Multi-Strategy Dynamic Service Composition in Opportunistic Networks

Nicolas Le Sommer, Yves Mahéo, Fadhlallah Baklouti

Multi-Strategy Dynamic Service Composition in Opportunistic Networks

Nicolas Le Sommer, Yves Mahéo and Fadhlallah Baklouti,

IRISA, Université Bretagne Sud, France
{nicolas.le-sommer, yves.maheo, fadhlallah.baklouti}@univ-ubs.fr

Abstract

Distributed computing's pervasiveness is nowadays undeniable, and will continue to grow as the usage of device-to-device communications and the number of connected things populating our daily environment increase. Due to the connectivity disruptions induced by the mobility of devices communicating through short-range wireless interfaces and by the sleep phases of devices, it is often difficult to exploit the resources offered by the connected things forming these pervasive environments through the services they exposed, and even harder to compose these services together so as to provide users with more useful and sophisticated services. This paper presents a service composition system adapted to opportunistic networks. This composition system relies on a service provision platform that exploits opportunistic networking and computing techniques to cope with connectivity disruptions. Service composition is performed dynamically, according to users' interests. A multi-strategy scheme is used for the invocation of composite services, and a recovery mechanism is possible through partial invocation. This paper also presents the evaluation of the proposed composition system on two different scenarios: one involving people roaming in an open area, and another one involving spectators of a running event.

Keywords: Opportunistic Computing, Service Composition, Composite Service Invocation

1 Introduction

Nowadays, wireless networks are composed of a variety of heterogeneous devices that can communicate directly with one another, ranging from resource-constrained sensors to mobile devices and smart home appliances. After being neglected during many years, especially in cellular networks, device-to-device communication (D2D) is receiving more attention as direct communication between proximate devices can help to perform peer-to-peer data exchanges between user equipment, to offload data traffic in cellular networks, to support machine-to-machine communications needed by the Internet of Things, to efficiently provide local services... Progress in exploiting D2D data exchanges for enabling network-wide communication has been possible as techniques such as Opportunistic Networking or DTN emerged to cope with the connectivity disruptions and the additional delays introduced by device mobility, short network interface radio range, and frequent and unpredictable periods of sleep mode to save energy. Indeed, during the last decade, numerous disruption/delay tolerant and opportunistic networking techniques, implementing the "store, carry and forward" principle, have been proposed to address the issues posed by connectivity disruptions and by multi-hop communication [26]. These techniques allow to transfer messages between devices even if there is no end-to-end path between them.

Furthermore, with the objective of providing a higher-level programming paradigm, several service delivery protocols relying on these networking techniques have also been studied. These protocols allow a device to discover the services provided by other devices in the network, and to invoke the remote services it has selected among them so as to exploit the resources accessible

through these services. A global concern in these protocols is to cope with the fact that network-wide exchanges may face long delays or simply fail. Cross-layering, caching, or location awareness have been explored in order to reduce and optimize the network traffic [25, 21, 23].

In many scenarios and applications relying on opportunistic communications, the network is composed of a set of wireless resource-constrained devices that can collect raw measurement data from the environment (e.g., opportunistic sensing), and of portable mobile devices used by people to produce, to exchange and to process data. Due to the limited network bandwidth, and to the limited local resources (processor, memory, storage and battery), applications are generally distributed over a collection of devices in these networks. Each device typically provides basic functionalities for collecting, generating or processing data, and the end-user is provided with a composite service, formed by a combination of these basic functionalities. In practice, a sequential chain of services is employed. More sophisticated combinations, expressed with a process algebra, could be imagined, but are out of the scope of this paper. Even in its simplest form, the composition of services in opportunistic networks is a challenging task, studied in a limited number of works [28, 2, 7, 13] so far. Indeed, composition strategies dedicated to conventional Internet contexts are likely to cause prohibitive delays, and to suffer from transmission failures. In Internet, these transmissions are more or less considered always possible and efficient. In opportunistic networks, this assumption no longer holds. Service composition multiplies the constraints because fulfilling a composite service request involves the invocation of multiple services, any of these service being susceptible to be out of reach at any moment.

If there is no latitude in the choice of the actual providers of the services present in the composition, the performance of the invocation of a composite service will rely solely on the opportunistic networking layer, and any failure in reaching an elementary service provider will inevitably leads to the failure of the whole process. However, it is commonly assumed that a certain redundancy exists in service provision, that is, a same service may be offered by several providers in the network. In this context, there is much room for optimization when designing the composition execution process because choosing the right provider at one step of a composition may significantly reduce invocation time. Additionally, service redundancy allows in theory the execution of the composite service to continue even when an unsuccessful partial invocation occurs, as it should be possible to dynamically fall back on an alternative provider. In other words, there is a need for a composition strategy that takes into account the reachability of service providers and implements a form of recovery in composite service invocation.

In this paper, we present a composition system designed for opportunitistic networks. This composition system relies on a service provision system that supports the discovery, the selection, the invocation of services in opportunistic networks. We assume that each device in the network can play the role both of service provider and of service client and that a same service may be offered by several providers in the network. New composite services are automatically built on each device from discovered services, according to the user's interests. The key contribution of this paper consists in the definition and the implementation of two service composition strategies, based respectively on choreography and on orchestration, that take into account the reachability of service providers. These strategies can be parameterized with a utility function that makes it possible to estimate the delay and the success of a transmission between a service client and a service provider, and to select the providers according to these values. Two implementations of the utility function are considered and evaluated. The first one is based on location, and the second one is based on the reception times of the service advertisements emitted by service providers. In this paper, we also compare the success ratio and the execution time provided by the composition strategies and the utility functions in two different scenarios. The first scenario involves people moving around in an open area. The second scenario involves the spectators of a running race who move along predetermined paths.

The rest of this paper is organized as follows. Section 2 presents research works addressing service composition issues, especially in opportunistic networks. Section 3 presents the main features of the service composition system we have implemented in a service-oriented middleware platform. Section 4 details and formalizes the processes of discovery, selection and composition of the services. Section 5 presents the evaluation results we obtained for our system on the two

above-mentioned scenarios. Section 6 summarizes our contribution.

2 Related Work

Composition of services and invocation of composite services have been widely studied in the past, mainly considering Web services in stable Internet legacy networks [24, 12]. They also have been considered in networks composed of resource-constrained devices in the Internet of Things context [1, 10, 16, 4], as well as in more challenged networks, such as in dynamic pervasive environments formed by mobile ad hoc networks [17, 3, 32]. Most of solutions proposed for these kinds of networks do not tolerate the connectivity disruptions that can unpredictably occur between devices, and that can introduce additional delays and failures in the invocation processes. For instance in [3], Barakat et al. have proposed an adaptive service composition algorithm that triggers an adaptation process as soon as service changes occur in the environment, without interfering with the initial execution process, unless necessary. This adaptation process applies only a minimal number of modifications to the initial composition graph to avoid a service re-selection from scratch. The new composition graph resulting from this service re-selection is executed in a parallel process in order to maximize the chances of a successful recovery and to reduce the execution time of the composite service. This work does not describe the service selection criteria, and only consider the selection problem from a high level point of view, which does not make it possible to determine whether the proposed approach is suitable or not for dynamic network environments that are subject to frequent and unpredictable connectivity disruptions.

In [8], Capra et al. present a service composition framework for mobile environments that allows to improve the service composition reliability by considering the mobility pattern of devices and their colocation duration (the probability of service access is related to the colocation duration). The work described in [31] also tries to improve service composition dependability by characterizing the service providers' mobility and by tolerating uncertainty in this mobility. Like in [8], SeSCo (Seamless Service Composition) [19] exploits the nodes' colocation property to find, in the vicinity users, new providers offering the services that become unreachable following the users' movement. It also supports the execution of partial compositions and implements a simple recovery strategy. This colocation property is also considered in [9] through virtual communities of mobile users who are geographically close. By defining such communities, the authors try to minimize as much as possible the delays and failures arising during the delivery of composite services in mobile environments. These works only consider direct connections with service providers, and do not support communications relying on the "store, carry and forward" principle, which is the basic principle on which opportunistic and delay/disruption tolerant networking techniques rely on. Such a communication principle should, however, greatly help enhance the delivery of composite services by allowing to discover, to select and to invoke services that are not in the direct vicinity of users, but that can be reached through intermediate nodes even if there is not existing end-to-end links between service clients and providers. Furthermore, these works, except SeSCo, do not implement recovery strategies. Consequently if a service provider becomes unavailable, the partially executed composition remains in an inconsistent state.

