frac{k}{V} - x)})) * \frac{\delta^n x^{n-1} e^{-\delta x}}{(n-1)!} dx =$$

$$= \int_0^{k/V} e^{-[\delta(\frac{k}{V} - x) + \delta x]} * \frac{\delta^n x^{n-1}}{(n-1)!} dx =$$

$$= \int_0^{k/V} e^{-\delta \frac{k}{V}} * \frac{\delta^n x^{n-1}}{(n-1)!} dx =$$

$$= e^{-\delta \frac{k}{V}} \delta^n * \int_0^{k/V} \frac{x^{n-1}}{(n-1)!} dx = e^{-\delta \frac{k}{V}} * \frac{(\frac{\delta k}{V})^n}{n!}$$

substituting this value to the formulation of the trasform of B, we have:

$$L_B(s) = e^{-sk/V} \sum_{n=0}^{\infty} e^{-\delta \frac{k}{V}} * \frac{(\frac{\delta k}{V})^n}{n!} * \left(\frac{\delta'}{\delta' + s}\right)^n =$$

$$= e^{-(s+\delta)k/V} \sum_{n=0}^{\infty} \left(\frac{\delta k \delta'}{V(\delta' + s)}\right)^n * \frac{1}{n!} =$$

Given that $\sum_{n=0}^{\infty} c^n/n! = e^c$, we have:

$$= e^{\frac{-(s+\delta)k}{V}} * e^{\frac{\delta k \delta'}{V(\delta'+s)}} = e^{\frac{-(s+\delta)k(\delta'+s)+\delta k \delta'}{V(\delta'+s)}} = e^{\frac{-(\delta+\delta'+s)ks}{V(\delta'+s)}}$$

To obtain the expected value of B, we calculate the value of the derivate function of the transform:

$$L'_B(s) = \frac{k}{V} * \left(\frac{(\delta\delta' + (\delta' + s)^2)}{(\delta' + s)^2} * e^{(-\frac{ks(\delta+\delta'+s)}{V(\delta+s)})}\right)$$

And we calculate it for $s = 0$:

$$E[B] = (-1) * L'_B(0) = \frac{k}{V} * \left(\frac{\delta\delta' + (\delta')^2}{(\delta')^2}\right) = \frac{k}{V} * \left(1 + \frac{\delta}{\delta'}\right)$$

Using the transform $L_B(s)$ it is also possible to extract the value of the variance of B, given that we only need to find the second moment $E[B^2]$ of B and substract

the square of the expected value. We can obtain the second moment by calculating the second derivative $L''_B(s)$ of the laplace trasform of B and put $s = 0$.
With simple calculations we have that:

$$L''_B(s) = \frac{k(k(\delta\delta' + (\delta' + s)^2)^2 + 2\delta\delta' V(\delta' + s)}{V^2(\delta' + s)^4} * e^{\frac{-(\delta+\delta'+s)ks}{V(\delta'+s)}}$$

$$E[B^2] = L''_B(0) = \frac{k(k(\delta\delta' + (\delta')^2)^2 + 2\delta V(\delta')^2}{V^2(\delta')^4} =$$

$$= \frac{k(k(\delta\delta')^2 + k(\delta')^4 + 2k\delta(\delta')^3 + 2\delta V(\delta')^2)}{V^2(\delta')^4} =$$

$$= \frac{k(k\delta^2 + k(\delta')^2 + 2k\delta\delta' + 2\delta V)}{V^2(\delta')^2} =$$

$$Var(B) = E[B^2] - E[B]^2 = \frac{k(k\delta^2 + k(\delta')^2 + 2k\delta\delta' + 2\delta V)}{V^2(\delta')^2} -$$

$$-\frac{k^2}{V^2}\left(\frac{(\delta')^2 + 2\delta\delta' + \delta^2}{(\delta')^2}\right) = \frac{2k\delta}{V(\delta')^2}$$

To calculate the expected value of the time to transfer data not knowing whether to start from an intercontact or contact period, we use thee results from B:

- $\theta = B$ if there is a contact time.
- $\theta = B + T_{IC}$ otherwise, thanks to the memorylessness propery of exponential distributions.

If we indicate with p_C and p_{IC} the probabilities that the previous cases happen, we can calculate the expected value of θ as:

$$E[\theta] = E[B] * p_C + E[B + T_{IC}] * p_{IC} =$$

We substitute the values;

$$= \frac{k}{V}\left(1 + \frac{\delta}{\delta'}\right) * \frac{E[T_C]}{E[T_C] + E[T_{IC}]} + \left(\frac{k}{V}\left(1 + \frac{\delta}{\delta'}\right) + \frac{1}{\delta'}\right) * \frac{E[T_{IC}]}{E[T_C] + E[T_{IC}]} =$$

$$= \frac{k}{V}\left(1 + \frac{\delta}{\delta'}\right) * \frac{\delta'}{\delta' + \delta} + \left(\frac{k}{V}\left(1 + \frac{\delta}{\delta'}\right) + \frac{1}{\delta'}\right) * \frac{\delta}{\delta' + \delta} =$$

$$= \frac{1}{\delta' + \delta} * \left((\delta' + \delta) * \frac{k}{V} * \left(1 + \frac{\delta}{\delta'}\right) + \frac{1}{\delta'} * \delta\right) =$$

$$= \frac{k}{V} * \left(1 + \frac{\delta}{\delta'}\right) + \frac{\delta}{\delta'} * \frac{1}{\delta' + \delta}$$

We use the same approach to extract, from the definition of B, the value of the variance of θ:

$$Var(\theta) = E[\theta^2] - E[\theta]^2$$

- Second moment:

$$E[\theta^2] = E[B^2] * p_C + E[(B + T_{IC})^2] * p_{IC} =$$

$$= E[B^2] * p_C + E[B^2 + 2B * T_{IC} + T_{IC}^2] * p_{IC} =$$

$$= E[B^2] * (p_C + p_{IC}) + 2 * E[B] * E[T_{IC}] * p_{IC} + E[T_{IC}^2] * p_{IC} =$$

$$= E[B^2] + 2 * E[B] * E[T_{IC}] * p_{IC} + E[T_{IC}^2] * p_{IC}$$

- Square of the expected value:

$$E[\theta]^2 = (E[B] * p_C + (E[B] + E[T_{IC}]) * p_{IC})^2 =$$

$$= E[B]^2 * p_C^2 + (E[B]^2 + E[T_{IC}]^2 + 2 * E[B] * E[T_{IC}]) * p_{IC}^2 +$$

$$+2(E[B] * p_C * (E[B] + E[T_{IC}]) * p_{IC}) =$$

$$= E[B]^2 * (p_C^2 + p_{IC}^2 + 2p_C p_{IC}) + 2 * E[B] * E[T_{IC}] *$$

$$*(p_{IC}^2 + p_C p_{IC}) + E[T_{IC}]^2 * p_{IC}^2 =$$

$$= E[B]^2 + 2 * E[B] * E[T_{IC}] * (p_{IC}^2 + p_C p_{IC}) + E[T_{IC}]^2 * p_{IC}^2$$

With the second moment and the square of the expected value, we can show the formulation of $Var(\theta)$:

$$Var(\theta) = E[\theta^2] - E[\theta]^2 =$$

$$= E[B^2] + 2 * E[B] * E[T_{IC}] * p_{IC} + E[T_{IC}^2] * p_{IC}$$

$$-E[B]^2 - 2 * E[B] * E[T_{IC}] * (p_{IC}^2 + p_C p_{IC}) - E[T_{IC}]^2 * p_{IC}^2 =$$

We note that $E[B^2] - E[B]^2$ is equal to $Var(B)$.

$$= Var(B) + 2E[B]E[T_{IC}](p_{IC} - p_{IC}^2 - p_C * p_{IC}) + E[T_{IC}^2] * p_{IC} - E[T_{IC}]^2 * p_{IC}^2 =$$

$$= Var(B) + 2E[B]E[T_{IC}](p_{IC}(1 - p_{IC} - p_C)) + E[T_{IC}^2] * p_{IC} - E[T_{IC}]^2 * p_{IC}^2 =$$

Given that $(1 - p_{IC} - p_C) = 0$, we can delete $2E[B]E[T_{IC}]$.

$$= Var(B) + E[T_{IC}^2] * p_{IC} - E[T_{IC}]^2 * p_{IC}^2 =$$

We substitute the explicit values of the probabilities and the expected values:

$$= \frac{2k\delta}{V(\delta')^2} + \frac{2}{(\delta')^2} * \frac{\delta}{\delta' + \delta} - \frac{1}{(\delta')^2} * \frac{\delta^2}{(\delta' + \delta)^2} =$$

$$= \frac{2k\delta}{V(\delta')^2} + \frac{2\delta\delta' + 2\delta^2 - \delta^2}{(\delta')^2(\delta' + \delta)^2} = \frac{2k\delta}{V(\delta')^2} + \frac{2\delta\delta' + \delta^2}{(\delta')^2(\delta' + \delta)^2} =$$

$$= \frac{2k\delta}{V(\delta')^2} + \frac{(\delta' + \delta)^2 - (\delta')^2}{(\delta')^2(\delta' + \delta)^2} =$$

$$= \frac{2k\delta}{V(\delta')^2} + \frac{1}{(\delta')^2} - \frac{1}{(\delta' + \delta)^2}$$

3.3.4 Queue waiting time DQ

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 $M^{[X]}/G/1$ [37] 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 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 can be computed [37] if we assume that the first two moments of the general distribution of the service execution time DS (with expected value d and $d^{(2)}$ its second moment) and of the random variable L (with expected value l and $l^{(2)}$ 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 we extract the average rate λ of the query batches arrivals to the provider and compute the average load ρ of the provider as $\lambda * l * d$. Starting from these values, the expected value of DQ can be computed, as shown in [37], as:

$$E[DQ] = \frac{\lambda l d^{(2)}}{2(1-\rho)} + \frac{l^{(2)} d}{2l(1-\rho)}$$

3.3.5 Service execution time DS

The random variable DS 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 by collecting the execution times of that service and transmits this value in each opportunistic contact.

3.4 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.

As described in Section 3.2 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 3.6 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

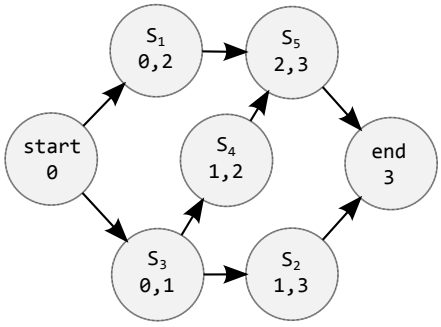


Figure 3.6: The Service Graph

with the nodes (providers) offering it. The *Composition Graph*, shown in Figure 3.7

is defined by 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 3.7.

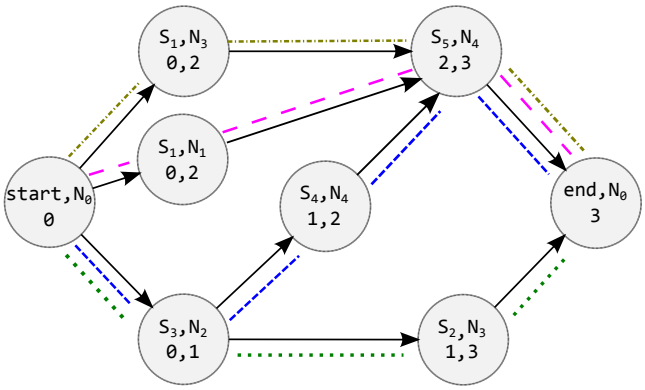


Figure 3.7: The Composition Graph

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 and intercontact rates δ' 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.

3.4.1 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 3.3, 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 3.3), 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 to find the best alternative.

3.5 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 [20]. The experiments exploit a set of mobility traces generated according to the RandomWayPoint model, modified as discussed in [4] in order to avoid problems related to the initial transient phase of the mobility model.[1]

3.5.1 Composition Load Evaluation

In the first set of experiments we evaluate the system in scenarios with different kinds of requested compositions. The main parameters of the simulations are described in Table 3.4.

Table 3.4: 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.
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

[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 [4]

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 component 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.

3.5.2 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 3.8 and in Figure 3.9 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

Figure 3.8: Average completion time (*general case*)

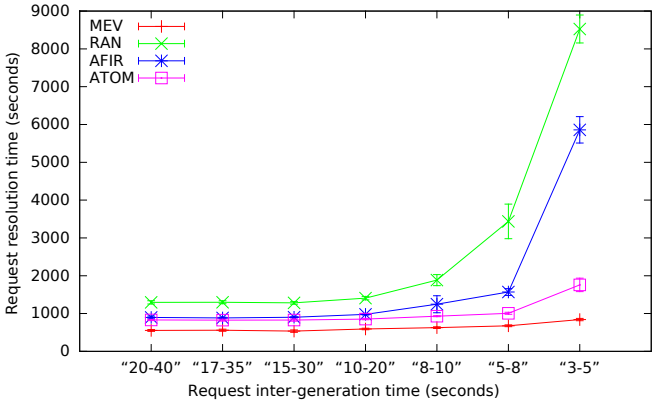
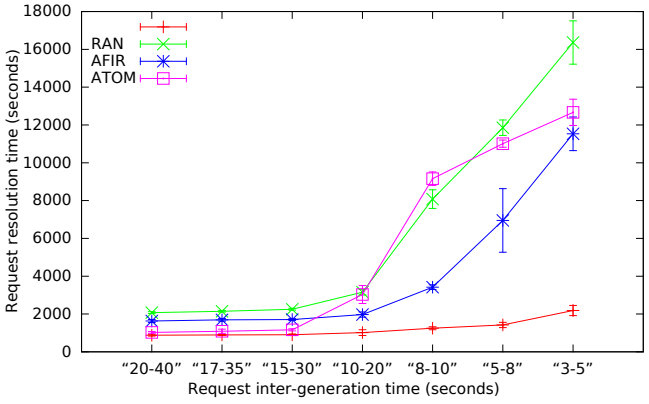


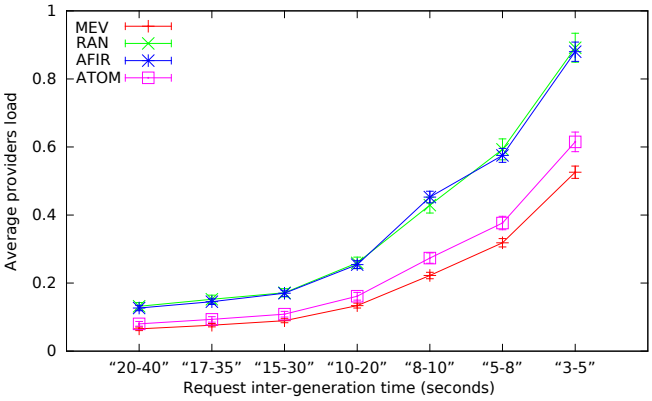
Figure 3.9: Average completion time (*case08*)



slightly larger than *MEV* in the general case, despite it requires a single component 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 3.10), *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. Neverthe-

Figure 3.10: Average provider load (*general case*)



less, this result in an average load even lower than in *ATOM*, showing that *MEV* is able to efficiently distribute the additional load.

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.

3.5.3 Data Transfer and Load System Evaluation

In this section we analyse the behaviour of our system in different simulated scenarios differing for request load and data transfer sizes for long periods of time. The simulation parameters are described in 3.5. For each simulation scenario, we averaged the results upon 5 simulation runs. We consider 30 devices that move in a square with sides $500m$ long with the devices having a $90m$ transmission range and a $2\ Mbps$ maximum bandwidth, so there is a high probability that each node can have many different direct contacts with other nodes at any time during the simulation.

Each simulation run covers $400000s$ of simulated time, with the first $10000s$ allotted to only the exchange of mobility data to build the knowledge base of the nodes. Then service requests are generated for $360000s$ (100 simulated hours). The last $30000s$ are left to complete the service resolution processes.

Table 3.5: Default simulation parameters

Simulation runs	5
Number of nodes	30
Simulation space	$500m \times 500m$
Total simulation time	$400000s$
Mobility warm-up period	$10000s$
Statistics warm-up period	$360000s$
Request generation phase duration	$390000s$
Connectivity range	$90m$
Transmission speed	$2Mbps$
Density of each service	25%
Input type range	$i \in [0, 7]$
Requests output type range	$o \in [1, 8], o > i$

Each service can be provided by 25% of the nodes of the network. The behaviour of the policies are evaluated in three main scenarios. In the first one, all the requests are evaluated and solved using exclusively single service executions (we will call this scenario “Single”), so that compositions cannot be used (and the RAN and ATOM policies coincide). In the second one (the “Comp” scenario) requests cannot be solved using a single service (ATOM is not evaluated in this scenario because of the inability to abide to the scenario limitation). Finally, we consider a “Mixed" scenario, where all four policies can use any composition length, according to the policy behaviour.