The work presented in [18] is one of the few that addresses issues introduced by frequent wireless link failures on the provision of (composite) services. Jiang and al. investigate service composition and two levels of recovery: a network level and a service level. For the network-level recovery, the service providers involved in the provision of a composite services remain the same, and only the network path between them is subject to a recovery process. This process consists in discovering an alternative route to replace the broken link/path and restarting the data delivery. The service-level recovery process involves three operations: (1) finding alternative service providers, (2) finding a network path inter-connecting the new providers, (3) restoring the service states to recover the invocation process of the composite service. Service-level recovery thus takes much more time than network-level recovery. To provide an effective service composition and recovery solution, the authors additionally propose a heuristic to predict the service link lifetime based on node location and velocity using a linear regression.

By supporting, like [18], partial compositions and a recovery process, and by additionally integrating the execution phase into the composition phase, [13] tries to reduce the overall execution time of composite services in opportunistic networks. The advantage of this solution is that it addresses each selected service provider only once, and requires less data exchanges than a traditional consecutive approach that runs in two phases: a binding phase to select the providers, then a phase to invoke them. In [15], the authors propose an extension of the solution they described in [13] and [14]. This extension makes it possible to invoke several providers in parallel, thus creating several branches in the service composition graph. The branches can be split and merged, and the results of the composition process can be combined using semantic features. In [5], Clarke and al. present a goal-driven service composition model to dynamically compose services in mobile and pervasive computing environments. This model extends both the opportunistic resource allocation scheme and the service composition model proposed respectively in [15] and [6]. This new model 1) defines a heuristic to prevent the flooding of service requests in the network, 2) implements a dynamic adaptation and selection of execution paths, based on path reliability, in order to accomplish service execution with less failure, and finally 3) makes it possible to automatically merge new discovered services into existing service-flows. In [15], "directions" with the best quality value are chosen for the next-hop invocation. The quality value is calculated from the service execution times and an estimation of the remaining execution path's reliability. This estimation is derived from both the length of the remaining execution path and a value reflecting the network's dynamism. This value is computed from the node density, the transmission range and the nodes' average speed. In [28], Sadic et al. propose an approach close to that of Clarke and al. Indeed, in order to reduce as much as possible the composition delay, and thus to provide users with a certain quality of service, they propose to select the service providers to invoke following two metrics: the *shortest temporal distance,* which is the minimum time needed to send data from one node to another one, and the *service load,* which reflects the workload of a given service. Nevertheless unlike some of previous presented works, Sadic et al. do not consider partial invocations of composite services and recovery of invocation processes.

The above-mentioned works only consider —implicitly or not— a choreography-based invocation strategy of composite services. This strategy consists in repeatedly choosing a device for the first elementary service execution and delegating to this device the execution of the rest of the composition. An alternative strategy, called orchestration-based invocation strategy, also deserves to be studied in order to verify if it proves to be more suitable in some situations. In this strategy, the providers are selected and invoked by the very device that has initiated the invocation of a composite service. So, the composition is never transmitted to the service providers enrolled in the invocation process of a composite service, and invocations of next services are never delegated to them.

In the next two sections, we present the main features of a service composition system that supports the choreography- and the orchestration-based strategies, the partial invocation of composite services, the recovery of partial invocation processes and the creation of service compositions driven by the users' interests. This platform furthermore selects the providers that must be enrolled in an invocation process using a utility function. Two implementations of this function are currently provided by the platform: one is based on location and another one based on the reception times of the service advertisements emitted by service providers.

3 Design of a Dynamic Service Composition System for Opportunistic Networks

The design of service composition is strongly related to the way the basic functionalities of service provision (namely service discovery, selection and invocation) are offered, as well as the service model itself. Although this paper focuses on service composition, we will also briefly explain the characteristics of the other components of the service-oriented middleware platform we consider (depicted in figure 1). Thus, this section first presents the service model and the service discovery scheme implemented in the provision system we rely on, which has been developed upon the

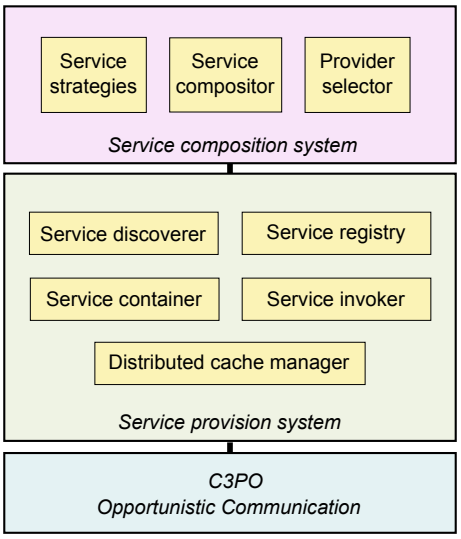

Fig. 1: Overall architecture of the service-oriented middleware platform

opportunistic communication middleware C3PO [22]. We then show how users' interests are used to create interest-driven composition graphs automatically. The selection of service providers and the execution/invocation of composite services are shortly introduced in this section and further detailed in the next one.

3.1 Service Model

In this work, we assume that services provided in the opportunistic networks are REST services, that it may exist several instances of a same service in the network, and that each service is described by a 5-uplet $S = \langle N, D, K, I, O, Q \rangle$, where N is the service name, D is a user comprehensible textual description of the service, K is a set of key words semantically characterizing the service, I is the set of input parameters, O is the set of output results and Q is a set of quality of service parameters. Each input parameter I_p of set I is defined as $I_P = \langle P, T_P, C_P, L_P \rangle$, where P is the name of the parameter, T_P is the type of parameter P, C_P are the pre-conditions applied on parameter P and L_p is a label providing semantic information about parameter P. Each output result O_R of set O is defined as $O_R = \langle T_R, C_R, L_R \rangle$, where T_R, L_R and C_R are respectively the type, the semantic label and the post-conditions of the output result. Finally, Q is a set of quality of service parameters. These parameters are notably used, as we will see later in this paper, to specify an estimation of the success ratio and of the invocation time of a composite service. This service description is compatible with the one that connected things could expose in a Web of Things (WoT) context, where WoT platforms dynamically instantiate REST services from the description exhibited by connected things [30], so that the resources offered by these things could be exploited by other devices. Like in the Thing Description [30], ontologies, such as SAREF (Smart Appliances REFerence) [11] or OM (Ontology of Units of Measure and Related Concepts) [27], can be used to semantically characterized the services and their parameters.

3.2 Service Discovery

In our service provision system, service discovery is performed following a peer-to-peer distributed discovery scheme. Each instance of this system maintains in is own service registry an up-to-date list of its local services and of the remote services it has discovered. A triplet $E = \langle H, T, R \rangle$ is associated to each service description in the service registry, where H is the C3PO identifier of the device providing the service, T is the date of the last update of the entry in the registry, and R is the result returned by the utility function implemented in the system to estimate the service

reachability considering two parameters: the success ratio and the invocation time of the service. To provide an efficient service discovery and to minimize as much as possible the network overhead related to the broadcast of service advertisements, the above-presented service descriptions have been piggybacked into the beacon messages that are exchanged by instances of C3PO to build their 2-hops neighborhood. When a beacon message is received from a given provider, the pieces of information related to this provider (i.e., the service descriptions S and the triplets E) are added in the local service registry, or updated if they already exist, in order to take into account the network topology changes. If a service registry entry has not been updated for a certain time, this entry is automatically removed by the service registry itself. The expiration period is a configuration parameter of the service registry. Cross-layering techniques are implemented in the C3PO communication middleware and in the service provision system so that the platform can compute, from message properties (emission date, sender location, etc.), information that can help select appropriated service providers such as for example the propagation time of messages and the distance between the providers and the local device.