For all scenarios, we have run two sets of simulations where different amounts of requests are generated. In the “5-8” scenario, a new request is generated after the previous one, after a time uniformly distributed in the interval between 5 and 8 seconds has passed. Each request is then assigned to a randomly chosen node, also following a uniform distribution, who will act as a seeker. The “20-40” scenario is similar to “5-8”, but the intervals between request generations are chosen between 20 and 40 seconds.

Finally, we consider different sizes for the input/output parameters of the service components: 80 KBytes, 320 KBytes and 1280 KBytes for the “5-8" setting and 80 KBytes, 320KBytes, 1280 KBytes and 5120 KBytes for the “20-40" setting. In each simulation, the input and output data sizes are the same for all services and requests. All the results shown are the average results of the 5 independent simulation runs executed for each scenario. We also show the confidence intervals, with a 95% confidence level.

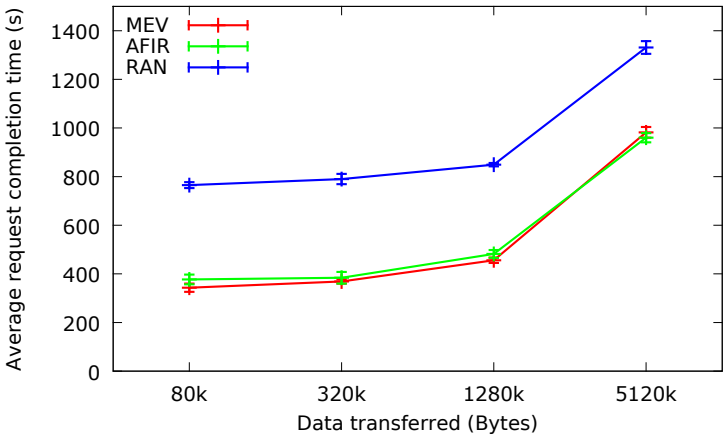


Figure 3.11: Service provisioning time by data transfer sizes, scenario “Single”, load 20-40

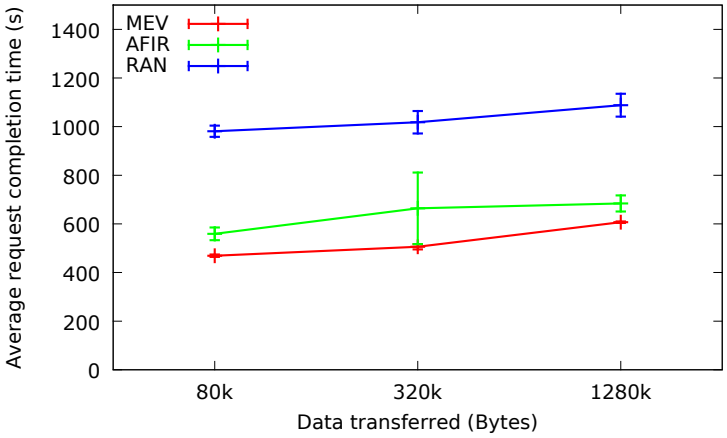


Figure 3.12: Service provisioning time by data transfer sizes, scenario “Single”, load 5-8

In the "Single" scenario we force each policy to use the least possible number of service requests to solve any generated user request. In Figure 3.11 we can see the average service provisioning time for various data transfer size settings with a low request load (inter-generation interval “20-40”). AFIR and MEV show similar performance, and both drastically outperform RAN, with up to 50% less time spent

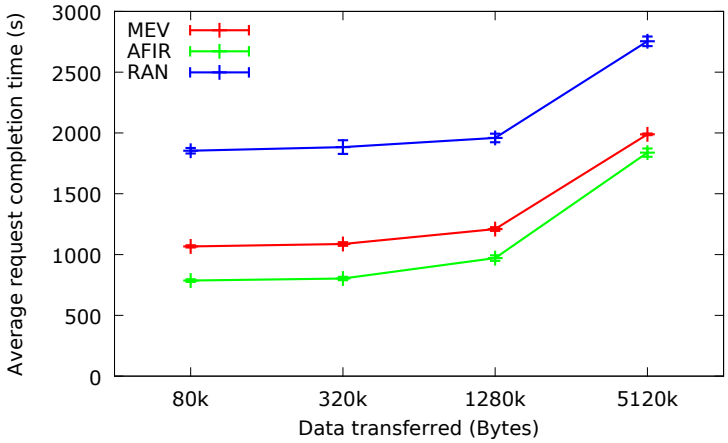


Figure 3.13: Service provisioning time by data transfer sizes, scenario “Comp”, load 20-40

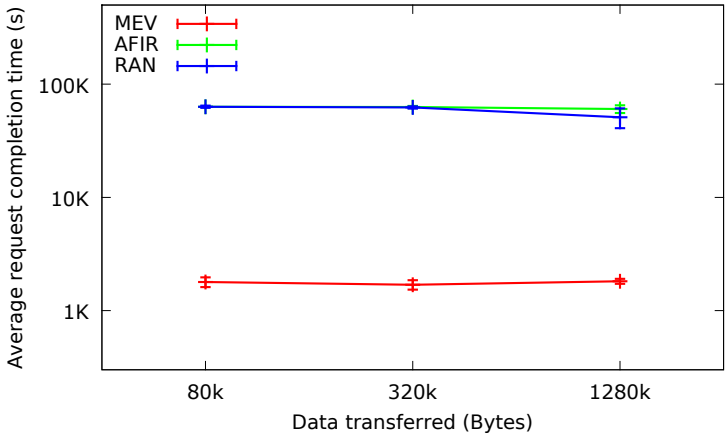


Figure 3.14: Service provisioning time by data transfer sizes, scenario “Comp”, load 5-8

on small input data sizes.
Once we also set a higher load of requests to the system, we can see (in Figure 3.12) that the advantage in using MEV over AFIR becomes more apparent, with MEV coping better with the increase of the queue waiting time, keeping at least at 45% the time saved against RAN.

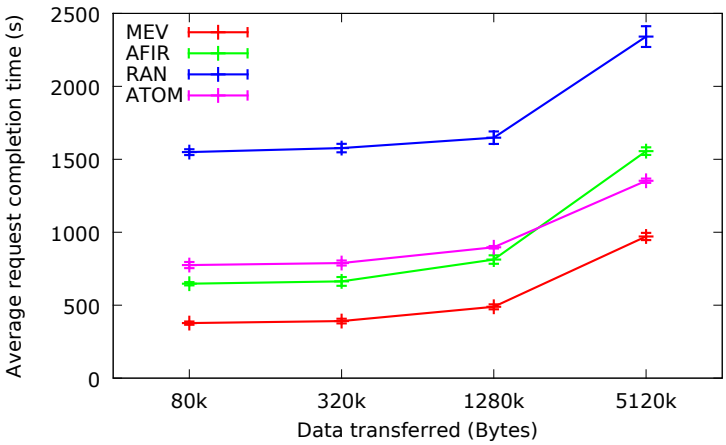


Figure 3.15: Service provisioning time by data transfer sizes, scenario “Mixed”, load 20-40

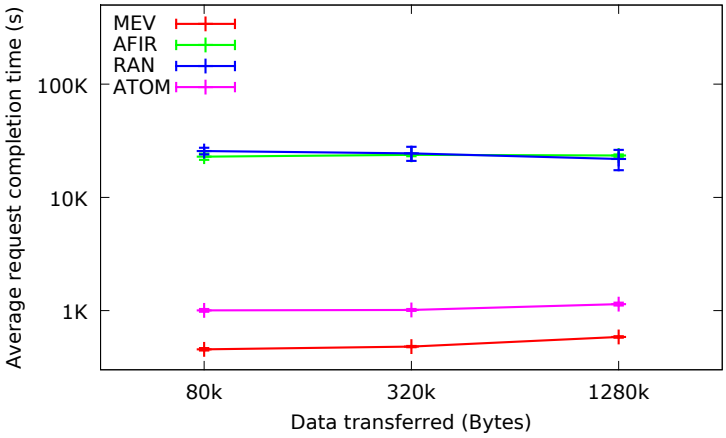


Figure 3.16: Service provisioning time by data transfer sizes, scenario “Mixed”, load 5-8

In the “Comp” scenario, the unavailability of single service solutions favours the use of solutions that can handle the load of requests without flooding the network or that can save time on each data transfer phase in the compositions. We can see in Figure 3.13 that with low request load, AFIR manages to outperform MEV exploiting the knowledge of the state of the connections between providers after

each service is executed. This advantage is amplified by low data transfer sizes that make possible to exchange data before the used contact may end. This is confirmed noting that with bigger data sizes, the difference between MEV and AFIR decreases. When compared against RAN, MEV still manages to achieve up to 40% better performances. With an heavier load scenario (as seen in Figure 3.14), both AFIR and RAN cannot handle the flow of requests given by the use of compositions, with average times 40 times greater than the ones achieved by MEV (note the logarithmic scale on the y axis).
In the “Mixed” scenario, each policy has the opportunity to use any possible composition length to solve a request (apart from ATOM that uses only single service executions). In this kind of scenario, MEV can exploit the wide range of choices keeping into account the stability of the network. In Figure 3.15, we can see how MEV outperforms the other policies even at low loads, saving near to 40% of time in respect with AFIR and 75% with respect to RAN with low transfer data sizes. With bigger data sizes, MEV outperforms ATOM with at least 35% shorter times, exploiting the fact that ATOM isn’t aware of the computing capabilities of the providers.
In Figure 3.16 we see the high load scenario, where the difference between MEV and the other policies is even more remarkable, with AFIR and RAN suffering from overloading the nodes, and ATOM being incapable of giving good performances generating the least possible amount of requests between all policies, with MEV achieving times near to 50% shorter (note again the logarithmic scale in the y axis).
The under-performing of AFIR is caused by its choices of compositions for solving requests. Without any guidance about the load state and the service execution performance of each provider in the composition paths, AFIR is not able to exploit the knowledge of the state of the network at each execution step. Also, without any reference about the lengths of the chosen compositions, AFIR tends to saturate the provider with service requests.

3.6 Summary

In this chapter 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 that our system outperforms other policies in a varied range of scenarios in terms of average resolution times of users’ requests. In particular, our solution is drastically better as the network becomes more and more

congested, either due to load at service providers or congestion at the network level. This is because our solution is able to detect those condition, spreads the load more intelligently, and thus avoids creating bottlenecks in the service provisioning process.

4

Service Provisioning In Mobile Environments through Opportunistic Computing

4.1 Overview

In this chapter, we investigate how, upon a service request at a given device our solution aims at identifying the composition that minimises the time required by the seeker to obtain the results of the composed service (throughout referred to as *service provisioning time*). Our solution takes into account the fact that nodes providing service components may be only intermittently connected with the seeker and with each other, because of the nature of the underlying mobile environment. To this end, we modify the stochastic model presented in the previous chapter in order to find estimates of the service provisioning times for the available composition alternatives that accurately represent the influence of the devices mobility on each specific request resolution process. This new approach is particularly important for single component services, where the connection status between a seeker and its chosen provider impacts the duration of all phases that compose the resolution process. The structure of the underlying system for the model is the same described in Section 3.2, meaning that there is no additional requirements for the devices that are part of the network.

As seen in the previous solution, nodes exchange, upon each encounter, information about the sought and provided services, and their current load. Each node also monitors data transfers to estimate the available bandwidth for communications, as well as time intervals between contact opportunities with other nodes. Based on this information, each node builds a localised view of services available in the network. Upon a service request, a seeker can thus identify the known available compositions, and estimate the time required by using each of them. To this end, we develop an analytical model that estimates online the service provisioning time when using each of the possible compositions.

We validate the model accuracy via trace-driven simulations using reference mobility traces using TheOne [20], which is a reference simulation environment for opportunistic networks. For greater flexibility, we also use synthetic mobility traces to validate the model against varying service request loads, sizes of service input and output parameters and length of the compositions used. We also highlight the behaviour of the solution as a function of service availability. Results show that the model used to estimate service provisioning time is accurate, as the maximum estimation error is in the order of 15%. Then we compare the performances of the system against other alternatives using real mobility traces. We show that it can achieve up to 43% shorter average service provisioning times with respect to the closest benchmark. Therefore, the proposed model is a viable practical tool to implement efficient service provisioning between mobile nodes.

The chapter is organized as follows. The revised stochastic model is presented in Section 4.2 and Section 4.3 for single and multiple component service executions respectively. In Section 4.4 we characterise the performance of the proposed service provisioning approach. Finally, concluding remarks are reported in Section 4.5.

4.2 Modelling service provisioning time

This section introduces the stochastic model exploited to estimate the service provisioning time. The basis of this model and its structure are taken from the model in the previous chapter, but the formulation of all its parts is different in order to better describe the relations between the phases recurring in the service provisioning process and the evolution of the state of the network during the process itself. Without loss of generality, we present the model by focusing on the provisioning time of a service requested by a chosen seeker. For the sake of explanation, let us first assume that this service can be provided directly by a chosen provider encountered by the seeker (i.e., the service composition is made of 1 service component only). We then extend it to the general case of composed services. Note that in the following, when required, we denote with h the index of the chosen seeker, and j the index of the chosen provider. Most of the variables used in the model refer to the pair (chosen seeker, chosen provider). When this dependence is clear and straightforward, we omit to use the indices h, j with these variables, to make the notation simpler. The service provisioning time in this case (hereafter denoted with R) is made up of five consecutive phases, as follows:

- *Contact of the service provider* (W). This is the time needed by the seeker to encounter the provider after the point in time when the service request is

generated. It is determined by the inter-contact time between the seeker and the provider. This value depends on their relative mobility.

- *Data transfer* (Input Time B, Output Time θ). This is the time needed to transfer the input parameters from the seeker to the provider and the output parameters from the provider to the seeker (after the execution time is complete). In the latter case, we also include in this phase the time required by the provider to meet the seeker from the point in time when the service execution is complete. Note that in an opportunistic network data transfers between two nodes may be affected by connection disruptions, due to the nodes mobility, but also by transfer from and to other nodes that use the same shared medium, or even by other concurrent transfers between the seeker and the provider involving other requests. This implies that, in general, both the duration of contact and inter-contact times impact on the duration of these phases.
- *Queue waiting time* (DQ). Once onto the provider, actual execution may be delayed due to previous pending requests. 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). This is the time to execute a service on the provider. It depends on both its computational capabilities and on the type of the service.

Each of these phases can be modelled as a separate random variable as we will describe in the following of the section. Since they are sequential, we obtain for a single component service the following expression:

$$R_{single} = W + B + DQ + DS + \theta \tag{4.1}$$

For the case of a service composition made of n components, the service provisioning time can be expressed as follows:

$$R_{comp} = W + B + \sum_{i=1}^{n}(DQ_i + DS_i + \theta_i) \tag{4.2}$$

In (4.2), W and B refer to the time to meet the first provider in the composition and transfer the input parameters to it, respectively. DQ_i and DS_i are the length of queuing at the i-th provider and executing the i-th component, respectively. θ_i represents the time required by provider i to encounter provider $i+1$ (or to encounter the seeker, for provider n), and transfer to it the output parameters of component i which become input parameters of component $i+1$ (or, for provider n, to transfer the final output parameters to the seeker).

In the following of the section we explain how to estimate the expected time taken by these phases. We discuss the derivation of B and θ last, as this requires a number of steps. Through these estimates and using (4.1) and (4.2) each seeker can estimate the expected service provisioning time of the available compositions, and pick the best one. The components of (4.2) need some slight modifications with respect to what we derive for the case of single service compositions, as discussed in Section 3.2.

4.2.1 Contacting the service provider W

We introduce the random variables T_C and T_{IC} modelling, respectively, the contact and inter-contact times between two nodes h and j. For each pair of nodes, we assume that contact and inter-contact times between those nodes are independent and identically distributed (i.i.d.). We also assume that contact and inter-contact times of different pairs of nodes are independent of each other. Finally, we assume that the variables T_C and T_{IC} follow exponential distributions with rates δ and δ'. As shown by real trace analysis presented, for example, in [13, 39], although controversial, exponential contact and inter-contact times is one of the possibilities, and is a common assumption in the literature on opportunistic networking and computing (e.g. [26, 36]). Since a node cannot know beforehand the values of δ and δ', each node computes an estimate of these values by averaging the values of contact and inter-contact times with other nodes collected during opportunistic contacts.

The time for seeker h to contact the generic service provider j, is denoted by the random variable W. This is equal to 0 if, at the time when the evaluation is done, h and j are in contact, while it is equal to the residual inter-contact time T_{IC} otherwise (and, under our assumption, its expected value is equal to $E[T_{IC}]$ due to the memoryless property of the exponential distribution). The expected value of W is therefore:

$$E[W] = \begin{cases} \frac{1}{\delta'} & \text{if } h \text{ and } j \text{ are not connected} \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

4.2.2 Service execution time DS

The random variable $DS_{s_i,j}$ for the time needed to execute service component s_i on a provider j is influenced both by the device computational capabilities and by the implementation of the requested service component. We assume each provider keeps an estimate of the expected value of $DS_{s_i,j}$ based on the previous

local executions, and sends this value to encountered nodes. Similarly, providers also keep an estimate of the second moment of $DS_{s_i,j}$ and send it to other nodes upon encounters. This is used in the estimation of the waiting time DQ, as explained next.

4.2.3 Queue waiting time DQ

We assume that the seeker locally generates requests addressed to the provider according to a Poisson process with rate λ. Therefore, when the provider is encountered, in general a batch of requests may have been accumulated, whose input parameters need to be transferred. To model this behaviour, we model the service provider with a $M^{[X]}/G/1$ queue. As shown in [37], this is exact when (i) requests are generated according to a Poisson process, (ii) inter-contact times are exponential, and (iii) all requests stored at the seeker are transferred to the provider during a contact. Under our assumptions, hypothesis (iii) may not hold. However, we still use the $M^{[X]}/G/1$ model, and assess the approximation level by validating the results obtained using this model against simulation in Section 3.5.2.

Let us denote with L the number of requests in a batch received by the provider. The expected value of the random variable DQ can be computed [37] based on the first two moments of the service execution time $DS_{s_i,j}$ (with expected value d, average service rate μ and second moment $d^{(2)}$) and of the random variable L (with expected value l and second moment $l^{(2)}$). These values can be estimated by monitoring the batches arriving to the provider and the executed services. In addition, the provider estimates the rate λ of the request batches and computes the average load ρ of the provider as $\lambda * l * d$. Starting from these values, the expected value of DQ can be computed, as shown in [37], as:

$$E[DQ] = \frac{\lambda l d^{(2)}}{2(1-\rho)} + \frac{l^{(2)} d}{2l(1-\rho)} \tag{4.4}$$

4.2.4 Data transfer time B and θ

Unlike most analytical models in opportunistic networks, we assume that the throughput available to nodes during contact times is finite. Therefore, in our model we need to take into account the possibility that data transfer may be interrupted by disconnections, and therefore needs to be resumed at the next contact event. In the following, we denote with V the average throughput experienced between the chosen seeker and provider and assume it is estimated by the two nodes through a

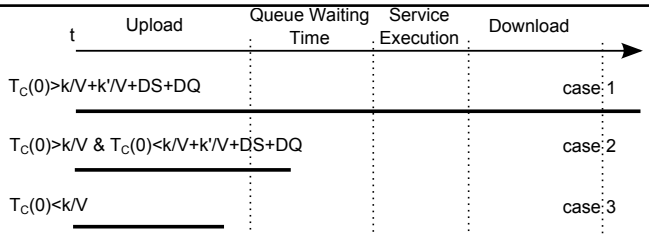


Figure 4.1: Accounted lengths of the first contact period

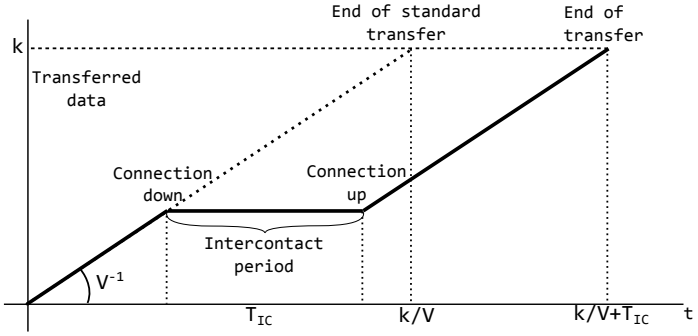


Figure 4.2: Modelling Data Transfer

conventional smoothed average algorithm using the throughput samples obtained during actual data transfers between them, we also denote with k and k' the size of respectively of the input and output data to be transferred of the requested service (for more clarity in the following sections, we will refer as V to the average throughput between each couple of nodes and we will refer as k and k' to the size of all input/output data transfers). Fig. 4.2 shows an illustrative example where one disconnection occurs during the transfer of the input data of size k. In addition to the time needed to actually transfer data between the nodes, we have to take into account additional inter-contact times between consecutive contact events.

First of all, we analyse the case of data transfers in single service executions. In this scenario we identified three cases to model depending on the time when the first contact (that we refer as $T_C(0)$) used to transfer data ends. In Fig. 4.1 we can see a representation (the continuous line represents the duration of the first contact): the first case (*case 1*) is a scenario where all the phases of the execution can be completed without any interruption. The second case (*case 2*) has only the first data transfer (the one used to transfer the input data of size k for the service execution) completing without interruptions, while there is at least

one disconnection before the seeker completely receives the result of the service execution of size k'. The last case (*case 3*) considers the presence of at least one disconnection before the input data transfer phase is completed.

In the following we provide the expressions for B and θ in the three cases, considering their formulations involving seeker h and provider j. Then, we model the probabilities of the cases, i.e. p_1, p_2 and $p_3 = 1 - p_1 - p_2$. Note that B is the same in cases 1 and 2. The expected values of B and θ can be immediately derived by applying the law of total probability:

$$E[B] = E[B|case1, 2] * (p_1 + p_2) + E[B|case3] * p_3 \tag{4.5}$$

$$E[\theta] = E[\theta|case1] * p_1 + E[\theta|case2] * p_2 + E[\theta|case3] * p_3 \tag{4.6}$$

Analysis of case 1

In *case 1*, with no interruptions, the input and output transfer times for a request from seeker h for a service provided by node j, depend on the throughput available between the nodes V, the sizes of the input and output parameters (k_{data}, k'_{data} respectively, which value depends on the requested service) and the size of the queued data that has to be transferred between h and j before the input and the output data transfers can be started (respectively called k_{queue} and k'_{queue}). The value of k_{queue} can be directly observed by the seeker during the evaluation of the alternatives, while k'_{queue} is estimated as the average size of queued data in j addressed to h at the end of service executions. Therefore, B and θ for *case 1* can be modeled as follows:

$$B|case1 = \frac{k}{V} \quad \theta|case1 = \frac{k'}{V} \tag{4.7}$$

Where $k = k_{data} + k_{queue}$ and $k' = k'_{data} + k'_{queue}$. These values of $\frac{k}{V}$ and $\frac{k'}{V}$ can be called the *net transfer time* for B and θ, as they are the minimum estimated data transfer times without the presence of any interruption.

Analysis of case 2

The second case (*case 2*) considers the scenario where the first data transfer can be done during the contact, but the rest of the process cannot be completed in time. The input time B is the same already analysed in (4.7).

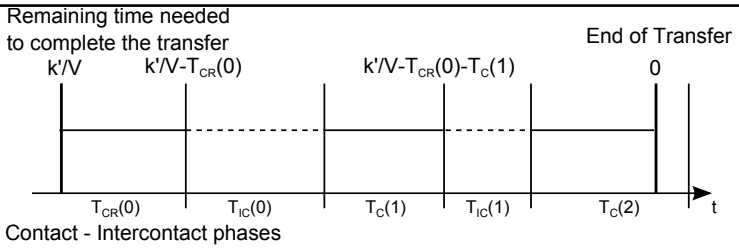


Figure 4.3: Phases of an output transfer starting during a contact, with 2 disconnections afterwards

We analyse θ by considering the two possible cases: the case where the (first) contact [1] ends while the transfer of output parameters is already ongoing (case 2A), and the case when the contact ends before the output data transfer has started. In the first sub-case, the first contact can be used to start the output data transfer, but the same contact is not long enough to complete the entire transfer (otherwise it would fall in case 1). Otherwise, if the first contact ends before the service execution is completed, θ may start during a contact period (case 2B) or an inter-contact period (case 2C).

θ in case 2A can be expressed as in (4.8). The key characteristics of case 2A are that (i) θ starts during the first contact between the seeker and provider (ii) then θ always includes the following inter-contact time ($T_{IC}(0)$ in (4.8), and (iii) it then finally may include an additional number $N_{2A} \geq 0$ of inter-contact times. An example of case 2A can be seen in Figure θ, where is represented an output transfer phase with 2 disconnections. As shown in the following, N_{2A} can be characterised based on the number of contacts needed to transfer k' bytes minus the data already transferred during the first contact. Considering that θ is made up of the net transfer time (k'/V) plus the additional inter-contact times needed to complete the transfer, it can be expressed as follows:

$$\theta|case2A = \frac{k'}{V} + T_{IC}(0) + \sum_{i=1}^{N_{2A}} T_{IC}(i) \tag{4.8}$$

The characteristics of case 2B are that (i) θ starts during a contact time , and (ii) it may include a number N_{2B} of inter-contact times. As shown below, as we assume that contact times are exponential and thus memoryless, N_{2B} can be

[1] Here first contact still refers to the contact between the seeker and provider at the beginning of the input data transfer phase. It is not the first contact during which the output data transfer starts.

characterised based on the number of contacts needed to transfer k' data (this is why N_{2B} is stochastically different from N_{2A}). Therefore, θ in case 2B can be expressed as follows:

$$\theta|case2B = \frac{k'}{V} + \sum_{i=1}^{N_{2B}} T_{IC}(i) \tag{4.9}$$

Finally, the characteristics of case 2C are that (i) θ starts during an inter contact time, and (ii) it may then include an additional number $N_{2C} \geq 0$ of additional inter-contact times. It is easy to see that, as we have assumed that contact times are memoryless, N_{2C} is stochastically equivalent to N_{2B}. Therefore, by denoting with $T_{ICR}(0)$ the residual duration of the inter-contact time during which it starts, θ in case 2C can be expressed as follows:

$$\theta|case2C = \frac{k'}{V} + T_{ICR}(0) + \sum_{i=1}^{N_{2B}} T_{IC}(i) \tag{4.10}$$

To find the expected values for θ in this three sub-cases, we need to derive the distributions of N_{2A} and N_{2B}.

Lemma 1. *The probability that, in case 2A, θ includes $N_{2A} = n$ additional inter-contact times (after the first one that it always includes) is as follows:*

$$P\{N_{2A} = n\} = e^{-\delta\frac{k+k'}{V}} * \frac{(\frac{\delta k'}{V})^{n+1}}{n+1!} * \frac{(1-\rho)\mu}{\delta + \mu(1-\rho)} \tag{4.11}$$

Proof. We denote by $T_{CR}(0)$ the part of the first contact time between the seeker and the provider, that is the initial part of θ. Therefore, the remaining transfer time is $k'/V - T_{CR}(0)$. In order for N_{2A} to be equal to n, the remaining transfer time must be longer than n contact times but shorter than $n+1$ contact times. Therefore,

$$\begin{aligned} &P\{N_{2A} = n\} = \\ &\quad = P\{\sum_{i=1}^{n} T_C(i) < \frac{k'}{V} - T_{CR}(0) < \sum_{i=1}^{n+1} T_C(i) | T_{CR}(0) < \frac{k'}{V}\} \end{aligned} \tag{4.12}$$

Given that all contact periods $T_C(i)$ between h and j are independent and identically distributed exponential random variables, we can consider $\sum_{i=1}^{n} T_C(i)$ as an Erlang variable $S_{C,n}$ with n components, each with rate δ.

Based on this result, we can thus condition on a specific duration of $T_{CR}(0)$ and apply the law of total probability:

$$P\{N_{2A} = n\} =$$

$$= \int_0^{\frac{k'}{V}} P\{S_{C,n} < \frac{k'}{V} - t < S_{C,n} + T_C(n+1)|$$

$$|T_{CR}(0) = t\} * P\{T_{CR}(0) = t\}dt =$$

$$= \int_0^{\frac{k'}{V}} \int_0^{\frac{k'}{V}-t} \left(1 - F_{T_C(n+1)}\left(\frac{k'}{V} - t - x\right)\right) *$$

$$f_{S_{C,n}}(x) * f_{T_{CR}(0)}(t)dxdt =$$

Recalling that $T_C(n+1)$ is exponentially distributed, while $S_{C,n}$ follows an Erlang distribution, we obtain

$$F_{T_C(n+1)}(t) = 1 - e^{-\delta t} \tag{4.13}$$

and

$$f_{S_{C,n}}(t) = \frac{\delta^n t^{n-1} e^{-\delta t}}{(n-1)!} \tag{4.14}$$

Therefore, we obtain:

$$P\{N_{2A} = n\} =$$

$$= \int_0^{\frac{k'}{V}} \int_0^{\frac{k'}{V}-t} e^{-\delta(\frac{k'}{V}-t-x)} * \frac{\delta^n x^{n-1} e^{-\delta x}}{n-1!} *$$

$$* \frac{\delta e^{-\delta(t+\frac{k}{V})} * (1-\rho)\mu}{\delta + \mu(1-\rho)} dxdt =$$

$$= e^{-\delta\frac{k+k'}{V}} * \frac{(\frac{\delta k'}{V})^{n+1}}{n+1!} * \frac{(1-\rho)\mu}{\delta + \mu(1-\rho)} \tag{4.15}$$

□

Similarly, we obtain the following expression for N_{2B}:

Lemma 2. *The probability that, in cases 2B and 2C, θ includes $N_{2B} = n$ inter-contact times is as follows:*

$$P\{N_{2B} = n\} = P\{N_{2C} = n\} =$$

$$= P\{\sum_{i=1}^{n} T_C(i) < \frac{k}{V} < \sum_{i=1}^{n+1} T_C(i)\} = e^{-\delta\frac{k}{V}} * \frac{(\frac{\delta k}{V})^n}{n!} \tag{4.16}$$

Thanks to the results in Lemma 1 and Lemma 2, it is possible to find a closed formula for the expected values of θ in all sub-cases:

Lemma 3. *The expected value of θ case2A is equal to:*

$$E[\theta|case2A] = \frac{k'}{V} + \frac{1}{\delta'} + \frac{1}{\delta'} * e^{-\delta\frac{k+k'}{V}} * \\ * (e^{\delta\frac{k'}{V}} * (\delta\frac{k'}{V} - 1) + 1) * \frac{(1-\rho)\mu}{\delta + \mu(1-\rho)} \tag{4.17}$$

Proof. This is straightforward by the formula for θ in case 2A seen in (4.8) and the distribution of N_{2A}, seen in (4.11). □

Lemma 4. *The expected value of $\theta|case2B$ is equal to:*

$$E[\theta|case2B] = \frac{k'}{V} + E\left[\sum_{i=1}^{N_{2B}} T_{IC}(i)\right] = \frac{k'}{V}\left(1 + \frac{\delta}{\delta'}\right) \tag{4.18}$$

Proof. The proof is analogous to the one of Lemma 3. □

Lemma 5. *The expected value of $\theta|case2C$ is equal to:*

$$E[\theta|case2C] = \frac{k'}{V} + E[T_{IC}(0)] + E\left[\sum_{i=1}^{N_{2C}} T_{IC}(i)\right] = \frac{1}{\delta'} + \frac{k'}{V}\left(1 + \frac{\delta}{\delta'}\right) \tag{4.19}$$

Proof. This result directly follows from (4.18) with the addition of $E[T_{IC}(0)]$ (due to the fact that inter-contact times are exponential, and therefore $E[T_{ICR}(0)] = E[T_{IC}(0)]$), that gives us the formula in the lemma. □

To conclude the analysis of case 2 we need to derive the probabilities of the three subcases, 2A 2B and 2C, conditioned to the fact that we are in case 2[2]. The probability of case 2A is provided in the following Lemma:

Lemma 6. *The probability of the first contact lasting enough to complete the input data transfer, but not enough to reach the beginning of the output data transfer, is:*

$$p(Case2A) = \frac{e^{-\delta\frac{k}{V}} * \mu(1-\rho)}{\delta + \mu(1-\rho)} \tag{4.20}$$

[2] We derive the probabilities of cases 1, 2 and 3 later, in Section 4.2.5

Proof. This probability can be written as the probability that the first contact time $T_C(0)$ is longer than the time for the transfer of the input parameters (k/V) plus the queuing time at the provider (DQ_j) plus the service computation time (DS), but not long enough to also include the transfer of the output parameters (θ). By recalling that we are in case 2 and, therefore, $T_C(0)$ is not shorter than k/V (captured in case 3) and shorter than the total service provisioning time without any disconnection (case 1), we can write:

$$p(Case2A) = P\{\frac{k}{V} + DQ_j + DS < T_C(0)| \\ |\frac{k}{V} < T_C(0) < \frac{k}{V} + DQ_j + DS + \frac{k'}{V}\}$$

That, isolating the durations of DQ_j and DS becomes:

$$p(Case2A) = \int_0^\infty P\{\frac{k}{V} + x < T_C(0)| \\ |\frac{k}{V} < T_C(0) < \frac{k}{V} + x + \frac{k'}{V} \wedge DQ_j + DS = x\} * P\{DQ_j + DS = x\}dx = \\ = \int_0^\infty (1 - F_{T_C(0)}(\frac{k}{V} + x)) * f_{DQ_j+DS}(x)dx$$

Using the expression for the density of $DQ_j + DS$ and the formula for the cumulative probability of $T_C(0)$

$$p(Case2A) = \int_0^\infty (1 - (1 - e^{-\delta(\frac{k}{V}+x)})) * (1-\rho)\mu e^{-\mu(1-\rho)x}dx = \\ = e^{-\delta\frac{k}{V}}\mu(1-\rho)\int_0^\infty e^{-x(\delta+\mu(1-\rho))}dx = \frac{e^{-\delta\frac{k}{V}} * \mu(1-\rho)}{\delta + \mu(1-\rho)}$$

□

The probabilities of sub-cases B and C are calculated using the complement of the probability of case A "weighted" with the steady state probabilities of contact and inter-contact phases. Specifically, Case 2B corresponds to the case where the end of the service computation time falls in a contact time. For simplicity, we assume that this occurs with a probability equal to the steady state probability that a random point in time falls inside a contact. Analogously, case 2C occurs with the steady state probability that a random point in time falls inside an inter-contact. Therefore we obtain:

$$p_{case2B} = (1 - p_{case2A}) * \frac{E[T_C]}{E[T_C] + E[T_{IC}]} \tag{4.21}$$

$$p_{case2C} = (1 - p_{case2A}) * \frac{E[T_{IC}]}{E[T_C] + E[T_{IC}]} \tag{4.22}$$

Using this probabilities we can find the expected value for $\theta|case2$ through the law of total probability.