It must be noticed that in opportunistic networks, which are intrinsically dynamic, a distributed peer-to-peer discovery scheme is more suited than a discovery scheme relying on a centralized registry that indexes and locates service providers in the network. Indeed, a centralized discovery scheme introduces additional delays in the discovery process because two messages (a service discovery request and the service advertisement returned in response) must be exchanged using opportunistic communication techniques to find a service.

3.3 Interest-driven Service Composition

Service compositions are modeled as directed acyclic graphs. A graph represents a chain of services that must be invoked successively, the result of the execution of a service being passed as parameter for the invocation of the next service. Compositions are triggered by the discovery of new elementary or composite services. When a new composition is created, the resulting composite service is added into the local service registry in order, on the one hand, to enable the discovery of this new service by the local applications and by the remote devices, and, on the other hand, to trigger the composition of another service with this composite service. All feasible compositions are not considered, but only those involving services that are compatible with the user's interests, defined as a set of keywords. The way these keywords can be created is out of the scope of this paper. They can for instance be derived from the description of the applications used by people.

The algorithm executed upon the discovery of a new (elementary or composite) service is presented in Algorithm 1. This algorithm makes it possible to compose services two by two, avoiding cycling compositions (line 3 : a service cannot be composed with a composite service that already integrates it). Compositions such as those illustrated in Figure 2 can be created using this algorithm (lines from 4 to 10 make it possible to obtain the composition 1, while lines from 12 to 18 produce the composition 2). Compositions are built independently of the service providers. The providers are selected when the composite services are executed/invoked. In this work, two strategies (orchestration and choreography) are considered to invoke the composite services. These strategies are detailed in section 4.

When a provider is considered as unreachable, the (composite and elementary) services it provides are removed from the local service registry. The local device will also removed from its registry the composite services it provides itself, and that rely on those offered by the missing provider, if it no longer has references on similar (elementary or composite) services offered by alternative providers. We recall that it can exist, in the networks we consider, several instances of a same service (i.e., a same service can be delivered by different devices).

3.4 Provider Selection

The selection of service providers is a key feature in a service composition system, especially in a system that relies on multi-hop communications. Indeed, it aims at choosing, among the providers that have been discovered, the provider that should be the fastest and the most reliable one to reply

Algorithm 1 Algorithm triggering service composition upon discovery of new services.

Data:

S_n=$\langle N_n, D_n, K_n, I_n, O_n, Q_n\rangle$: the new discovered service

R: the service registry

U: User'interests

S_c: The new service to compose

$\approx$: semantic compatibility test

φ: Utility function

1: **if** ($\exists\ i \in U\ /\ i \approx S_n.K_n$) **then**
2: **for all** $S_r \in R$ **do**
3: **if** ($S_n \notin S_r$) **then**
4: **if** ($\exists\ p \in S_r.I_r$ & $\exists\ o \in S_n.O_n$ / $(p.L_p \approx o.L_o)$ & $(p.T_p = o.T_o)$ & $(o.C_o \subseteq p.C_p)$) **then**
5: $S_c.N_c \leftarrow$ "($S_n.N_n$,$S_r.N_r$)"
6: $S_c.I_c \leftarrow S_n.I_n \cup S_r.I_r$ - $\{p'\}$ / $p' \in S_r.I_r$ & $\exists\ o' \in S_n.O_n$ / $(p'.L'_p \approx o'.L'_o)$ & $(p'.T'_p = o'.T'_o)$ & $(o'.C'_o \subseteq p'.C'_p)$
7: $S_c.O_c \leftarrow S_r.O_r \cup S_n.O_n$ - $\{o'\}$ / $o' \in S_n.O_n$ & $\nexists\ p' \in S_r.I_r$ / $(o'.L'_o \approx p'.L'_p)$ & $(o'.T'_o = p'.T'_p)$ & $(o'.C'_o \subseteq p'.C'_p)$
8: $S_c.D \leftarrow$ "Composite service formed by $S_n.N_n$ and $S_r.N_r$. $S_n.D_n$. $S_r.D_r$"
9: $S_c.K_c \leftarrow S_n.K_n \cup S_r.K_r$
10: $S_c.Q_c \leftarrow \{q_1,q_2,q_3, q_4\}$ / $q_1 \in S_n.Q_n$ & $q_1 \notin S_r.Q_r$, $q_2 \notin S_n.Q_n$ & $q_2 \in S_r.Q_r$, $q_3 = \min(q_3^{S_n}, q_3^{S_r})$ / $q_3^{S_n} \in S_n$ & $q_3^{S_r} \in S_r$, q_4=$\varphi(S_r)$
11: **else**
12: **if** ($\exists\ p \in S_n.I_n$ & $\exists\ o \in S_r.O_r$ / $(p.L_p \approx o.L_o)$ & $(p.T_p = o.T_o)$ & $(o.C_o \subseteq p.C_p)$) **then**
13: $S_c.N_c \leftarrow$ "($S_r.N_r$,$S_n.N_n$)"
14: $S_c.I_c \leftarrow S_r.I_r \cup S_n.I_n$ - $\{p'\}$ / $p' \in S_n.I_n$ & $\exists\ o' \in S_r.O_r$ / $(p'.L'_p \approx o'.L'_o)$ & $(p'.T'_p = o'.T'_o)$ & $(o'.C'_o \subseteq p'.C'_p)$
15: $S_c.O_c \leftarrow S_n.O_n \cup S_r.O_r$ - $\{o'\}$ / $o' \in S_r.O_r$ & $\nexists\ p' \in S_n.I_n$ / $(o'.L'_o \approx p'.L'_p)$ & $(o'.T'_o = p'.T'_p)$ & $(o'.C'_o \subseteq p'.C'_p)$
16: $S_c.D \leftarrow$ "Composite service formed by $S_r.N_r$ and $S_n.N_n$. $S_r.D_r$. $S_n.D_n$"
17: $S_c.K_c \leftarrow S_r.K_r \cup S_n.K_n$
18: $S_c.Q_c \leftarrow \{q_1,q_2,q_3, q_4\}$ / $q_1 \in S_n.Q_n$ & $q_1 \notin S_r.Q_r$, $q_2 \notin S_n.Q_n$ & $q_2 \in S_r.Q_r$, $q_3 = \min(q_3^{S_n}, q_3^{S_r})$ / $q_3^{S_n} \in S_n$ & $q_3^{S_r} \in S_r$, q_4=$\varphi(S_n)$
19: **end if**
20: **end if**
21: **end if**
22: **end for**
23: **end if**
24: $R \leftarrow R \cup \{S_n\}$
25: **if** ($S_c \neq \phi$) **then**
26: $R \leftarrow R \cup \{S_c\}$
27: **end if**

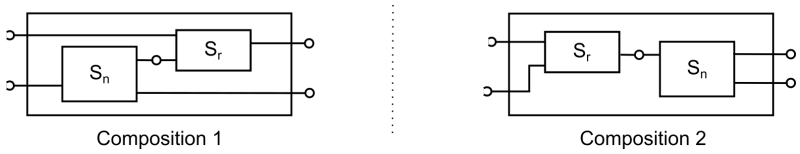

Fig. 2: Example of service compositions.

to a given request. In the type of networks we consider, the selection of the intermediate devices is a preponderant aspect regarding the execution time and the success of a service invocation and of a service composition. Many disruption-tolerant and opportunistic protocols have been proposed over the last decade [26], devising routing strategies based on different criteria. As shown in [21, 25], information about the reachability of devices should also be used at the service management level so as to select the most relevant providers. That's why, cross-layering techniques are implemented in the system. In our composition system, the service provider selection relies on a utility function. Two implementations of this function are considered. These implementations are detailed in section 4.

3.5 Invocation of Composite Services

Invoking a composite service mainly consists in successively calling the (elementary or the composite) services that form this one, following the direct acyclic graph representing the sequence of services and their interconnections. Such a graph, which is a part of a composite request (the invocation parameters completing this graph), can be processed following two approaches: the choreography and the orchestration approaches. The choreography-based invocation of a composite service consists in transmitting the composite request to the first selected service provider, and in delegating to this one both the execution of the first service and the selection of the provider of the next service. When the execution is completed, the provider of this service passes the results of the service execution and the composite request to the next provider. In the end, the last provider returns the results to the device that has initiated the composite request. In contrast, in the orchestration-based invocation of a composite service, the composite request is never transmitted to other service providers, and there is no delegation of service invocation: the providers are selected and invoked only by the very device that has initiated the invocation of the composite service. These two approaches, which can be complementary according to the network topology and to the distribution of services on the devices forming the network, are detailed in the Section 4 considering two parameters, namely the execution time and the success ratio of an invocation request of a composite service.

3.6 Partial Execution of Composite Services and Recovery

Executions of composite services are subject to unpredictable and frequent failures due to the mobility and the volatility of the devices forming the opportunistic networks. A failure is detected when a device responsible for the execution of a composite service has no longer reference to an elementary service that is a part of the composite service. The reference/description of this service may have indeed been removed from the local service registry because the device has no longer information about the device providing this service. This deletion can be done during the execution of the composite service. When the failure is detected, the composite service is removed from the local registry in order to no longer be advertised and considered as available. Moreover, the result of the partial execution of the composite service is returned to the device responsible for the execution of the parent composite service, or to the client that has invoked the "main" composite service. This propagation of the failure and of the partial results are done recursively until finding a device able to recover the execution or until reaching the service client, thus backtracking the execution of composite services one after the other. A recovery is triggered, and the propagation of the failures and of the intermediate results are stopped, when a device

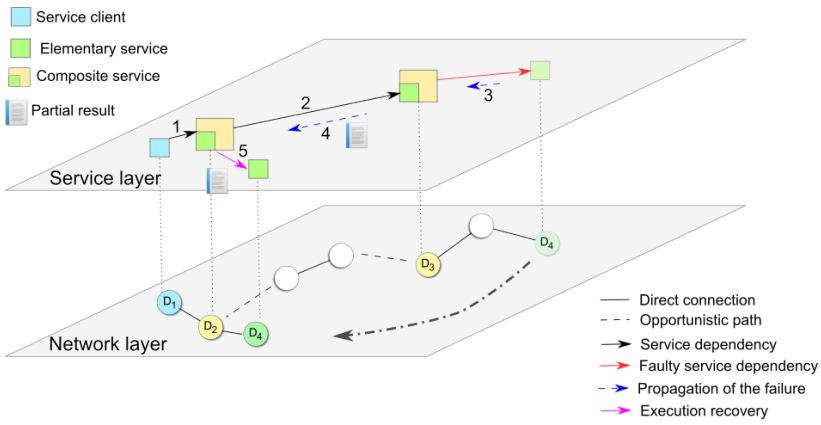

Fig. 3: Illustration of a recovery situation.

involved in the composition process can substitute a faulty service by another one. The partial results make it possible for a device to invoke the alternative service with the results that have been previously computed, thus avoiding to restart the invocation of a composite service from the beginning. Figure 3 illustrates a composite service invocation failure and recovery situation. In this figure, we consider a scenario in which device D_1 is invoking a composite service provided by device D_2. This composite service is composed of both an elementary service provided by D_2 itself and by a composite service provided by device D_3. The composite service provided by D_3 is composed of two elementary services: one provided by D_3 itself and another one provided by device D_4. In this scenario, we also suppose that D_4 has moved and D_3 has remained stationary, and that the description related to the service provided by D_4 has been removed from the registry of D_3 while this one was executing the service, thus causing a failure in the execution of the composite service. As shown in Figure 3, this failure is propagated to the parent composite service provided by D_2 with the partial execution result of the composite service provided by D_3 (i.e., with the result of the execution of the elementary service provided by D_3). In the scenario depicted in Figure 3, D_4 appears in the neighborhood of D_2 and the service provided by D_4 is added in the registry of D_2. Thanks to the partial result it has obtained from D_3, D_2 can recover the invocation by invoking the service provided by D_4. Doing so, it can return the result to the client service on D_1. It must be noticed that in this scenario we have considered that the recovery is done using the service provided by D_4, but another instance of the same service provided by another device could be used, as we assume a certain service redundancy in the opportunistic network.

4 Composite Service Execution: A Multi-Strategy Solution

The composition system we have designed dynamically selects the most efficient and reliable invocation strategy of composite services (i.e., the choreography-based or the orchestration-based strategy) according to two criteria of quality of service: the success ratio of a composite request, and its execution time. These two criteria are transposed to the selection of service providers that must be enrolled in an invocation process of a composite service. Providers are thus selected according to an estimation of the time needed to reach them, and to an estimation of the ratio of successful transmissions. These estimations are computed by a utility function. In the rest of this section, we present two implementations of this function, and how they are used.

4.1 Selection of providers

Service providers are selected by the utility function φ defined as follows:

$$\begin{cases} \varphi(S_i) = P_j, f(P_j) = max(f(P_l)), P_l \in R[S_i] \\ f(x) = \frac{\mu}{t(x)} + (1-\mu) \times s(x) \end{cases}$$

In this formula, P_j is the provider having the highest value calculated by the sub-function f among the providers that offer service S_i (identified by $R[S_i]$ in the above formula, where R stands for the local registry). Function f computes a value based on the estimation of the time needed to reach a provider and on the estimation to reach that provider successfully. These estimations are respectively noted $t(x)$ and $s(x)$ for provider x. Parameter μ makes it possible to promote one quality of service parameter to the detriment of the other. $t(x)$ and $s(x)$ are computed either from temporal or geographical information. Hereafter, we show how these values are calculated.

Moreover, if the computed value of f for a given provider is lower than a certain threshold, the entry, in the service registry that corresponds to this provider, is removed. Likewise, we only consider providers that are relevant.

Time-based implementation of the utility function The time needed to send a service request to a provider is estimated on the basis of the average of the transmission delays of the service advertisements broadcast by the provider. The success ratio of a transmission is the proportion of advertisements received by the local host among the advertisements emitted by a provider. To estimate this ratio, we assume that all devices send service advertisements with the same period of time (noted δ in the formulas hereafter). The time-based utility function we have defined operates on a sliding window of k values (i.e., we only consider the last k advertisements that are received by the local host). The estimated time t and the success ratio s for provider P are therefore formally defined by:

$$t(P) = \frac{\sum_{i=1}^{k}(\beta_i^P - \alpha_i^P)}{k}$$

$$s(P) = \frac{k}{1 + \lfloor \sum_{i=2}^{k}((\beta_i^P - \beta_{i-1}^P)/\delta) \rfloor} = \frac{k}{1 + \lfloor ((\beta_k^P - \beta_1^P)/\delta) \rfloor}$$

where α_i^p and β_i^p are respectively the emission time and the reception time of the i^{th} advertisement received from provider P. By computing an average of the transmission delays instead of considering only the last one, we promote, to the detriment of the providers that are met in fleeting way, those that are reached the most frequently with a minimum of delay either directly or via intermediate devices.