Analysis of case 3

In the third and final case (*case 3*), we capture the scenario where the contact cannot last long enough to complete the input data transfer, that may happen for large inputs to transfer or short contact periods. In this case B still starts during a contact periods, but, before the end of the time needed to transfer the input data (k/V), a first inter-contact period (denoted as $T_{IC}(0)$) happens. Then, after resuming the transmission, an additional number $RN_{IC} \geq 0$ of inter-contact periods $T_{IC}(i)$ may occur. Therefore the formulation for $B|case3$ can be written as:

$$B|case3 = \frac{k}{V} + T_{IC}(0) + \sum_{i=1}^{RN_{IC}} T_{IC}(i) \tag{4.23}$$

To find the expected value of $B|case3$, we need a formulation for $P\{RN_{IC} = n\}$. This is not the same to the one seen for $P\{N_{2A} = n\}$ in Lemma 1, given that the elapsed time of the first contact period before the start of B is unknown. We can then approximate the residual of the first contact with the entire contact duration, obtaining the following lemma.

Lemma 7. *If there is at least a connection interruption during the input data transfer phase, the probability of having exactly n additional interruptions, other than the first one, during the phase is equal to:*

$$P\{RN_{IC} = n\} = \frac{e^{-\delta \frac{k}{V}} \left(\delta \frac{k}{V}\right)^{n+1}}{n+1!} \tag{4.24}$$

Proof. The result is straightforward from the formulation of $P\{RN_{IC} = n\}$ once we consider this probability as the probability of n contact periods $\sum_{i=1}^{n} T_C(i)$ to be long enough to transfer the data that could not be transfered during the first contact $\frac{k}{V} - T_C(0)$. □

Thanks to Lemma 7, we can find the expected value of $B|case3$:

Lemma 8. *The expected value of a data transfer phase that starts during a contact period, but having at least one disconnection period, is equal to:*

$$E[B|case3] = \frac{k}{V}\left(1 + \frac{\delta}{\delta'}\right) + \frac{e^{-\frac{\delta k}{V}}}{\delta'} \tag{4.25}$$

Proof. Using Lemma 7, the expected value can be found using the law of total probability. □

To formulate $\theta|case3$, likewise to what we have seen in cases 2B and 2C, we divide the formulation into in two sub-cases 3A and 3B depending on the connection state between the seeker and the provider at the start of the phase. I

In case3A the output data transfer phase starts during a contact and in case3B during an inter-contact period, without any assumption on the number of disconnections occurring in the phase. So, $\theta|case3A$ can be formalized exactly as seen in (4.9) and $\theta|case3B$ can be formulated as in (4.10).

For simplicity, in this case we don't keep track in the analysis of the time evolution of the previous phases of service provisioning time with respect to the contact and inter-contact processes. Therefore, we approximate p_{case3A} and p_{case3B} using the steady state probabilities of contact and inter-contact phases:

$$p_{case3A} = \frac{E[T_C]}{E[T_C] + E[T_{IC}]} \tag{4.26}$$

$$p_{case3B} = \frac{E[T_{IC}]}{E[T_C] + E[T_{IC}]} \tag{4.27}$$

With the formulation of θ in the two sub-cases and their probability, we obtain the following lemma:

Lemma 9. *The expected value of* $\theta|case3$ *is*

$$E[\theta|case3] = \frac{k'}{V}\left(1 + \frac{\delta}{\delta'}\right) + \frac{\delta}{\delta'(\delta' + \delta)} \tag{4.28}$$

Proof. This immediately follows by using the results in Lemma 4 and Lemma 5 to then apply the law of total probability. □

4.2.5 Single service cases probabilities

Having derived all the components of the service provisioning time in all cases, we now need to derive expressions for the probabilities of the three cases in which we have divided the analysis.

Remember that case 1 is the case when the entire service provisioning time R is shorter than the first contact time between seeker and provider. Therefore the probability of case 1 is $P\{R < T_C(0)\}$. The following lemma provides a closed form for this probability.

Lemma 10. *The probability $p_1 = P\{R < T_C(0)\}$ of a single-service request resolution process, involving a seeker j, a provider h and the execution of service s_i, ending during the same contact event when the input data transfer phase started can be approximated as:*

$$p_1 = \frac{\mu(1-\rho)e^{-\delta\frac{k+k'}{V}}}{\delta + \mu(1-\rho)} \tag{4.29}$$

Proof. This is straightforward by recalling that, in case 1, $R = \frac{k}{V} + DS_{s_i,j} + DQ + \frac{k'}{V}$. We can condition on the value of $DS_{s_i,j} + DQ$ and apply the law of total probability. We obtain:

$$\begin{aligned} P\{R < T_C(0)\} = \\ &= P\left\{\frac{k}{V} + DS_{s_i,j} + DQ + \frac{k'}{V} < T_C(0)\right\} = \\ &= \int_0^\infty \left(1 - F_{T_C}\left(\frac{k+k'}{V} + t\right)\right) * f_{DS_{s_i,j}+DQ}(t)dt \end{aligned} \tag{4.30}$$

that gives the formulation of p_1. □

We describe the value of the probability of the third case p_3 before the second because it allows us to achieve a simple formulation for the probability of the second case. The value of p_3 is the probability that the residual of the first contact $T_{CR}(0)$ is shorter than the time k/V needed to transfer the input data without interruptions.

Lemma 11. *The probability of the first contact terminating before finishing a input data transfer is:*

$$p_3 = P\{T_{CR}(0) < \frac{k}{V}\} = F_{T_C(0)}\left(\frac{k}{V}\right) = 1 - e^{-\frac{\delta k}{V}} \tag{4.31}$$

Proof. This is straightforward, thanks to the assumption that contact times are exponentially distributed. □

The probability of the third case can be immediately evaluated as $p_2 = 1 - p_3 - p_1$. Its expression is provided in the following Lemma.

Lemma 12. *The probability of the first contact terminating after a input data transfer but before the end of output data transfer is:*

$$p_2 = 1 - p_3 - p_1 = e^{-\frac{\delta k}{V}} - \frac{\mu(1-\rho)e^{-\delta\frac{k+k'}{V}}}{\delta + \mu(1-\rho)} \tag{4.32}$$

With the expected values of B and θ for all three cases and the probabilities of the cases, we can evaluate the expected value of B and θ as shown in Sections 4.5 and 4.6.

4.2.6 Data transfer for service compositions

To complete the analysis, we now extend the derivation of the service provisioning time to the case of service composition. As discussed before, the service execution time is expressed by (4.2), i.e.:

$$R_{comp} = W + B + \sum_{i=1}^{n}(DQ_i + DS_i + \theta_i)$$

where, W is the time needed to contact the first provider of the composition, B is the first service input data transfer time, DQ_i is the queue waiting time before the execution of the $i-th$ service, that last for a period DS_i, and finally θ_i is the time needed to transfer the results of the $i-th$ service execution to the next provider, or, in the case of the final results, to the seeker.

For W, B, DS_i and DQ_i we use the same formulation used in the single service executions, as they have no differences.

For each i, θ_i clearly depends on the seeker-provider pair corresponding to component i. Also in this case, to simplify the notation, without loss of generality, we omit the indices of the specific pair, and provide the expression of θ_i for a generic component i, that we simply refer as θ. In the formulation of θ, the first factor includes the time to encounter the next provider, and possibly the time needed to transfer previously queued data (e.g., other input/output parameters) to be exchanged between these two nodes. For simplicity, we keep this factor as a model parameter (called *average transfer queue time* TQ), and we assume that nodes estimate its value by monitoring previous data transfers between the same nodes. On the other hand, the second part is analogous to θ for single service composition in case 2B, i.e. when it starts at the beginning of contact[3], and therefore can be expressed as in (4.9). Therefore, θ becomes as follows:

$$\theta = TQ + \frac{k'}{V} + \left\lceil \sum_{i=1}^{N_{2B}} T_{IC}(i) \right\rceil \tag{4.33}$$

Given that TQ is a non-random parameter, we can calculate the expected value of θ, using again the results provided in Lemma 4, as:

[3] More precisely, in case 2B, θ starts *during* a contact. However, the average values become the same due to the assumption of contact times being exponentially distributed

$$E[\theta] = TQ + \frac{k'}{V}\left(1 + \frac{\delta}{\delta'}\right) \tag{4.34}$$

4.3 Choice of the best alternative

The model in Section 4.2 allows us to compute all possible service provisioning times for all alternatives. To make this process more efficient, we use the Composition Graph (Fig. 3.4), weighting the edges of the graph according to the model (the details on how this is done are presented in the following of the section). Thanks to this weighed graph it is possible to use a standard shortest path algorithm to find the alternative with the least estimated service provisioning time.

Referring to the formulations for the random variable R for a composition of any length (4.2) or a single component composition (4.1), the expected value of R is the sum of the expected values of its phases, so any weight on the graph must be mapped to a set of phases of a composition.

Remember that any edge $(s_i, n_j)(s_k, n_h)$ in the Composition Graph means that it is possible to compose services s_i (provided by node n_j), and s_k (provided by node n_h). For any such edge, its weight ω is the expected time between the end of the execution of service s_i on node n_j and the end of the execution of service s_k on node n_h, both for single service execution and sequential compositions. Given the definition of the phases described in Section 3.3, we can identify what are the values of these weights.

We can identify three types of edges that need different types of weights:

- Starting edges: The edges outgoing from the $start$ node, represented as $(start, n_j)(s_k, n_h)$, must be weighted with the estimated time to wait for the next contact with the provider, plus the time to transfer the input data for the service provisioning, the queue waiting time and the service component execution time, obtaining $\omega((start, n_j)(s_k, n_h)) = E[W + B + DQ_{n_h} + DS_{s_k,n_h}]$
- Ending edges: The edges incoming to the end node, represented as $(s_i, n_j)(end, n_h)$, are only weighted with the estimated time $E[\theta]$ to transfer the output of the service provisioning to the seeker. We have to consider that the formula is different for single service service provisions and service compositions, so we identify the case looking whether the type I_{s_i} (i.e., the input required by service component s_i) matches the output O_{start} of node $start$, which means that, actually, node n_j is the only provider. To underline the difference between the formulations of θ in these two cases, we rename as θC output data transfer phases that are part of a composition.

- Intermediate edges: These edges $(s_i, n_j)(s_k, n_h)$ are only between two providers, so they are part of a composition. Their weight, similarly to the starting edges, is the sum of the estimated time $E[\theta C]$ to transfer data between the providers, the estimated queue waiting time $E[DQ_{n_h}]$ on the second provider and the estimated service execution time $E[DS_{s_k,n_h}]$ for service s_k.

The resulting definition of the weights $\omega((s_i, n_j)(s_k, n_h))$ is thus as follows:

$$\begin{cases} E[W + B + DQ_{n_h} + DS_{s_k,n_h}] & \text{if } s_i = start. \\ E[\theta] & \text{if } s_k = end \wedge I_{s_i} = O_{start}. \\ E[\theta C] & \text{if } s_k = end \wedge I_{s_i} \neq O_{start}. \\ E[\theta C + DQ_{n_h} + DS_{s_k,n_h}] & otherwise. \end{cases}$$

4.4 Performance Evaluation

In this section we analyse the behaviour of our system in different simulated scenarios. We first validate the model presented in Section 3.3 by comparing the estimated service provisioning times with the values obtained in simulations. Specifically, in simulations seekers decide which service composition to use based on the estimates provided by the model for each possible alternative. For the selected composition, we compare the estimate of the model with the actual service provisioning time experienced in simulation. In the following we also use the PMTR and Haggle real mobility traces [24] [32] to simulate a real world scenario in order to also evaluate the performances of our system against other policies for the selection of providers.

Simulations are based on TheOne simulation environment, which is a de-facto standard for opportunistic networking [20]. For nodes mobility we used the RandomWayPoint model, modified as discussed in [25] in order to avoid problems related to the initial transient phase of the mobility process.[4].

In the validation simulations, the main parameters are described in Table 4.1. The total simulation time is 400000s of which the first 10000s are used to let nodes

[4] 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 [4]

Table 4.1: Default simulation parameters

Simulation runs	5
Number of nodes	30
Simulation space	$500m \times 500m$
Total simulation time	$400000s$
Mobility warm-up period	$10000s$
Statistics warm-up period	$360000s$
Request generation phase duration	$390000s$
Connectivity range	$90m$
Transmission speed	$2Mbps$
Density of each service	25%
Input type range	$i \in [0, 7]$
Requests output type range	$o \in [1, 8], o > i$

collect knowledge of their average contact and inter-contact times. After this warm-up period service requests are generated and sent to the system, statistics are collected only for service requests generated during the last 40000 seconds of simulation, to make sure that the system is in a stable state. As previously described, each service is identified by the type of its input/output which is represented by an integer. In our simulation an input type *i* is selected in the range [0,7], while output type *o* in the range [1,8], with the constraint that *i* must be less than *o* to avoid cyclic compositions. We tested the performance of the proposed approach analysing both the entire service provisioning time and the duration of the phases described in Section 3.3. We will not show the results for the queue waiting time and the service execution time, given the space constraints and also because the accuracy of the model relies for those phases on the standard formulas for the $M^{[X]}/G/1$ system.

We consider two scenarios for simulations: in the first one, requests are evaluated and solved using single service executions (we will call this scenario “Single”), while in the other one, the requests are solved by using at least two service components (scenario “Comp”). For both scenarios we have run two sets of simulations in which we vary the rate of request generation at seekers. In the “5-8” scenario, a new request is generated after a time interval in the range [5,8]s after the previous one, selected according to a uniform distribution. Each request is then assigned to a randomly chosen node, also following a uniform distribution, that acts as a seeker. The “20-40” scenario is similar to “5-8”, but the intervals between request generations are chosen between 20 and 40 seconds.

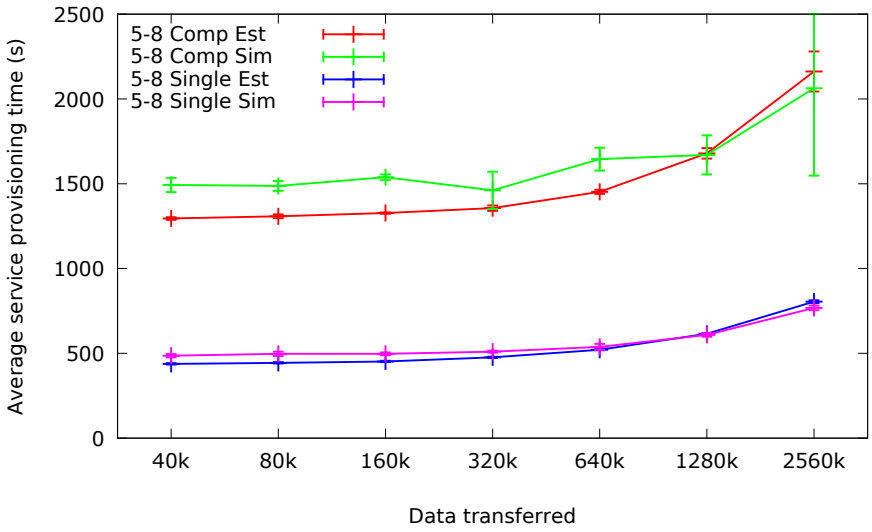


Figure 4.4: Service provisioning time, heavy load

Simulations are repeated increasing the amount of data that must be transferred between nodes as input and output parameters from 40 KBytes up to 2560 KBytes for the “5-8” scenario, while for the “20-40” scenario the data to be transmitted goes from 40 KBytes up to 5120 KBytes [5]. In each simulation, the input and output data sizes are the same for all services and requests. All the results shown are the average results of 5 independent simulation runs executed for each scenario, with 95% confidence intervals.