Location-based implementation of the utility function When sending an advertisement, the composition system includes in this advertisement the location of the local host. Doing so, the devices receiving this advertisement can compare their own location with that of the provider. Similarly to the previous utility function, this function operates on the last k advertisements received from a given provider. This function computes an estimation of the transmission delay considering the average of the distances between a provider and the local host. By computing the average of the distances between the local host and a provider instead of considering only the last advertisement of that provider, we promote the providers that are the closest to the local host during a given period of time instead of those that are briefly close. The estimation of the time needed to reach service provider P is equal to the reception delay of advertisement i, such that the distance d_i^P between the local host and the provider is the closest to the average of the distances traveled by the k advertisements received by the local host:

$$t(P) = (\beta_i^P - \alpha_i^P), \quad d_i^P \approx \frac{\sum_{l=1}^{k} d_l}{k}, i \in [1, k]$$

To estimate the success ratio of a transmission between a client and a provider, we consider the k last advertisements received by a (local) client from a provider, and we compute the average of the distances between them based on their respective location at emission time and at reception time. The probability of reaching a provider usually decreases when the distance between the local device and the provider increases. That is why we have defined the success estimation function as a multiplicative inverse function, shifted to the left by -1, which takes d^P as a parameter. This function returns a value close to 1 when d^P is small and a value close to 0 when d^P is big. It is defined as follow:

$$s(p) = \frac{1}{1+d^p}, \quad d^P = \frac{\sum_{j=1}^{k} d_j^p}{k}$$

4.2 Invocation of Composite Services

Hereafter, we present both the estimations of the execution time and of the success ratio of an invocation of a composite service for the two strategies implemented in our composition system. For that, let us consider a composite request C of n services identified respectively by $S_i, i \in [1, n]$, a set of m providers that offer one or several services. Let us also consider that composite request C is emitted by a requester Λ.

4.2.1 Estimation of the execution time

The invocation/execution time of a composite service C is defined for the orchestration-based strategy by:

$$\tau(C) = \sum_{i=1}^{n} 2 * t_\Lambda(P_l), \quad P_l = \varphi_\Lambda(S_i)$$

where P_l is the provider of the service S_i that has been selected by the utility function φ. In the orchestration-based strategy, the response must be returned to the device Λ that has initiated the composite request. Thus, the time $t_\Lambda(P_l)$ is multiplied by 2 to consider this round trip.

Concerning the choreography-based strategy, the estimation of the execution time of a composite request is defined by:

$$\begin{cases} \tau(C) = t_\Lambda(P_1) + \sum_{i=1}^{n-1} t_{P_i}(P_{i+1}) + t_{P_n}(\Lambda) \\ P_1 = \varphi_\Lambda(S_1) \\ P_{i+1} = \varphi_{P_i}(S_{i+1}) \end{cases}$$

where $t_\Lambda(P_1)$ is the estimation of the time needed to send the composite request from the requester Λ to the first provider, $t_{P_n}(\Lambda)$ is the estimation of the time needed to send the result of the composite request from the provider P_n of the last service S_n to the composite requester Λ. The rest of the formula $\tau(C)$ is the sum of the estimations of intermediate execution times, where $t_{P_i}(P_{i+1})$ is the estimation of the time needed to send the composite request from the provider P_i to the next provider P_{i+1}.

It must be noticed that these estimations are in practice computed recursively and set as a quality of service parameter in the description of each composite service (the parameter is stored in the set of parameters Q, see section 3.1).

4.2.2 Estimation of the success ratio

As mentioned before, in the orchestration-based strategy, the response of each service invocation is returned to the requester Λ. Thus, the estimation of the success of an orchestration-based invocation of a composite request is defined as the product of the square of the transmission success estimation of a request from the requester Λ to the providers enrolled in the invocation process. This estimation is defined by:

$$\gamma(C) = \prod_{i=1}^{n} s_\Lambda(P_l)^2, \quad P_l = \varphi_\Lambda(S_i)$$

Regarding the choreography-based strategy, the estimation of the success of an execution is defined as the product of the transmission success estimation of a request from the requester Λ to the first provider (P_1) with the transmission success estimation of the execution result from the last provider (P_n) to the requester Λ, and with the intermediate transmission success estimations.

$$\begin{cases} \gamma(C) = s_\Lambda(P_1) \times s_{P_n}(\Lambda) \times \prod_{i=1}^{n-1} s_{P_i}(P_{i+1}) \\ P_1 = \varphi_\Lambda(S_1) \\ P_n = \varphi_{P_{n-1}}(S_n) \\ P_{i+1} = \varphi_{P_i}(S_{i+1}) \end{cases}$$

5 Evaluation of the composition system

5.1 Rationale

The entire service-oriented platform described in this paper has been implemented on smartphones running Android and allowed us to run a few field experiments, at a small scale. These experiments give us confidence on the actual feasibility of our approach but cannot be easily reproduced nor parameterized in order to properly evaluate the service composition system we propose.

Our main objective is to evaluate the two implementations of the utility function that is at the basis of the provider selections, and the advantage of having a system capable of dynamically choosing the most suitable strategy (choreography or orchestration) according to the network topology. Evaluating this is not an easy task as the composition system is obviously dependent on the other components of the service-oriented middleware platform.

Our approach for this evaluation is based on emulation: the largest part of real code that implement our system is used and the rest of the system is simulated. We use the LEPTON emulator [29] for conducting the evaluation (see [29] for a detailed description of the advantages of the emulation approach in opportunistic computing). This allows us to run the same code as the one we deployed on Android mobile devices for almost all the service middleware platform (the C3PO opportunistic networking layer as well as the service provision layer and the service composition layer), while remained simulated the mobility of nodes, the D2D radio communication, and the application behavior.

We tried to be reasonably realistic when applying simulation. Although there are multiple situations in which our service platform could be used, we focused our evaluation on handheld devices powerful enough to run the platform (typically smartphones or nano computers), namely for practical reasons, and because there are generic enough to encompass a wide range of applications in which opportunistic networking is a necessity and the volume of data exchanged could be high (of the order of mega-bytes per transmission). Admittedly, some interesting networks and applications, such as networks of low-power devices communicating with Bluetooth LE or LoRa for monitoring purposes are de facto excluded, but they often involve few mobility and consequently comes less down to opportunistic computing. As far as the mobility of nodes is concerned, we considered a synthetic mobility model as well as traces reflecting a real scenario. Several applicative behaviors were chosen, associated with some randomization, with a number of services that is, in our opinion, significant at least on a medium scale. As for D2D communication, we didn't chose to be very abstract, as it is often done for example with many evaluations with The One simulator [20], but rather strive to simulate as closely as possible a D2D communication that proved to be effective in the field. As Wifi-Direct and Bluetooth are the two available technologies in off-the-shelf mobile devices, we have carried out preliminary experiments to compare them and select the one we would simulate. The results we have obtained are given in table 1. Connection

	Wifi-Direct	**Bluetooth**
Discovering time (average/median)	4.1 s / 4.5 s	5.5 s / 2.9 s
Connection time (average/median)	15.6 s / 13.2 s	5.7 s / 2.9 s
Max throughput	54 Mb/s	2 Mbit/s
Max radio range (in open area)	180 m	70 m
Radio range average (in presence of obstacles: buildings, cars, ...)	52 m	15 m

Tab. 1: Measured performance of Wifi-Direct and Bluetooth using Motorola G4 mobile phones.

Parameter	**Value(s)**
Open area size	500 m × 500 m
Interval between composite service requests	between 2 and 5 min.
Evaluation duration	1 hour
Number of nodes (N)	200
Radio range	80 m
Service advertisement period	10 seconds
Speed range	between 0.5 and 2m/s

Tab. 2: Parameters common to all evaluations.

time with Wi-Fi Direct is higher but this is compensated by the fact that the radio range with Wi-Fi Direct is significantly greater than with Bluetooth (at a speed of 2m/s, a mobile node travels 20 m during the 10 s separating the difference of connection times, which does not reach the 110 m that separates the two radio ranges). So, and a fortiori if we consider throughput, Wifi-Direct is a better choice (as far far as energy consumption is not a prior concern). So we have developed a module in LEPTON that simulates communication according to the distance between devices, and that respects the behavior of Wi-Fi direct communication entities, taking into account latencies in group formations. Indeed, Wi-Fi Direct organizes devices in communication groups. Each group is managed by a group owner (GO) that plays the role of a soft access point. A given device can only communicate with its group members and two GOs cannot be connected together. Thus, in order to exchange data with a device of another group, a device must leave its current group and join the group of the other device.

5.2 Evaluation setup

Two different scenarios involving 200 people have been considered. The people either move around according to the Levy walk mobility model in an open area of $500 \times 500\,m$, or attend a running event in the city of Vannes in France, moving in this case along predetermined paths.

The applicative behavior is the following. A fixed number of distinct services G is considered in an experiment. On each device are deployed L instances of services chosen randomly among the G ones. As $N = 200$ devices participate in the experiment, a service is provided redundantly by $L \times N/G$ instances in average. All along the experiment, compositions are automatically computed according to the discovery of services, considering that the user is interested in all the G services. Repeatedly, a composition that comprises C services is chosen randomly among the current computed ones, and the device launches the corresponding composite service request.

The evaluation is based on two metrics: the composition success ratio (that is, the ratio of composite service executions that have successfully ended) and the composition time (that is, the time between the launching of the composite service execution and the reception of the result of the last service in the composition). Each experiment has been repeated at least 10 times with the parameters specified in Table 2. The performances of the utility functions and of the invocation strategies are evaluated by varying the maximum number of hops between the service client and service provider (evaluation E_1 in Table 3), the number of services per experiment G (evaluation E_2 in Table 3) and the number of services per composition C (evaluation E_3 in Table 3).

Parameter	Value(s)		
	E_1	E_2	E_3
Maximum number of hops	1, 2, 3	2	2
Number of local service instances (L)	5	5	5
Number of services per request (C)	4	4	3,4,5,6
Number of services per experiment (G)	20	10, 15, 20, 25, 30	20

Tab. 3: Parameters varying according to the evaluations.

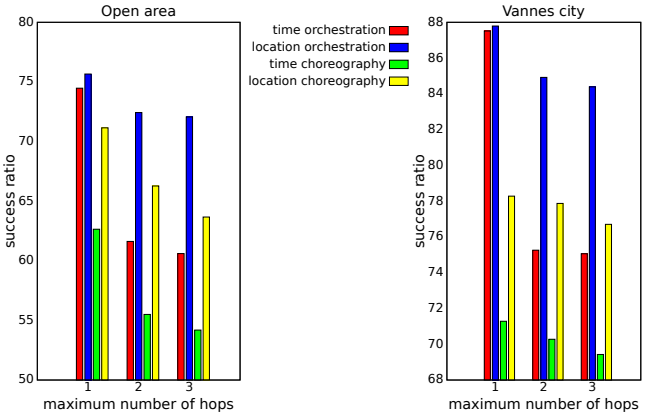

Fig. 4: Success ratio against the maximum number of hops between service clients and service providers.

5.3 Success ratio of composite request execution

Figure 4 shows the success ratio against the maximum number of hops between the service clients and the service providers for both utility functions and both invocation strategies. When the maximum number of hops increases, the success ratio decreases regardless of the strategy and the utility function. This is the result of the additional disruptions occurring between intermediate devices when the number of hops increases. We can observe that the executions relying on the location-based utility function (LUF) are less affected than the executions performed with the time-based utility function (TUF). This can be explained by the fact that closer providers are selected and enrolled in the invocation process. Indeed, disruptions between closer devices occur more rarely than between devices that are at the limit of their radio range. Moreover, closer mobile devices can meet together and can recover execution of composite services more quickly, thus increasing the success ratio of invocations. For example, in the open area scenario, the orchestration-based strategy (OS) relying on the LUF has an invocation success ratio between 72 and 76 % against between 60 and 74 % for the TUF. The choreography-based strategy (CS) provides an invocation success ratio between 64 and 71 % with the LUF against between 54 and 63 % with the TUF. Moreover, we notice that the success ratio is better in the sport event scenario, because the mobility of people is constrained as they move along the running path.

Figure 5 shows the success ratio against the number of services per composite request. The invocation success ratio decreases when the number of services to compose increases, because the number of failures (or interruptions) in the invocation processes increases with the number of services. In our scenarios, the failures or the interruptions are mainly induced by the mobility of the devices. Like in the first evaluations, the configuration that provides the better success ratio is the orchestration-based strategy with the LUF. The reasons are the same as those related to the variation of the number of hops. For example in the open area scenario, the orchestration–based strategy provides an invocation success ratio between 35 and 71 % with the LUF, and between 25 and 64 % with the TUF. In the sport event scenario, the gap between the two strategies and the two utility functions is reduced, and the ratio is better than in the open area scenario, due to the

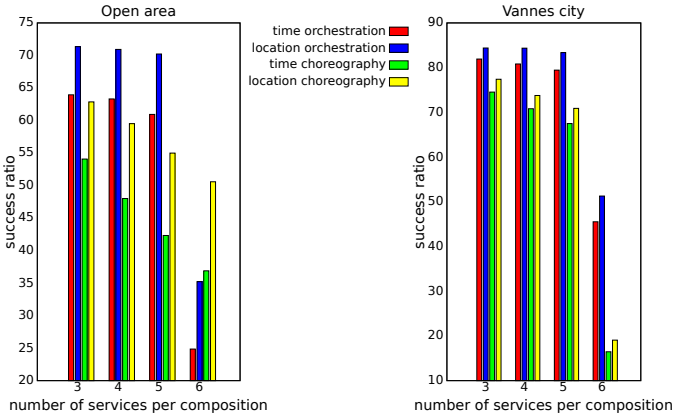

Fig. 5: Success ratio against the number of services per composition.

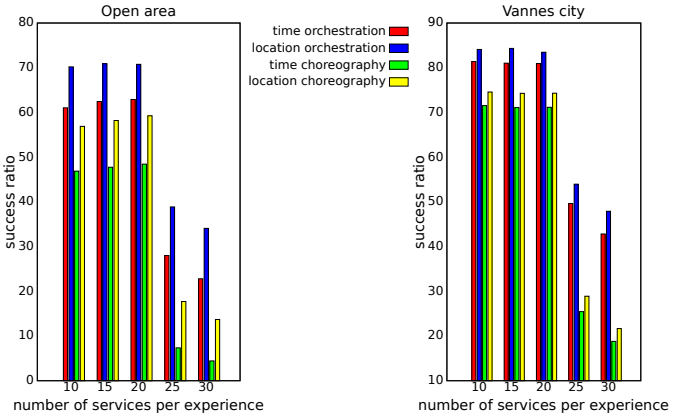

Fig. 6: Success ratio against number of services per experiment.

mobility of people being constrained by the running path, as mentioned before. Therefore it is more easy to meet service providers.

Figure 6 shows the success ratio against the number of services per experiment. Both strategies and both utility functions present the same trend. Indeed, the invocation success ratio stays fairly constant until the number of services per experiment reaches 25 services. Beyond this value, the ratio decreases in a substantial way. For instance in the sport event scenario, the orchestration-based strategy combined with the TUF provides an invocation success ratio approximately of 80 % when the number of services per experiment is between 10 and 20. This ratio falls drastically between 40 and 50 % when the number of services exceeds 20 services. This decrease is explained by the fact that the number of instances of a same service is lower in the network since the number of services deployed in the devices remains constant and the number of types of services increases. The probability of invoking a service provider successfully is consequently lower. It must be noticed that, once again, the LUF offers a greater invocation success ratio than the TUF regardless of the invocation strategy. With the LUF, the success ratio is between 34 and 70 %, whereas with the TUF the success ratio is between 23 and 61 %.