The first batch of results is about the average service provisioning time, evaluated changing the amount of data transferred between nodes, for both single service executions and longer compositions, with different rates of request generation.

In Fig. 4.4, we can see the average service provisioning time both for simulation (*5-8 Comp Sim* and *5-8 Single Sim*) and for the estimates provided by the model (*5-8 Comp Est* and *5-8 Single Est*).

The results show that for single services estimates are near or within the range of the confidence intervals of simulated service provisioning times. Results, when compositions are used, have a much larger variance, as expected, and are clearly

[5] This difference is due to the occurrences of overflowing in the nodes’ output buffers caused by the limitations of the simulator when high amount of data is transferred on a high load scenario.

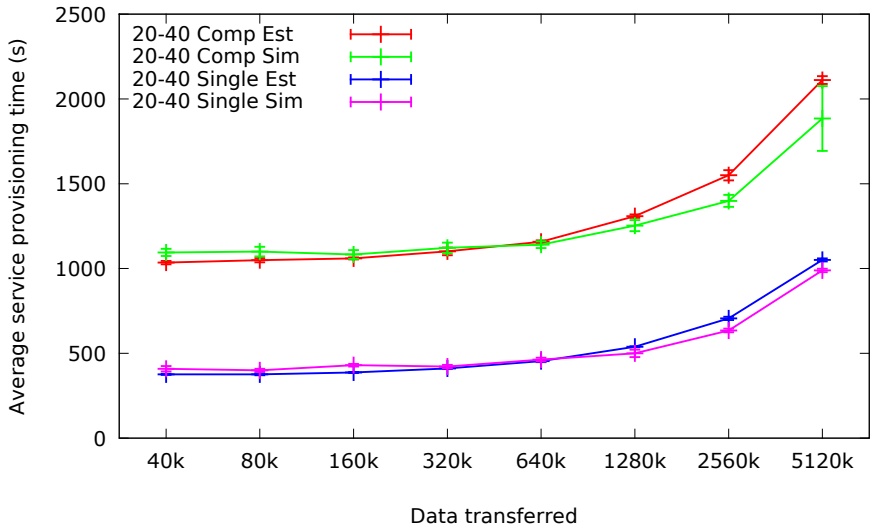


Figure 4.5: Service provisioning time, light load

more difficult to correctly model, due to the additional complexity brought by composition. Nevertheless, it is clear that the model estimates follow well the trend of the simulation results. The difference between the “Single” and “Comp” cases can be explained by realising that the amount of services executed in “Comp” is almost double than in “Single” and that we have at least one more data transfer between nodes for each request, so in the case of compositions, the network must handle a far heavier bandwidth usage and each provider must endure a higher load of requests.

In Fig. 4.5, we show the results for the "20-40" scenario. With a lower generation rate, the network is in general less loaded. The effect is particularly evident in the case of compositions. Average completion times are reduced by about 500s, and the maximum difference between estimated and real values is 13% for the case of compositions and 11% for the case of single service execution. Also in this case we observe a sub-linear increase of the service provisioning time with the size of the service parameters.

In Fig. 4.6 and 4.7 we show the results for the upload phases. We can see how in the two cases we obtain good estimates. We observe a higher overestimate between the model and the simulation results for the "Comp" case, in particular when the load increases (scenario "5-8") and for higher sizes of the parameters. The maximum difference between estimated and real values is around 25%. However,

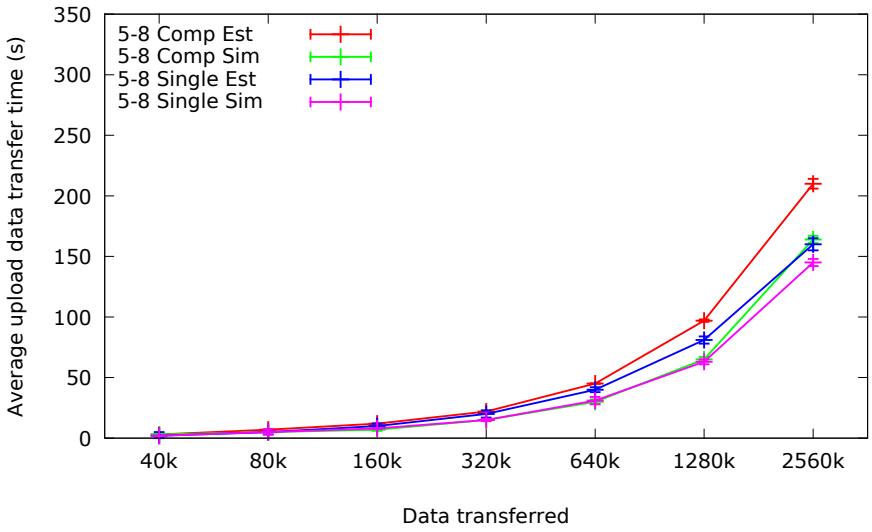


Figure 4.6: Average upload transfer time, heavy load

due to the relative short duration of this phase with respect to the total completion time, this overestimate does not dramatically impact on the overall estimates. In general, transfer times increase significantly with the size of the parameters, which is expected. Note that as the size increases, it becomes more and more likely that a single contact is not sufficient to complete the transfer. This results in a higher probability that also inter-contact times must be factored in the upload time.

In Fig. 4.8 and 4.9 we show the results for the download phases. The first aspect that can be noticed is that for every scenario our model gives and estimate that is very close to the average simulated times with a maximum error in the order of 10%. In this case, we do not observe a very steep increase of the curves for large parameter sizes (as we did in Fig. 4.6 and 4.7), while the download times are larger than upload times for small parameter sizes. This is because the download phase typically always includes at least an initial (residual) inter-contact time, because the seeker and provider are most of the time not in contact when service execution is over.

In general, the results show a good adherence between the estimates and the simulated times with a maximum error in the estimates of 10% of the simulated times for all the scenarios studied.

Based on the above results, we can conclude that the model is able to closely follow the real values of service completion time, and therefore can be reliably

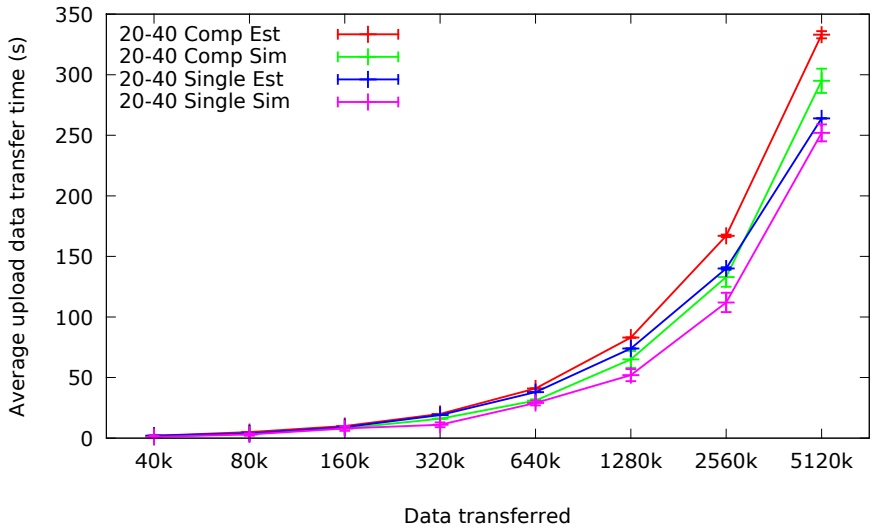


Figure 4.7: Average upload transfer time, light load

Table 4.2: Real traces simulation parameters

Parameters	PMTR	Haggle
Number of nodes	43	55
Total simulation time	$500000s$	$250000s$
Mobility warm-up period	$50000s$	$50000s$
Statistics warm-up period	$200000s$	$100000s$
Statistics collection period	$100000s$	$50000s$
Request generation phase duration	$450000s$	$200000s$
Density of each service	75%	50%
Average component execution time	$75s$	$75s$
Request inter-generation time	[40,80]s	[40,80]s

used to select which provider or composition of providers to use upon a service request. In addition, using the proposed approach we obtain a graceful degradation of service provisioning time as (i) the request load increases, and (ii) the size of the input/output parameters to be exchanged between nodes increases.

We then compared its performances of our system against other policies for service provider selection in scenarios where devices follow a real mobility trace. We used the results of the PMTR [24] and Haggle [32] experiments: real connection traces between mobile bluetooth enabled devices. For the PMTR traces we

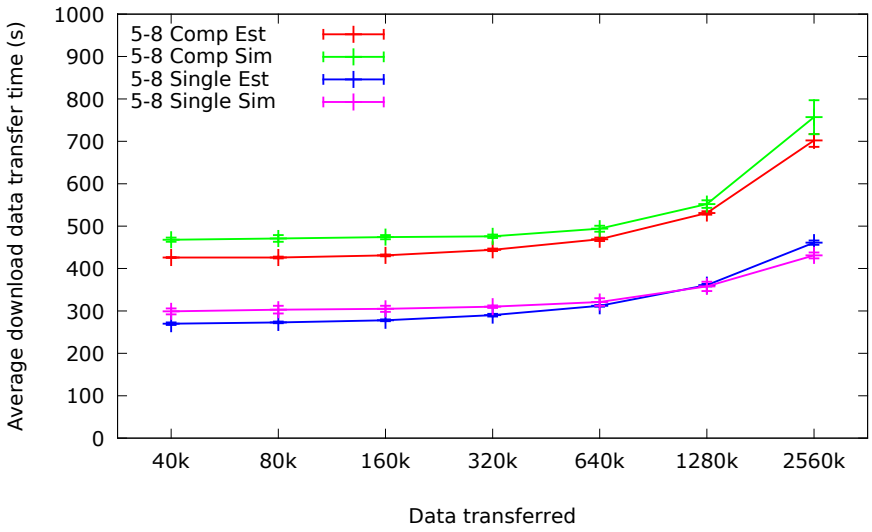


Figure 4.8: Average download transfer time, heavy load

cut the inactivity periods due to nights and weekends, while we left the Haggle traces as in the original experiments.

The parameters of the PMTR simulation scenario are listed in Table 4.2. The length of the simulation run has been extended to 500000s, the number of nodes with at least 10 contacts with other nodes in the PMTR traces is 43. Given that nodes become active at different points of the trace, the mobility warm-up period has been extended to 50000s after which the requests are generated. Statistics are collected for solved requests generated between the 200000s and 300000s marks. Nodes provider the same set of services as in the validation experiments, but the services are provided by 75% of the providers. Each service on each provider has an average execution time of 75 seconds. Service requests have a [40-80] inter-generation rate. For the Haggle traces, as in Table 4.2, simulations span for 250000s, while the analyzed requests are generated between the 100000 and 150000s marks. Services are provided by 50% of the 55 nodes.

We compared the average simulated service provisioning time of our own policy (that we call *MEV*, *Minimum expected value*) against the Random policy (called *RAN*) that selects a random path in the Composition Graph as the chosen composition of services and providers, the Atomic policy (*ATO*) that selects randomly a single component solution from the composition graph, and at last, in order to have a comparison with an approximation of the approach used by Serendipity

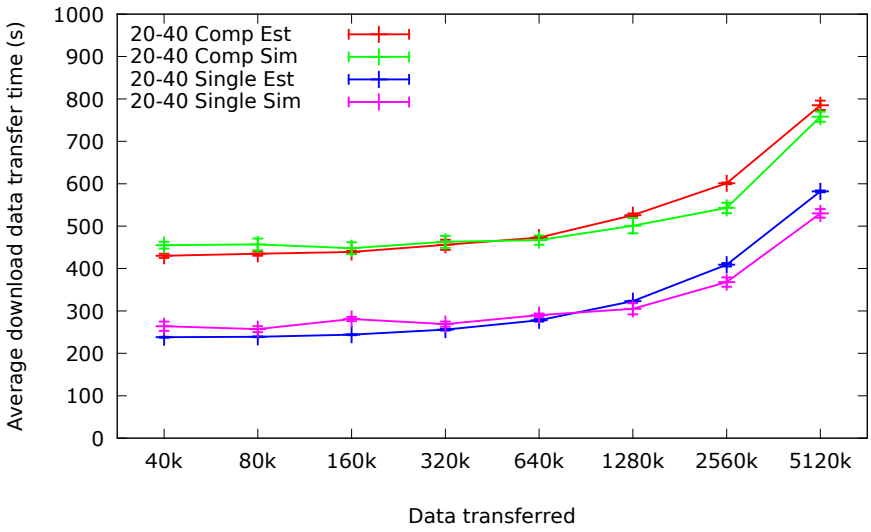


Figure 4.9: Average download transfer time, light load

[34] where there is an optimization on the presence of a connection between the parties involved in the resolution process, we used the Always First policy (AFIR) that randomly selects the first provider that can be used to further a composition process between the available ones that are connected with the process repeating at the end of each service execution phase. Differently to Serendipity, tasks cannot be divided into independent sub-tasks and the local resources cannot be used, but AFIR also does not transfer data back to the seeker at each composition step but permits direct communications between providers.

We executed 5 simulation runs for the 40KB and 1280KB data transfer scenarios. In Fig. 4.10 we can see the results of the simulations for PMTR traces, with MEV achieving 21% and 43% shorter average provisioning times (in the 40KB and 1280KB scenarios respectively) than the closest policy (AFIR). It can be noted that the average provisioning times decrease with bigger data transfers, that is due to a lesser completion rate for the generated requests for all the policies given that the low number of contact opportunities in the PMTR traces means that more nodes are unable to complete requested data transfer before the end of the simulation run. In Fig. 4.12 we can see the results for the Haggle traces, with the MEV policy having slightly better average service provisioning times when compared to AFIR (4% and 7% shorter), while other policies fare much worse. ATO and AFIR have higher completion rates given that in the Haggle scenario, contacts are sparse

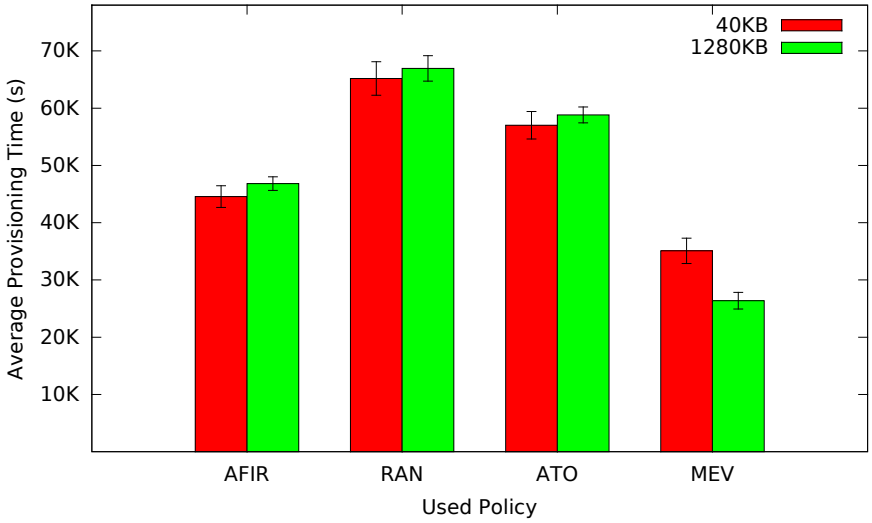


Figure 4.10: PMTR average provisioning time against other policies

and the examined window is earlier in the simulation, causing the system to have partial information about the network to build the statistics.

4.5 Summary

In this chapter we have presented an approach to select and compose service requests in an opportunistic environment. We have defined a mathematical model by which seekers can estimate the expected service provisioning time using different available compositions, and thus select the best one. The model is based only on local knowledge that nodes collect by exchanging a few information between each other during contacts. We have also presented performance evaluation results to (i) validate the accuracy of the estimates provided by the model, upon which selections are performed and (ii) characterise the behaviour of a system that adopts the proposed approach (iii). Simulation results show that the model is quite accurate. Specifically, the maximum error between the estimated service provisioning time and that observed in simulation is 13%. Moreover, results also show that using our approach seekers spread the load of service provisioning across the available providers, thus avoiding to saturate resources.