5.4 Execution time of composite request

Figure 7 shows the median value of the execution time against the maximum number of hops between the clients and the providers. The CS outperforms the OS regardless of the utility function and the scenario. This important gap between the two strategies is mainly due to their

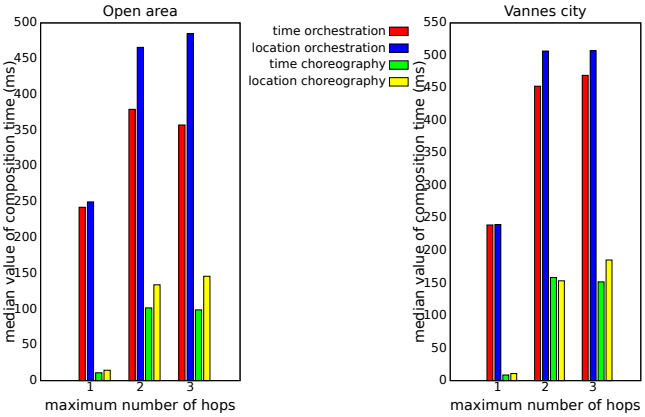

Fig. 7: Median of invocation time against the number of hops between service clients and service providers.

running principle, but also to the process of formation of groups in Wi-Fi Direct. Indeed as mentioned previously in this paper, with the OS, the composite request remains on the device that has produced this request, and this one selects and invokes the service providers it has discovered and that must be enlisted in an invocation. As shown in [2], with the OS and the communication technology we consider, the providers enrolled in an invocation are mainly one hop neighbors (i.e., by the Wi-Fi Direct group owners). This difference between the CS and the OS, is more significant when the maximum number of hops of messages is limited to one, because the device that has produced the composite request can only access its local services and those offered by its group owner in its Wi-Fi Direct group. Thus to complete an invocation, this device must often leave a group and join another one in order to invoke the rest of services. The CS does not have such limitations, even with a maximum number of hops equal to one, since the composite request is transmitted from one provider to another until the invocation is completed. For instance, with a maximum number of hops equal to one, a requester, which is a client of a group owner, can delegate the invocation to its group owner, and this one can delegate, in turn, the rest of the invocation to one of the providers hosted by its clients. Moreover, with the CS, intermediate responses are not returned to the requester, thus reducing the number of exchanges and the time between two successive invocations of services. Figure 7 also shows that when the maximum number of hops increases, the execution time of an invocation increases too. Indeed, when services to invoke are not available in the current communication group, the requester or the current provider should leave and joins another group where the next service is available. The process of leaving, traveling, and joining another group, introduces additional delays.

Figure 8 shows the median of execution time of an invocation against the number of services per composite request. The median of the invocation time increases logically and inevitably as the number of services listed in a service composite request increases. We can also observe that this median grows significantly for 6 services per composition. The explanation of this variation resides in the fact that is more difficult to find the sixth service locally or on the close devices, especially for the OS. Indeed, 6 different services are chosen among the 20 existing ones to define the composite requests, knowing that 5 services among the 20 ones are deployed on each devices. Like in Figure 7, we can see that the CS offers better execution times than the OS regardless the number of services per composition and the utility function. The reasons are the same that those detailed in the previous paragraph.

Figure 9 presents the average of invocation time against the number of services per experiment. The number of instances of a same service available in the network decreases, as the number of types of services increases while the number of services deployed on the devices remains the same. The probability of discovering and invoking a required service is therefore lower, and the invocation time higher. Moreover, with the OS, the difference between the execution time of the invocation

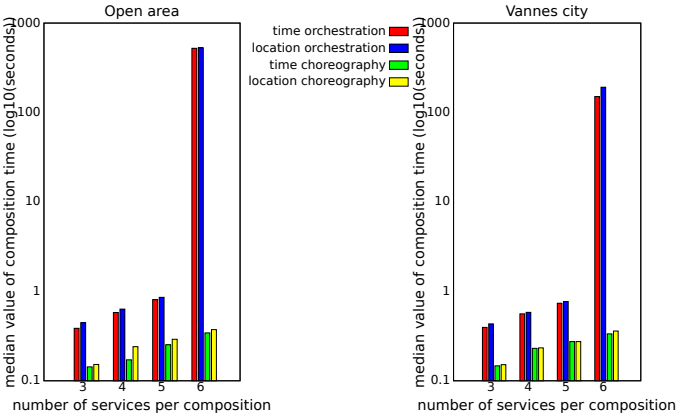

Fig. 8: Median of invocation time against the number of services per composition.

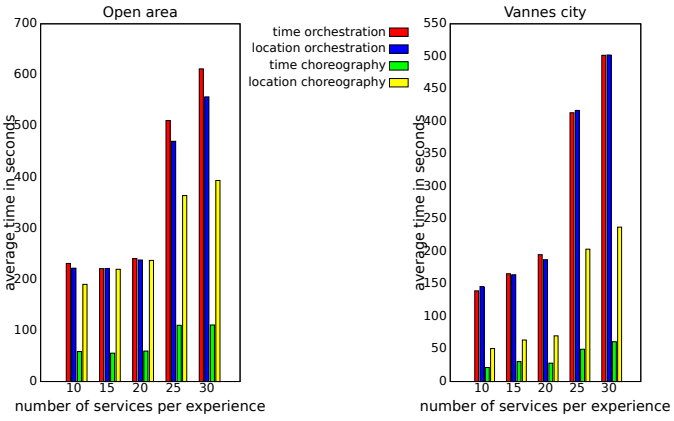

Fig. 9: Average of invocation time against number of services per experiment.

obtained with the TUF and the LUF seems to be not significant, whereas with the CS the difference is observable. CS combined with the TUF provides better performances than if it is combined with the LUF, because with LUF we select the service providers that are on average close to the service clients. These providers do not necessarily provide on average the best service invocation time.

We conclude that the location information helps make composite service invocations more reliable, but it does not help select the providers that offer the shortest invocation times. Moreover, the OS guarantees a higher success ratio than the CS, whereas, from the invocation time perspective, the CS proved to be significantly faster than the OS.

6 Conclusion

Device-to-device communication (D2D) is receiving more attention as direct communication between proximate devices through short-range wireless interfaces offer new data exchange schemes, that could help supporting the always growing data traffic, while increasing at the same time the computing pervasiveness of our physical environment. This kind of communication allows users to exploit directly, or through intermediate devices, the resources offered by the devices populating this environment, resources that are usually made exploitable through services. Due to the limited resources of the devices, applications designed to be used in such environments are generally developed as a combination of basic services that are distributed over a collection of devices. But the connectivity disruptions induced by device mobility and sleep phases hinder end-to-end communications, and therefore jeopardize the reliable overall execution of applications. Opportunistic networking techniques make it possible to cope with these connection disruptions, but they introduce, in return, delays in the exchange of data. In such a context, and assuming that a same service may be offered by several providers in the network, it is important to select the right providers enrolled in the execution of a composite service so as to reduce as much as possible the invocation time, while keeping a certain level of execution reliability.

In this paper, we have presented a service composition system that leverages service redundancy to improve the execution of applications designed as an assembly of composites services in opportunistic networks. This system implements an automatic composition of services from discovered services, according to the users' interests. In contrast to related works presented in Section 2, which only consider choreography-based invocations of composite services, the invocation process of a composite service in our system can follow two different strategies, based on orchestration or choreography. Like in [19, 15, 18, 13], invocations of composite services can be recovered from partial failures. A key feature of the composite service invocation process is the ability to select the best provider at each step. To do so, [8, 31, 19] consider the mobility pattern of devices and their colocation duration. In [15], this is the service execution times and an estimation of the remaining execution path's reliability that help select providers, while in [28] this is the minimum time needed to send data from one node to another and the service load. In this paper, we have investigated two alternative implementations of a utility function at the basis of this service selection: one based on location and the other based on the reception times of the service advertisements emitted by service providers. These two implementations make it possible to take into account the nodes' colocation property, an estimation of the time needed to invoke a remote service provider and an estimation of the success rate of a service invocation. These two implementations can serve both the orchestration and the choreography invocation strategies.

The evaluation of the composition system shows that there is no better invocation strategy than the other, and that both have their advantages and their drawbacks according to the network topology and service distribution. For instance, the choreography strategy combined with the time-based implementation provides a shorter execution time, while the orchestration strategy combined with the location-based implementation offers the best success ratio of composition. Thus, these results show that a relevant future work would be to weave the orchestration and the choreography invocation strategies when invoking a composite service, so that, at each step of a composition, the more efficient strategy could be applied.

References

[1] Idir Aoudia, Saber Benharzallah, Laid Kahloul, and Okba Kazar. Service composition approaches for Internet of Things: a review. *International Journal of Communication Networks and Distributed Systems*, 23(2):194–230, May 2019.

[2] Fadhlallah Baklouti, Nicolas Le Sommer, and Yves Mahéo. Choreography-based vs Orchestration-based Service Composition in Opportunistic Networks. In *13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2017), Rome, Italy*. IEEE, 2017.

[3] Lina Barakat, Simon Miles, and Michael Luck. Adaptive composition in dynamic service environments. *Future Generation Computer Systems*, 80:215–228, 2018.

[4] Nizar Ben-Sassi, Xuan-Thuy Dang, Johannes Fähndrich, Orhan-Can Görür, Christian Kuster, and Fikret Sivrikaya. Service Discovery and Composition in Smart Cities. In *30th International Conference on Advanced Information Systems Engineering (CAiSE 2018), Tallinn, Estonia*, volume 317 of *Lecture Notes in Business Information Processing*, pages 39–48. Springer, June 2018.

[5] Nanxi Chen, NicolÃ s Cardozo, and SiobhÃ n Clarke. Goal-Driven Service Composition in Mobile and Pervasive Computing. *IEEE Transactions on Services Computing*, 11(1):49–62, 2018.

[6] Nanxi Chen and Siobhán Clarke. A Dynamic Service Composition Model for Adaptive Systems in Mobile Computing Environments. In *12th International Conference on Service-Oriented Computing (ISCOC 2014), Paris, France*, pages 93–107. Springer, 2014.

[7] Marco Conti, Emanuel Marzini, Davide Mascitti, Andrea Passarella, and Laura Ricci. Service Selection and Composition in Opportunistic Networks. In *9th International Wireless Communications and Mobile Computing Conference (IWCMC 2013), Cagliari, Italy*, pages 1565–1572. IEEE, 2013.

[8] Lucia Del Prete and Licia Capra. Reliable Discovery and Selection of Composite Services in Mobile Environments. In *12th Enterprise Distributed Object Computing Conference (EDOC'08), Munich, Germany*, pages 171–180. IEEE, 2008.

[9] Shuiguang Deng, Longtao Huang, Javid Taheri, Jianwei Yin, MengChu Zhou, and Albert Y. Zomaya. Mobility-Aware Service Composition in Mobile Communities. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(3):555–568, 2017.

[10] Shuiguang Deng, Zhengzhe Xiang, Jianwei Yin, Javid Taheri, and Albert Y. Zomaya. Composition-Driven IoT Service Provisioning in Distributed Edges. *IEEE Access*, 6:54258–54269, 2018.

[11] ETSI. Etsi technical specification 103 264 v1.1.1, 2015.

[12] Martin Garriga, Cristian Mateos, Andres Flores, Alejandra Cechich, and Alejandro Zunino. RESTful service composition at a glance: A survey. *Journal of Network and Computer Applications*, 60:32–53, 2016.

[13] Christin Groba and SiobhÃ n Clarke. Opportunistic Composition of Sequentially-Connected Services in Mobile Computing Environments. In *International Conference on Web Services (ICWS 2011), Washington DC, USA*, pages 17–24. IEEE, 2011.

[14] Christin Groba and SiobhÃ n Clarke. Synchronising Service Compositions in Dynamic Ad Hoc Environments. In *1st International Conference on Mobile Services (MS'12), Hawaii, USA*, pages 56–63. IEEE, 2012.

[15] Christin Groba and SiobhÃ n Clarke. Opportunistic Service Composition in Dynamic Ad Hoc Environments. *IEEE Transactions on Services Computing*, 7(4):642–653, 2014.

[16] Marzieh Hamzei and Nima Jafari Navimipour. Toward Efficient Service Composition Techniques in the Internet of Things. *IEEE Internet of Things Journal*, 5:3774–3787, October 2018.

[17] Noha Ibrahim and Frédéric Le Mouël. A Survey on Service Composition Middleware in Pervasive Environments. *International Journal of Computer Science Issues*, 1:1–12, 2009.

[18] Shanshan Jiang, Yuan Xue, and Douglas C. Schmidt. Minimum disruption service composition and recovery in mobile ad hoc networks. *Computer Networks*, 53(10):1649–1665, 2009.

[19] Swaroop Kalasapur, Mohan Kumar, and Behrooz Shirazi. Seamless Service Composition (SeSCo) in Pervasive Environments. In *1st International Workshop on Multimedia Service Composition (MSC'05), Hilton, Singapore*, pages 11–20. ACM, 2005.

[20] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The ONE simulator for DTN protocol evaluation. In *2nd International Conference on Simulation Tools and Techniques (SIMUTools'09), Rome, Italy*, number 55. ICST, March 2009.

[21] Nicolas Le Sommer and Salma Ben Sassi. Location-based Service Discovery and Delivery in Opportunistic Networks. In *9th International Conference on Networks (ICN 2010), Les Ménuires, France*, pages 179–184. IEEE, 2010.

[22] Nicolas Le Sommer, Pascale Launay, and Yves Mahéo. A Framework for Opportunistic Networking in Spontaneous and Ephemeral Social Networks. In *10th MobiCom Workshop on Challenged Networks (CHANTS 2015), Paris, France*, pages 1–4. ACM, 2015.

[23] Nicolas Le Sommer, Romeo Said, and Yves Mahéo. A Proxy-based Model for Service Provision in Opportunistic Networks. In *6th International Workshop on Middleware for Pervasive and Ad-hoc Computing (MPAC'08), Louvain, Belgium*. ACM, 2008.

[24] Angel Lagares Lemos, Florian Daniel, and Boualem Benatallah. Web Service Composition: A Survey of Techniques and Tools. *ACM Computing Surveys*, 48(3):1–41, 2016.

[25] Ali Makke, Yves Mahéo, and Nicolas Le Sommer. Towards Opportunistic Service Provisoning in Intermittently Connected Hybrid Networks. In *4th International Conference on Networking and Distributed Computing (ICNDC 2013), Honk Kong, China*, pages 28–32. IEEE, 2013.

[26] Vinicius F. S. Mota, Felipe D. Cunha, Daniel F. Macedo, José M. S. Nogueira, and Antonio A. F. Loureiro. Protocols, Mobility Models and Tools in Opportunistic Networks: A Survey. *Computer Communications*, 48:5–19, 2014.

[27] Hajo Rijgersberg, Mark Van Assem, and Jan Top. Ontology of Units of Measure and Related Concepts. *Semantic Web Journal*, 4(1):3–13, 2013.

[28] Umair Sadiq, Mohan Kumar, Andrea Passarella, and Marco Conti. Service Composition in Opportunistic Networks: A Load and Mobility Aware Solution. *IEEE Transactions on Computers*, 84(8):2308–2322, 2015.

[29] Adrián Sánchez-Carmona, Frédéric Guidec, Pascale Launay, Yves Mahéo, and Sergi Robles. Filling in the missing link between simulation and application in opportunistic networking. *Journal of Systems and Software*, 142:57–72, 2018.

[30] W3C. Web of Things (WoT) Thing Description. W3C Candidate Recommendation, 2019.

[31] Jianping Wang. Exploiting Mobility Prediction for Dependable Service Composition in Wireless Mobile Ad Hoc Networks. *IEEE Transactions on Services Computing*, 4(1):44–55, 2011.

[32] Walid Younes, Sylvie Trouilhet, Françoise Adreit, and Jean-Paul Arcangeli. Towards an Intelligent User-Oriented Middleware for Opportunistic Composition of Services in Ambient Spaces. In *5th Workshop on Middleware and Applications for the Internet of Things (M4IoT 2018), Rennes, France*, pages 25–30. ACM, 2018.