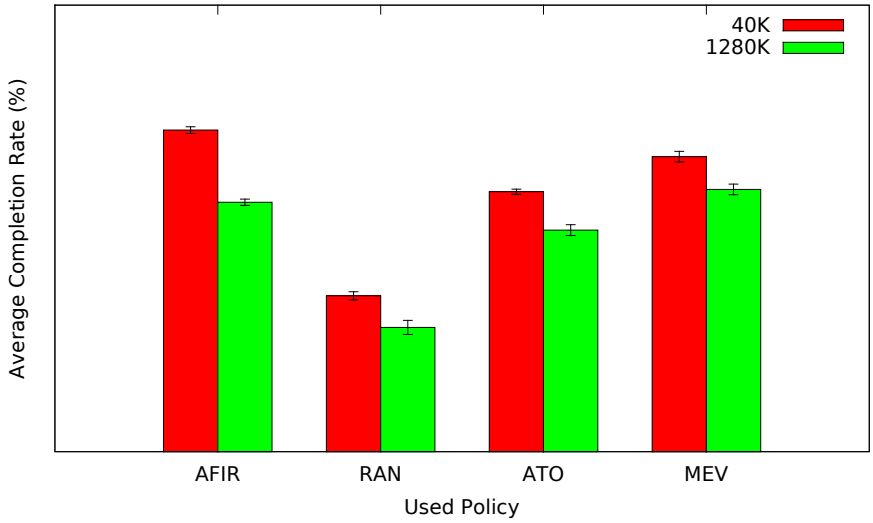


Figure 4.11: PMTR average completion rates against other policies

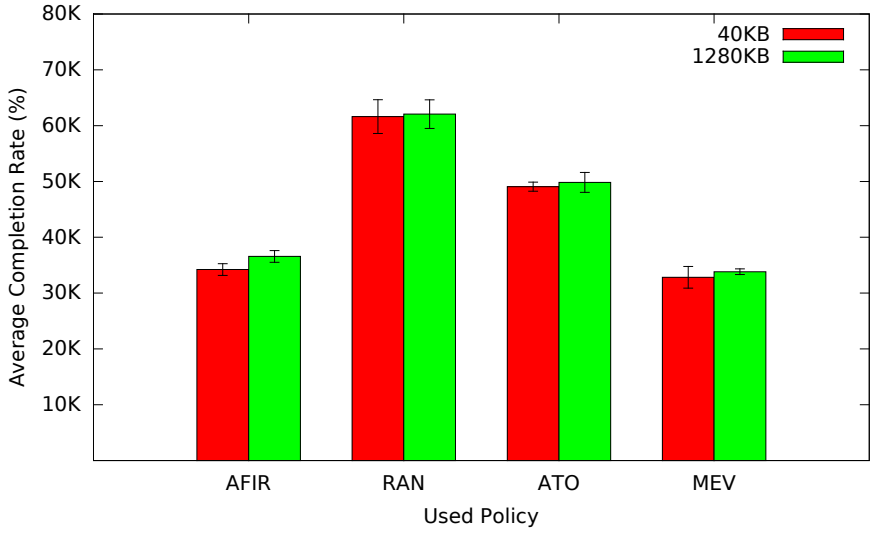


Figure 4.12: Haggle average provisioning time against other policies

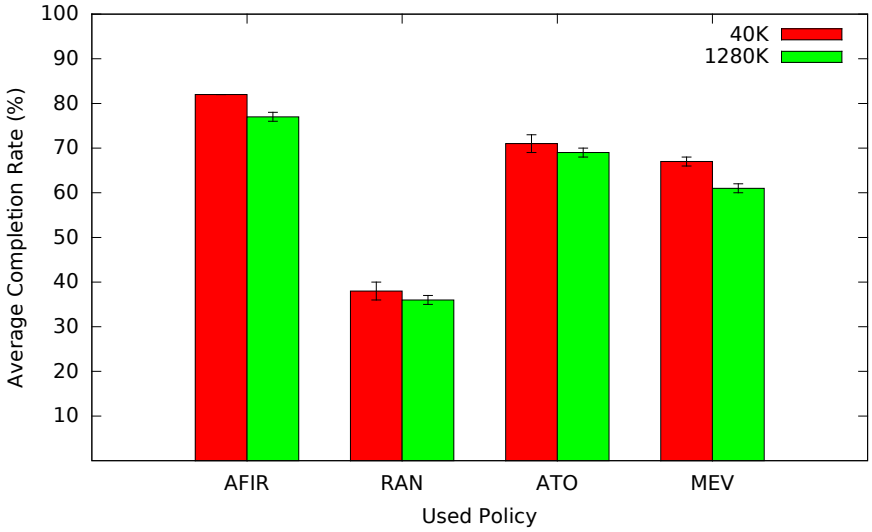


Figure 4.13: Haggle average completion rates against other policies

5

Offloading Service Provisioning on Mobile Devices in Mobile Cloud Computing Environments

5.1 Introduction

Mobile Edge computing is considered a very promising area in the cloud computing domain [12]. A popular approach to introducing mobile devices in the cloud computing paradigm consists in moving the execution of services from mobile users' devices to the cloud. This approach is motivated by the fact that executing services on the cloud, instead of locally on users' devices, saves mobile devices resources, and service execution times can be shortened thanks to the inherent scalability of cloud service provisioning platforms. The core assumption at the basis of this approach is that mobile devices are constantly connected to the Internet through an extremely high capacity wireless network, such that it is easy to move data back and forth between the mobile devices and the remote cloud platform. In this view, the capacity leap expected from 4G cellular networks (LTE-A) [6] is supposed to fully support this cloud computing paradigm.

Unfortunately, recent forecasts challenge the practical applicability of this approach. While 4G cellular networks will definitely provide much higher capacity compared to 3G, it is also expected that the data traffic generated by mobile users will increase much faster. For example, CISCO [6] foresees that mobile traffic demand will increase by at least ten times between 2014 and 2019, while cellular capacity will grow only by a factor of 1.4 in the same time frame. This challenges the possibility to support very frequent and possibly large data transfers required by this type of cloud computing solutions. In cases where the cellular network is congested, it would be too slow (or even impossible) to reach remote cloud services, thus making this approach technically unfeasible. In addition, this might also result in significant economic losses for cloud service providers, as it has been recently shown that there is a direct impact on the provider revenues of even small

additional delays (in the order of hundreds of milliseconds) in accessing remote cloud services [22]. Another possible scheme for mobile computing proposed in the literature is Mobile Edge Computing that traditionally consists in providing services directly at the edges of the infrastructure, i.e. on cellular base stations (eNodeB in the LTE terminology) [2]. This would not solve the aforementioned problem, as typically the bandwidth bottleneck would be in the cellular access network, and therefore even data transfer between mobile devices and eNodeBs might be problematic. To counteract the mismatch between mobile data traffic demand and cellular capacity, a promising mutation of this concept takes into account the introduction traffic offloading [28]. In one of the typical offloading scenarios, nodes receive data through direct device-to-device (D2D) communications with other mobile nodes, instead of through the cellular network. Opportunistic networking solutions are typically used [27], whereby mobile nodes exploit direct data transfer opportunities enabled by various wireless technologies (such as WiFi or Bluetooth in ad hoc mode) when they come close enough to be in each other's direct transmission range.

In this chapter, we exploit a conceptually similar approach to offload traffic related to service provisioning to mobile users. Specifically, we explore another concept for mobile edge computing, applicable when services can also be provided locally between mobile devices, by exchanging the related data between them during opportunistic contacts. Service provisioning between mobile devices through opportunistic contacts has been investigated in the literature as the opportunistic computing paradigm [23]. In opportunistic computing, mobile nodes form *mobile clouds at the edges of the global Internet infrastructure*, through which local service provisioning is supported. While exploiting opportunistic computing, our solutions goes one step beyond. In our solution, nodes requiring a service (hereafter referred to as *seekers*) evaluate whether it is more efficient to execute the service on a remote cloud, or on a locally available mobile node (not necessarily in contact with the seeker when the service request is generated). This approach is able to exploit both remote cloud platforms, when the cellular network is not congested, and local service provisioning, otherwise. As such, it takes the best of the conventional mobile edge computing approach and pure opportunistic computing paradigms. This approach is appealing also because of the resources already available on modern mobile personal devices. For example, high computational capability, ample storage and sensors, can be exposed to other users as services that can be accessed by other devices through direct contacts [12]. While it is clearly unreasonable to assume that any cloud service could also be provided locally, it is sensible to assume that a reduced set of services might be provided by

other mobile nodes in local proximity. Note that in some cases, this might indeed even preferable. For example, when services consist in elaboration of data locally available on mobile users, it might be more appropriate for privacy reasons that data stay on the device of their owners.

Together with the specification of the algorithms to realise this mobile edge computing approach, in this chapter we present simulation results showing that our solution is capable of offering better service provisioning time than a system where only the remote cloud is used. We also explore the effects on service provisioning performances of user service preferences and cellular network congestion. We show that the proposed system is able to autonomously adapt to the level of congestion of the cellular network, avoiding to contribute to its saturation, and still preserving low service provisioning times to the users, even in cases where the cellular network is highly congested or users have strong preferences on certain services.

This chapter is organized as follows. The structure and behaviour of the proposed system is presented in section 5.2. Performance evaluation results are presented and discussed in section 5.3. Finally, concluding remarks are reported in section 5.4.

5.2 Hybrid mobile edge computing solution for service provisioning

In this section we present the characteristics of our solution that enables the establishment of a local mobile cloud to support the execution of services available both on the cloud and on mobile devices in the area. The main components of the system can be seen in figure 5.1: the local mobile cloud, which is made up of mobile devices that can communicate with each other through wireless interfaces and that can request and provide services (pictured in the figure as S_1, S_2, S_3, S_4) to the other nodes; the eNodeB, which grants connectivity at the edge of the infrastructure to the local mobile cloud; the remote cloud, which hosts services the mobile nodes can access through the eNodeB.

At the high level, when a request for a service is generated at a mobile node (seeker), our algorithm decides whether this request should be served by the global cloud platform, or by some other mobile node nearby. We explain the details of the algorithm in the following subsections. Specifically we describe the system structure and behaviour by analysing the decision process involved in deciding how to solve a service request (subsection 5.2.1), the data that must be collected

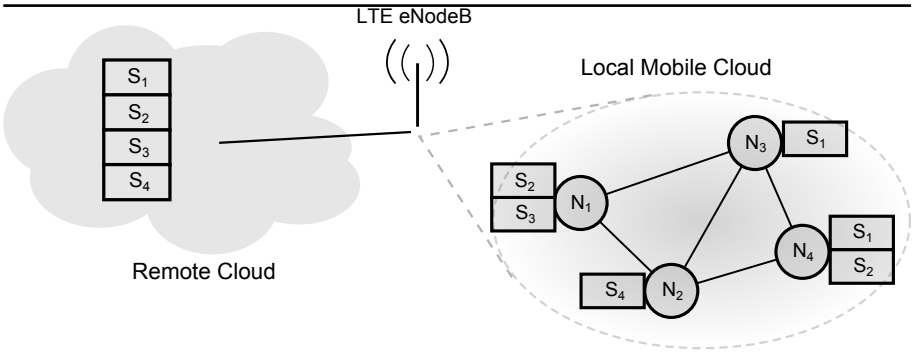


Figure 5.1: Actors of the systems

in order to take the decision (subsection 5.2.2) and the model used to determine how to resolve a request (subsection 5.2.3).

5.2.1 Resolution process

The resolution process is shown in figure 5.2 and starts with the *service request generation*, when a mobile node (*seeker*) runs an application that generates a request for a service. The seeker sends a message (*eNodeB inquiry*) to the eNodeB asking for information about the state of the LTE available data rates in upload and download, and an estimate of the time needed to execute the service on the remote cloud. [1] The eNodeB, at the reception of the message, observes the bandwidth occupation and sends this data as a response (*eNodeB response*) to the seeker, including the estimate on the service execution time on the remote cloud (*remote knowledge collection*).

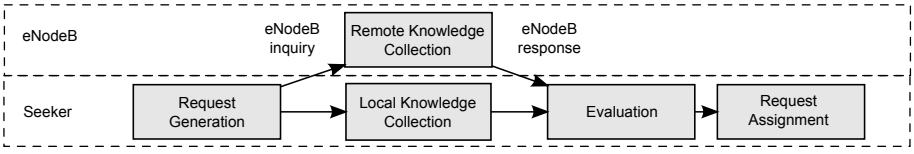


Figure 5.2: Request resolution process

At the reception of the response, the seeker estimates the total service provisioning time of the request using the remote cloud service. The seeker also uses

[1] Note that the size of this traffic is minimal, and therefore can be considered negligible from the cellular network congestion standpoint.

a local knowledge base containing previously collected data (*local knowledge collection*) on the other providers in the mobile cloud, like statistics on the mobility of the providers, the state of their computation queue and the offered services. The information in the local knowledge base is refreshed whenever two modes are in direct contact.

Thanks to the knowledge base, the seeker can evaluate the expected service provisioning time for all the known mobile providers that can be used to solve the request. These expected times are compared to the estimated service provisioning time of using the remote cloud service (*evaluation*).

If the seeker selects the remote cloud solution, it immediately starts sending the service request using the LTE infrastructure. Instead, if the selected provider is in the local mobile cloud and the seeker is currently not in contact with it, it waits the next contact with the selected provider in order to start sending the request. In this period of time further contacts between the seeker and other mobile providers may happen, triggering new information exchanges, a possible re-evaluation of the most suitable provider, and therefore a change in the service execution plan.

5.2.2 Data collection

The information, required to decide how to serve a request, consists in the upload and download data rates in using the LTE infrastructure and the average execution time of the service that is requested. This data is obtained by the seeker through the *eNodeB response message* that is created by the eNodeB. The eNodeB collects the average execution times of the services requested by the nodes and stores them in a database. It estimates the upload and download data rates based on the current traffic generated by mobile users in the cell.

As will be more clear from the following section, the information required about the other mobile provider is: (i) the average duration of contact and intercontact times with the seeker, (ii) the average data rate in the communications with the seeker, (iii) the service list of the provider, (iv) the provider queue statistics, like the average load, the average request arrival rate and the average service time, (v) the average queue of data to transfer from the provider to the seeker. This information is collected by each node by monitoring contacts with other nodes (for what concerns contact, intercontact times and average data rate), and by exchanging the other statistics during direct contacts.

5.2.3 Evaluation of service provisioning alternatives

The seeker uses two models to evaluate respectively the expected service provisioning time for each provider in the local mobile cloud that can solve the request and the expected service provisioning time using the remote cloud.

The first model is based on the model for opportunistic computing described in 4.2. For a given provider, this model gives a closed form expression for the expected value of the random variable representing the service provisioning time R_{mobile}, characterizing it as the sum of five successive periods of time that can be also formulated as random variables:

1. *Contact of the service provider* (W). The time needed by the seeker to encounter the provider after the point in time when the evaluation is performed. If the seeker is already in contact with the provider, the value is zero, otherwise it is the expected duration of the intercontact period.
2. *Data transfer* (Input time B, Output time θ). The time needed to transfer the input parameters from the seeker to the provider and the output parameters from the provider to the seeker (after the execution time is complete). These values include possible additional delays due to disconnection periods when the transfer is suspended as well as delays due to the presence of data from previous requests that need to be transferred to (or from) the same provider. The value for B is calculated as the time needed to transfer the data to the provider without disconnection, plus the expected duration of all the intercontact phases occurring before the end of the transfer. The expected value of θ is analogous to B, but it must consider the state of the connection seeker-provider at the end of the service execution: if θ starts during in intercontact period, it must consider an added delay to begin the transfer, if it starts during a contact it considers whether there could have been disconnections before the phases to estimate its residual duration.
3. *Queue waiting time* (DQ). Once onto the provider, actual execution may be delayed due to previous pending requests. To calculate the expected time of the phase, the model regards the provider as a $M^{[X]}/M/1$ queue and calculates the value using knowledge on the average load, service time and request arrival rate.
4. *Service execution time* (DS). The time to execute the service on the provider. It is calculated as the average previous executions on the provider of the requested service.

The formulation of the expected service provisioning time using a given mobile node becomes:

$$E[R_{mobile}] = E[W + B + DQ + DS + \theta]$$

For the remote cloud alternative, we can estimate of the service provisioning time t_{remote} using the information provided by the eNodeB in the eNodeB response and data locally available to the seeker. The service provisioning time can be estimated as the sum of the estimate of three delays: the time needed to upload data to the eNodeB t_{upl}, the time needed for the eNodeB to send data to the remote service provider and wait for the result of the computation t_{exec}, and the time needed for the seeker to download the output data of the service t_{down}. These estimates can be formulated as:

1. *LTE upload Time* t_{upl}. The time needed to transfer the service input data of size k_{input} and possibly queued data of size $k_{lte\ queue}$ from the seeker to the eNodeB, using the upload link that has a data rate of V_{upl}. k_{input} is a property of the service request generated and consequently known by the seeker. $k_{lte\ queue}$ is a value directly observable by the seeker at the moment of the evaluation. With these values, the total estimated LTE upload time can be formulated as:
$$t_{upl} = \frac{k_{input} + k_{lte\ queue}}{V_{upl}}$$
2. *Remote cloud latency and service execution time* t_{exec}. The time needed to transfer the input data from the eNodeB to the remote cloud provider, the time needed to execute the service, and the time needed to transfer the output data back to the eNodeB. Given that the amount of time spent transferring the data and executing the service is dependent on many factors that are out of the control of the system, like the actual provider location, the bandwidth available on the path to the provider, and the amount of resources dedicated to service executions, we can estimate t_{exec} using the average of previous actual values of the remote cloud latencies and service execution times for the same requested service.
3. *LTE download time* t_{down}. Similarly to the upload time, it represents the time needed to transfer the service output data, of size k_{output} which value is a property of the request, from the eNodeB back to the seeker, using the download link of data rate V_{down}, whose value is provided in the eNodeB response. t_{down} can be expressed as:
$$t_{down} = \frac{k_{output}}{V_{down}}$$

5.3 System evaluation

In this section we compare the performance of the hybrid solution explained in Section 5.2 with one that only uses a global cloud platform. We show a comparison of the average service provisioning times for both approaches in a range of scenarios that differ for amount of data that are transferred as service input and output for each request, and also for the amount of requests that are generated by the devices. We also detail the behaviour of the hybrid approach by analysing the fraction of requests that are solved using mobile providers in each scenario.

Table 5.1: Default simulation parameters

Simulation runs per scenario	10
Number of mobile nodes	30
Simulation space	$500m \times 500m$
Total simulation time	$400000s$
Mobility warm-up period	$10000s$
Statistics warm-up period	$10000s$
Request generation phase duration	$360000s$
Wi-fi connectivity range	$90m$
LTE download transmission speed	$300Mbps$
LTE upload transmission speed	$75Mbps$
Wi-fi transmission speed	$54Mbps$
Density of each service	25%
Number of different services	15
Average mobile service execution time	$10s$
Average remote cloud service execution time	$5s$

Simulation were developed using TheOne, which is a reference simulation environment for opportunistic networking and computing [20]. The basic simulation parameters used in this chapter are listed in Table 5.1. In these simulations, the mobile devices move following RandomWayPoint mobility traces as specified in [25]. We assume that mobility of nodes is confined within a single LTE cell, served by a unique eNodeB. Each simulation run lasts 400000s. The request is associated with a node (seeker) chosen with uniform probability. The requested service is picked among a set of 15 possible services either with uniform probability, or with a Zipf probability with parameters α equal to 1 or 3, to test the system under more or less skewed preferences towards specific services (the higher α, the more preference is skewed towards a small set of very popular services). We vary the

percentage of providers for each service between 10%, 25% and 50% of nodes, to test the performance with higher or lower "service capacity" of the mobile cloud. Simulated LTE data rates are 300Mbps for download and 75Mbps for upload based on current estimates of the maximum 4G capacity [10], opportunistic transfers are supposed to occur at the maximum capacity of 802.11g technology of 54Mbps.

We assume that a variable number of additional mobile devices generate traffic in the same LTE cell. The number of additional devices is generated according to a standard birth/death process. The total LTE capacity is shared between the active devices (i.e., the seekers and providers, plus these additional ones), such that the bandwidth available to the seekers and the providers changes over time based on the number of other active mobile devices in the cell. The number of additional nodes ranges in the intervals [0,20], [0,40], [0,80], [0,160], respectively, corresponding to fewer or more background nodes *possibly* competing for the LTE capacity. Transition rates to a new state are set for the previous ranges to 0.0025, 0.01, 0.04, 0.16 transitions per second both for birth events and death events. We replicate each simulated scenario 10 times. In all runs, the events related to the transition of the process defining the additional nodes activity are exactly the same. This guarantees that the congestion on the LTE network due to the additional nodes is the same when we vary the other simulation parameters.

The tests are repeated varying the rate of request generation by the system. In "10-15" scenarios a new request is generated after a time interval in the range [10,15]s after the previous one. This value is changed in the other scenarios to "15-20" and "20-30". For each of these values the tests are repeated changing the amount of data that has to be transferred as input and output of the services, from 40MB to 80MB and 160MB. In each simulation, the input and output data sizes are the same for all services and requests. All the results shown are the average results of the 10 independent simulation runs executed for each scenario, with 95% confidence intervals.

5.3.1 Service provisioning time comparison

Hereafter, we analyse the performances of hybrid service provisioning with respect to the previously described simulation parameters. Table 5.2 recaps the parameters values for the generated service requests used in the simulations (bold indicates default values).

Figure 5.3 compares the average service provisioning times for the hybrid approach and for the pure LTE approach. Specifically, points in the x axis are in the form `XXM YY-ZZ` where `XX` represents the size of input/output service parameters,

Table 5.2: Service provisioning parameters

Max. background nodes	20, **40**, 80, 160
Percentage of providers	10%, **25%**, 50%
Popularity of services	**uniform**, Zipf $\alpha = 1$, Zipf $\alpha = 3$

while YY and ZZ are the extremes of the time interval between service requests generations. Therefore, moving from left to right on the x axis we constantly increase the network load.

We can see that the average service provisioning times are faster for the hybrid approach in all scenarios, even when the request load is at its lowest (40MB 20-30), the hybrid approach achieves an average that is about 10% lower than the pure LTE approach. This difference in results grows as scenarios get heavier in load, with the 40MB 15-20 and 40MB 10-15 scenarios having differences respectively of about 18% and 32%.

This difference continues to grow as the traffic in the scenario grows, with the pure LTE approach that is unable to avoid saturation from the 80MB scenarios and is unable to complete the service requests for any seeker. The hybrid approach, instead is able to keep service provisioning consistent without overloading the infrastructure in all the analysed scenarios.

In Figures 5.5 and 5.6 we analyse the sensitivity with respect to the network load. Each curve in the plots corresponds to a different maximum number of background nodes competing for LTE bandwidth. Figure 5.5 shows the average service provisioning time, while Figure 5.6 shows the percentage of requests served by the mobile cloud.

It is interesting to note that service provisioning time increases gradually as the traffic load increases. Specifically, in Figure 5.5 we do not have signs of saturation such as an exponential increase of service provisioning time. From Figure 5.6, the main reason is the fact that the percentage of requests served by the mobile cloud increases as the cellular network becomes more and more congested. Interestingly, from Figure 5.5 we see that service provisioning time does not vary much for a varying number of background nodes, while (Figure 5.6) the fraction of service requests served in the local cloud does increase quite significantly. Therefore, as the LTE network becomes more and more congested, more services are provided locally, and this limits the effect of higher congestion on the cellular network. Finally, note that in case of 20 maximum background nodes and high load, service provisioning time is the highest (with respect to cases with greater numbers of

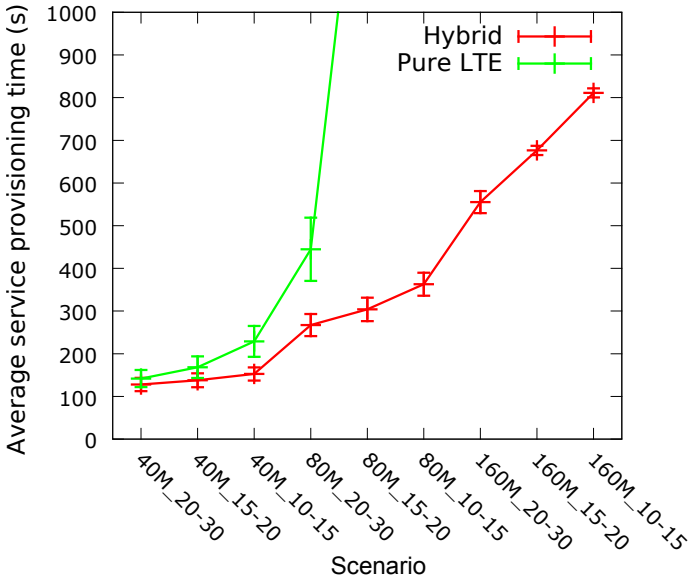


Figure 5.3: Service provisioning times, default service request parameters

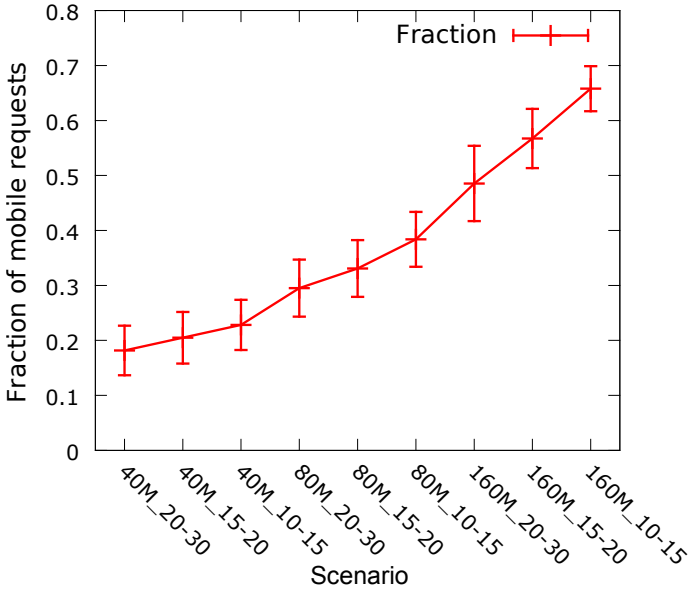


Figure 5.4: Fraction of requests

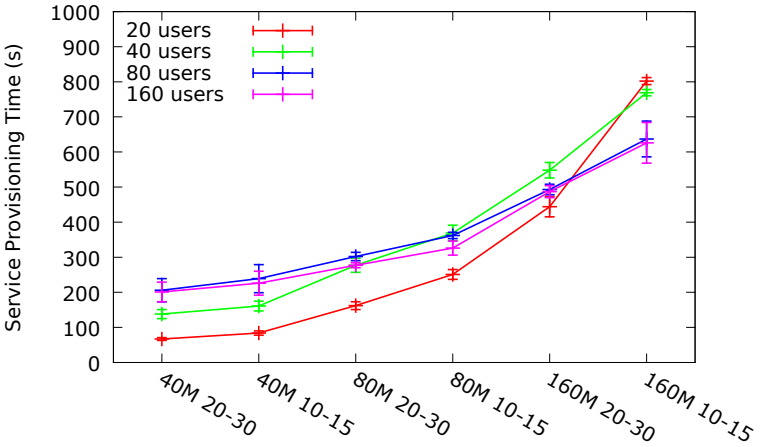


Figure 5.5: Service provisioning times - varying network load

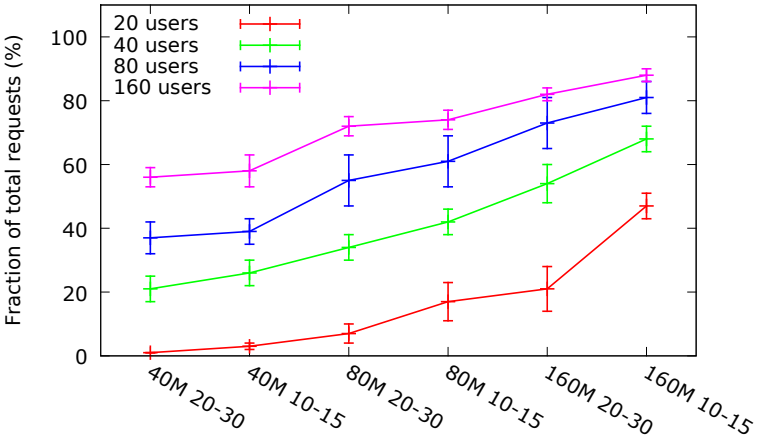


Figure 5.6: % of requests served by the mobile cloud - varying network load

background nodes), which is unexpected. From more detailed analysis of the various components of the delay, we found that this is due to the higher percentage of requests served by the global cloud in this case. This creates, at seekers, longer queues of messages to be sent to the eNodeB, and, in turn, this generates significant delay for seekers to get from the eNodeB the initial estimate of the service provisioning time using the global cloud. With a higher background load, fewer requests are selected to be served using the global cloud, which results in fewer

messages queued for transmission towards the eNodeB. Paradoxically, seekers receive the estimates for global service provisioning times quicker.

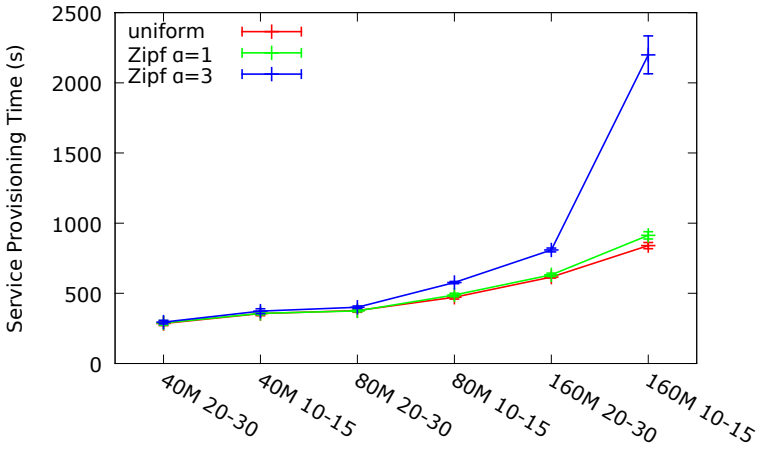


Figure 5.7: Service provisioning time - varying service popularity - 10% providers

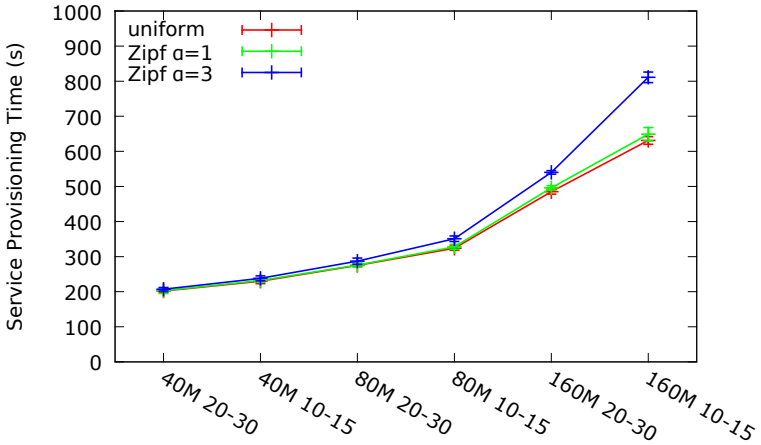


Figure 5.8: Service provisioning time - varying service popularity - 25% providers

Figures 5.7, 5.8 and 5.9 show the service provisioning time for varying skewness of service popularity and total service capacity in the mobile cloud. Again, it

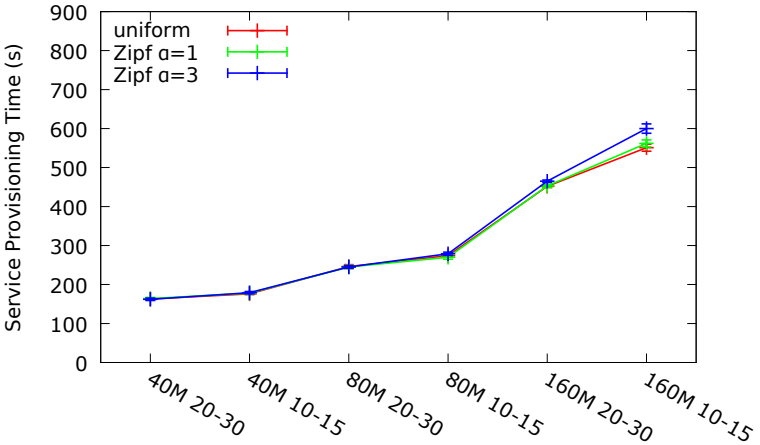


Figure 5.9: Service provisioning time - varying popularities - 50% providers

is confirmed that hybrid service provisioning provides graceful degradation of performance under significant loads. Specifically, signs of saturation appear only for a very unfortunate combination of parameters, i.e. when (i) service capacity is *very low* (10% of nodes are providers); (ii) network load is *very high*; and (iii) service requests are *very skewed* towards specific services (Zipf popularity with $\alpha = 3$).

5.3.2 Split of service executions in the hybrid approach

Figure 5.4 shows the fraction of requests served locally by mobile nodes in the different scenarios. We can see first of all that the shape of the graph resembles the one seen in figure 5.3, indicating a correlation. It is also notable that for the scenario with the lowest load the hybrid approach still assigns about 20% of requests to the local cloud. This indicates that local service provisioning might be useful even in cases when the LTE network is not particularly congested (this is the case, for example, when the seeker and provider are already in contact when the service request is generated, and the size of the input/output parameters is not that large). In the highest load scenario the ratio rises to an average of 65%, This indicates that our solution avoids cellular saturation, and is still able to exploit remote cloud execution when appropriate.

To further explore the behaviour of the system, we analysed the variation of the fraction of requests solved through the mobile cloud during specific simulation runs. To better understand this index, we plot it together with the fraction of additional nodes generated by the birth/death process (the fraction being computed

over the maximum number of nodes, i.e. 40). In Figure 5.10 we can see the results for run number 2 of the 10 total simulation runs for each scenario.

The graphs show a correlation between fraction of additional nodes generating traffic and the fraction of the requests assigned to mobile providers. Scenarios with the 40MB requests (blue lines), corresponding to a light transfer size due to service provisioning, have long periods of time where all requests are assigned to the remote cloud, until the added traffic is heavy enough. Instead the scenario with 160MB requests (red lines) rarely has periods with no requests assigned to mobile providers, and at the highest request generation rate (“10-15”) the ratio never goes below 20%. This last result indicate that the system consistently assign requests to mobile providers even during periods when the added traffic is negligible.

Based on the above results, we can conclude that the hybrid approach provides significant advantages in achieving better average service provisioning times. This is achieved also thanks to a dynamic detection of the status of the LTE network, that allows the proposed solution to correctly estimate whether remote or local service provisioning is more appropriate. This solution is thus able to avoid to saturate the LTE network, and to guarantee service provisioning also when the LTE network becomes congested.

5.4 Summary

In this chapter we presented a mobile edge computing solution that enables the creation of local mobile cloud networks to offload service provisioning from the remote cloud. We defined the behaviour of the system when a decision is to be taken whether a service should be provided from the remote platform or through some nearby mobile node, taking into account the state of the LTE network and of the surrounding devices. We presented sets of simulations to show the advantages in using this approach instead of relying exclusively on remote cloud services by showing that seekers experience better average service provisioning times and that the system is able to avoid congestion of the LTE network.

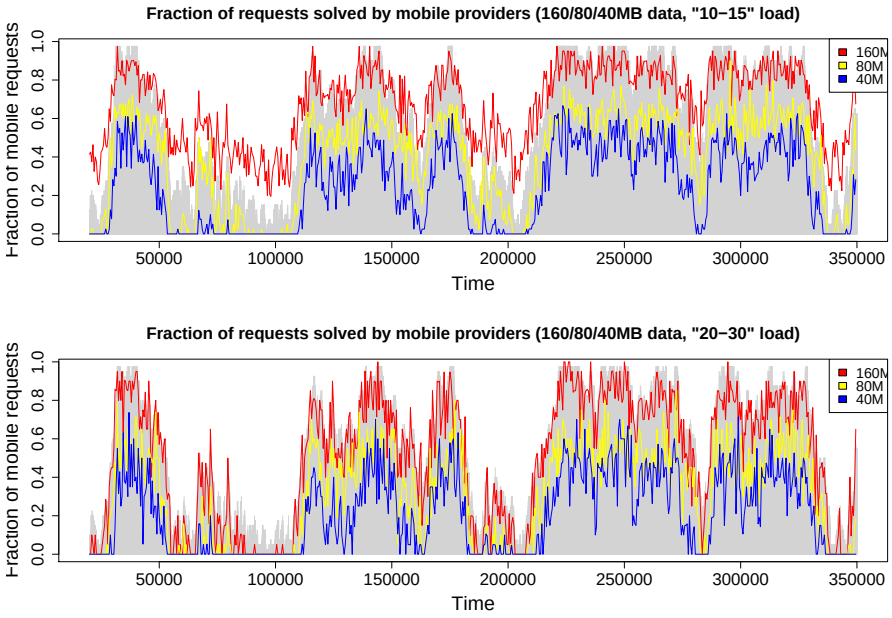


Figure 5.10: Share of requests

6

Conclusions

In this thesis we explored how the opportunistic computing paradigm can be used to design collaborative mobile systems for service provisioning in pervasive environments. We had multiple objectives that have been met and discussed: (i) the objective of designing a system that would enable mobile devices to autonomously share and access services, without assuming the presence of a network infrastructure. This system had also the objective to prioritize the performance for the users in the service provisioning process, without implementing solution that would create an overload on the mobile service providers that would have shared their resources. (ii) The objective of creating a stochastic model that could be able to precisely estimate the time needed to solve service requests in an Opportunistic system, by only using statistics elaborated from information directly exchanged between the devices and samples taken of the state of the network, the devices mobility and the providers capability to solve service requests. (iii) At last, the objective of creating a service provisioning system for environments where a cellular network infrastructure is present to enable for mobile devices access to services on a remote cloud, while the Opportunistic Computing paradigm can be used to improve the service provisioning performances and at the same time offloading data traffic from the infrastructure to the mobile devices in order to avoid congestion scenarios.

For objective (i), we have first identified a technique to collect context information on the devices network to find available services on other devices and compose them sequentially, if the case. These compositions are represented through a directed acyclic graph that is weighed using a stochastic model, producing estimates of the time needed to complete data transfers and computations. The weighed graph is used to find the alternative with the shortest estimated service

provisioning time. We then tested the system though simulations, managing to obtain better performances than other benchmark service allocation policies in scenarios differing for amount of data to be transferred as input/output of the services and generation rate of the service requests by the users. We show that this holds true also in all service composition cases, i.e. either when services are entirely available on individual devices, or when composition is the only possibility to provide the required function, or in mixed cases. Improvement of our algorithm with respect to the benchmark solutions can be as high as 50, 40, 75% in these three cases.

For objective (ii) we proposed a modelling solution for the stochastic estimation of disconnection events during the resolution process of single component services, by finding the relation between the phases of the service provisioning process and the evolution of the state of the connections between seekers and providers, managing to accurately determine the average performance offered by the opportunistic system. We extensively validated the resulting model by using synthetic and real mobility traces to show the advantages in using context information instead of using other benchmark policies. Results reveal that the model used to estimate service provisioning time is accurate, as the maximum estimation error is in the order of 15%. We then compare the service provisioning performances of the system against the other alternatives using real mobility traces. We show that it can achieve up to 43% shorter average service provisioning times with respect to the closest benchmark policy.

At last, for objective (iii) we designed a hybrid mobile edge solution for service provisioning, where mobile devices covered by a LTE cellular network infrastructure can autonomously form a mobile cloud of devices supporting the infrastructure in orchestrating computing services, all accessible through the infrastructure itself on a remote cloud, with also the added possibility of offloading them to other devices in the area, alleviating the load of requests from the cellular infrastructure. We showed in simulations that this type of system is capable of balancing provisioning of computation services between remote cloud and mobile cloud, even when the infrastructure is under heavy load, and rapidly redirect the flow of requests as network activity changes, keeping performances acceptable even when the network would become congested, and in the meantime obtaining a boost in performances for the users. As a demonstration, results show that the average service provisioning times are faster for the hybrid approach in all scenarios, even when the request load is at its lowest in the range of the simulation parameter, the hybrid approach achieves an average that is about 10% lower than the case when services are requested only from a remote service provider. For scenarios

with heavier load the difference grows up until the LTE-only solution is not able to handle it, saturating the LTE connection.

We reckon that these results are useful for showing how novel collaborative solutions can be practical and effective for service provisioning, with multiple positive effects: (i) creating service availability in the case remote services are not reachable, (ii) augmenting the service capacity and choice by composing resources on multiple mobile provider with positive effects in performance and stability, (iii) supporting existing cloud solutions avoiding infrastructure saturation scenarios, (iv) achieving better performances even accounting for extraneous network activity.

References

1. S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya. Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges. *Communications Surveys Tutorials, IEEE*, 16(1):337–368, First 2014.
2. S. Barbarossa, S. Sardellitti, and P. Di Lorenzo. Communicating while computing: Distributed mobile cloud computing over 5g heterogeneous networks. *Signal Processing Magazine, IEEE*, 31(6):45–55, Nov 2014.
3. D. Bianchini, V. De Antonellis, and M. Melchiori. An ontology-based architecture for service discovery and advice system. In *Database and Expert Systems Applications, 2005. Proceedings. Sixteenth International Workshop on*, pages 551–556, Aug 2005.
4. Chiara Boldrini and Andrea Passarella. Hcmm: Modelling spatial and temporal properties of human mobility driven by users' social relationships. *Comput. Commun.*, 33(9):1056–1074, June 2010.
5. Eleonora Borgia, Raffaele Bruno, Marco Conti, Davide Mascitti, and Andrea Passarella. Mobile edge clouds for information-centric iot services. Accepted for the The twenty-first IEEE Symposium on Computers and Communications, 27-30 June, 2016.
6. Cisco. Cisco visual networking index: Global mobile data traffic forecast update, 2014–2019, February 2015.
7. M. Conti, S. Giordano, M. May, and A. Passarella. From opportunistic networks to opportunistic computing. *Communications Magazine, IEEE*, 48(9):126–139, Sept 2010.
8. M. Conti, E. Marzini, D. Mascitti, A. Passarella, and L. Ricci. Service selection and composition in opportunistic networks. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, pages 1565–1572, July 2013.
9. Marco Conti, Davide Mascitti, and Andrea Passarella. *Euro-Par 2015: Parallel Processing Workshops: Euro-Par 2015 International Workshops, Vienna, Austria, August 24-25, 2015, Revised Selected Papers*, chapter Offloading Service Provisioning on Mobile Devices in Mobile Cloud Computing Environments, pages 299–310. Springer International Publishing, Cham, 2015.
10. Erik Dahlman, Stefan Parkvall, and Johan SkÃ¶ld. *4G LTE/LTE-Advanced for Mobile Broadband*. Academic Press, Oxford, 2011.
11. Lucia Del Prete and Licia Capra. Reliable discovery and selection of composite services in mobile environments. In *In Proc. of 12th IEEE EDOC*, 2008.

12. Niroshinie Fernando, Seng W. Loke, and Wenny Rahayu. Mobile cloud computing: A survey. *Future Generation Computer Systems*, 29(1):84 – 106, 2013. Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures.
13. Wei Gao, Qinghua Li, Bo Zhao, and Guohong Cao. Multicasting in delay tolerant networks: A social network perspective. In *Proceedings of the Tenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '09, pages 299–308, New York, NY, USA, 2009. ACM.
14. Christin Groba and Siobhán Clarke. Opportunistic service composition in dynamic ad hoc environments. *IEEE Transactions on Services Computing*, 99(PrePrints):1, 2014.
15. Sara Hachem, Animesh Pathak, and Valerie Issarny. Service-oriented middleware for large-scale mobile participatory sensing. *Pervasive and Mobile Computing*, 10, Part A(0):66 – 82, 2014. Selected Papers from the Eleventh Annual {IEEE} International Conference on Pervasive Computing and Communications (PerCom 2013).
16. Dijiang Huang, Xinwen Zhang, Myong Kang, and Jim Luo. Mobicloud: Building secure cloud framework for mobile computing and communication. In *Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on*, pages 27–34, June 2010.
17. Gonzalo Huerta-Canepa and Dongman Lee. A virtual cloud computing provider for mobile devices. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, MCS '10, pages 6:1–6:5, New York, NY, USA, 2010. ACM.
18. S. Kalasapur, M. Kumar, and B. Shirazi. Dynamic service composition in pervasive computing. *Parallel and Distributed Systems, IEEE Transactions on*, 18(7):907–918, July 2007.
19. John E. Kelly and Steve Hamm. *Smart Machines: IBM's Watson and the Era of Cognitive Computing*. Columbia Business School Publishing, 2013.
20. Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The one simulator for dtn protocol evaluation. In *Proceedings of the 2Nd International Conference on Simulation Tools and Techniques*, Simutools '09, pages 55:1–55:10, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
21. Karthik Kumar, Jibang Liu, Yung-Hsiang Lu, and Bharat Bhargava. A survey of computation offloading for mobile systems. *Mob. Netw. Appl.*, 18(1):129–140, February 2013.
22. Greg Linden. Marissa mayer at web 2.0.
23. Davide Mascitti, Marco Conti, Andrea Passarella, and Laura Ricci. Service provisioning through opportunistic computing in mobile clouds. *Procedia Computer Science*, 40(0):143 – 150, 2014. Fourth International Conference on Selected Topics in Mobile and Wireless Networking (MoWNet'2014).
24. Paolo Meroni, Sabrina Gaito, Elena Pagani, and Gian Paolo Rossi. CRAWDAD dataset unimi/pmtr (v. 2008-12-01). Downloaded from http://crawdad.org/unimi/pmtr/20081201, December 2008.
25. W. Navidi and T. Camp. Stationary distributions for the random waypoint mobility model. *Mobile Computing, IEEE Transactions on*, 3(1):99–108, Jan 2004.
26. A. Passarella, M. Kumar, M. Conti, and E. Borgia. Minimum-delay service provisioning in opportunistic networks. *Parallel and Distributed Systems, IEEE Transactions on*, 22(8):1267–1275, Aug 2011.

27. L. Pelusi, A. Passarella, and M. Conti. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *Communications Magazine, IEEE*, 44(11):134–141, November 2006.
28. F. Rebecchi, M. Dias de Amorim, V. Conan, A. Passarella, R. Bruno, and M. Conti. Data offloading techniques in cellular networks: A survey. *Communications Surveys Tutorials, IEEE*, 17(2):580–603, Secondquarter 2015.
29. U. Sadiq, M. Kumar, A. Passarella, and M. Conti. Service composition in opportunistic networks: A load and mobility aware solution. *Computers, IEEE Transactions on*, 64(8):2308–2322, Aug 2015.
30. Z. Sanaei, S. Abolfazli, A. Gani, and M. Shiraz. Sami: Service-based arbitrated multi-tier infrastructure for mobile cloud computing. In *Communications in China Workshops (ICCC), 2012 1st IEEE International Conference on*, pages 14–19, Aug 2012.
31. Mahadev Satyanarayanan, P. Bahl, R Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *Pervasive Computing, IEEE*, 8(4):14–23, Oct 2009.
32. James Scott, Richard Gass, Jon Crowcroft, Pan Hui, Christophe Diot, and Augustin Chaintreau. CRAWDAD dataset cambridge/haggle (v. 2009-05-29). Downloaded from http://crawdad.org/cambridge/haggle/20090529, May 2009.
33. Cong Shi, Mostafa H. Ammar, Ellen W. Zegura, and Mayur Naik. Computing in cirrus clouds: The challenge of intermittent connectivity. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 23–28, New York, NY, USA, 2012. ACM.
34. Cong Shi, Vasileios Lakafosis, Mostafa H. Ammar, and Ellen W. Zegura. Serendipity: Enabling remote computing among intermittently connected mobile devices. In *Proceedings of the Thirteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '12, pages 145–154, New York, NY, USA, 2012. ACM.
35. T. Soyata, R. Muraleedharan, C. Funai, Minseok Kwon, and W. Heinzelman. Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In *Computers and Communications (ISCC), 2012 IEEE Symposium on*, pages 000059–000066, July 2012.
36. Thrasyvoulos Spyropoulos, T. Turletti, and K. Obraczka. Routing in delay-tolerant networks comprising heterogeneous node populations. *Mobile Computing, IEEE Transactions on*, 8(8):1132–1147, Aug 2009.
37. Hideaki Takagi. *Vacation and Priority Systems, Part 1*. Elsevier Science Publishers B.V., 1991.
38. S.A. Tamhane, M. Kumar, A. Passarella, and M. Conti. Service composition in opportunistic networks. In *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, pages 285–292, Nov 2012.
39. P. Tournoux, J. Leguay, F. Benbadis, J. Whitbeck, V. Conan, and M. Dias de Amorim. Density-aware routing in highly dynamic dtns: The rollernet case. *Mobile Computing, IEEE Transactions on*, 10(12):1755–1768, Dec 2011.
40. Jianping Wang. Exploiting mobility prediction for dependable service composition in wireless mobile ad hoc networks. *IEEE Trans. Serv. Comput.*, 4(1):44–55, January 2011.