Opportunistic composition of sequentially-connected services in mobile computing environments

Christin Groba and Siobhán Clarke
Lero Graduate School of Software Engineering
Distributed Systems Group
School of Computer Science and Statistics
Trinity College Dublin, Ireland
grobac, Siobhan.Clarke@scss.tcd.ie

***Abstract*—Dynamic service composition has emerged as a promising approach to build complex runtime-adaptable applications as it allows for binding service providers only shortly before service execution. However, the dynamic and ad hoc nature of mobile computing environments poses a significant challenge for dynamic service composition. In particular, the lack of central control and the potential volatility of service providers increase the complexity and failure probability of the composition process. Although, current research has led to decentralised composition algorithms and failure recovery strategies, the key question of how to reduce the failure probability of a composition still remains. We address this question and propose opportunistic service composition, an optimised execution model for complex service requests. The model merges the execution phase into the dynamic binding phase and supports the immediate fulfilment of partially composed service requests. We evaluated our model in mobile ad hoc network simulations. The results show an improvement over a baseline approach regarding composition success, response time, and communication effort.**

***Keywords*-service binding; service execution; distributed; mobile; ad hoc**

I. Introduction

The composition of new value-added services from existing ones is one of the main benefits of service-oriented computing. Building such service compositions at runtime allows for adapting the composition dynamically to changes in the user requirements or the operating environment. For example, service providers can be dynamically selected or replaced based on their current availability and actual runtime characteristics [1].

Advances in mobile device technology have led to smart personal devices that are capable of communicating with each other and of providing access to their data and functional capabilities. These services may be of use to a wide range of consumers and shared by the device owner. In this domain, service composition will play an important role when it comes to mining the computational power that is created when many owners of such smart devices meet. For example, a complex service request for a route to a destination may only be satisfied with the collective effort of multiple smart phones, GPS-enabled sports gear, and on-board navigation systems that happen to gather on a street corner.

However, dynamic service composition in mobile environments is challenging for two reasons: First, service providers may join and leave the network at runtime and their fluctuating presence implies a high failure probability for the composition [2]. Second, a central entity of control is infeasible in such dynamic environments because of the large overhead required to keep global state information up to date. The lack of central control introduces additional complexity into the composition process because a dedicated infrastructure for composition-related tasks is missing [3].

Existing approaches present distributed composition and execution algorithms to address the problem of absent control, to balance the system load, and to remove the single hot spot for communication and computation. However, the problems regarding dynamic provider availability and resource constraints remain because state-of-the-art approaches are based on a consecutive organisation of the service composition and execution phase. Starting execution only after the composition is complete has two major drawbacks: First, assigned service providers may no longer be available by the time the execution phase starts and cause the composition to fail. Second, the composition phase binds service providers to parts of the request which may not get executed due to conditional service path or premature termination. The communication involved consumes scarce battery power of wireless devices in vain.

This work proposes an optimised execution model for service compositions to address these issues. In particular, it integrates the execution phase into the composition phase and allows for executing a partial composition. The advantage of such an opportunistic approach is that it addresses each assigned service provider only once and requires less

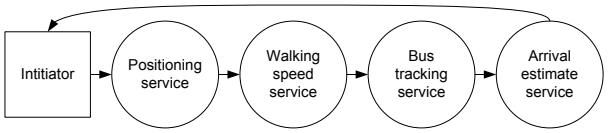

Figure 1. Service request to decide whether a bus will be caught in time

network communication than a traditional consecutive approach. We demonstrate the effectiveness of opportunistic service composition by comparing it to a baseline approach with regard to composition success, response time, and communication effort. We focus on the evaluation of sequentially connected services and leave more complex composition logic such as conditional, parallel, and iterative paths to future work.

The main contribution of this paper is the evaluation of a new model for managing complex application requests in resource-constrained mobile environments. In particular, it introduces opportunistic service composition as a solution to reducing the failure probability caused by mobile service providers. The paper describes the approach in the context of a smart city in which data and capabilities of mobile devices are shared ad hoc across different device owners. The evaluation with a mobile ad hoc network simulator shows that the opportunistic service composition has a higher success ratio, shorter response time, and less communication overhead than consecutive bind-execute approaches.

The remainder of this paper is organised as follows: Section II motivates dynamic service composition in mobile environments with an scenario in a smart city. Section III summarises work related to the composition and execution of complex service request. Section IV describes the opportunistic execution model. Section VI presents the evaluation and result. Section VII summarises the findings of this paper and VIII discusses the implications of the opportunistic service composition model.

II. Scenario: Service sharing on the street

Crossroads in the city centre are usually a busy places. Pedestrians and vehicles arrive from different directions and carry mobile devices such as smart phones, music players, GPS-receivers, and navigation systems. Currently, these devices offer their data and capabilities only to their owners. However, in future device owners may agree to share their devices' services as they do today with files and personal information on the Internet. This would give way for complex service requests in mobile ad hoc environments. For example, a pedestrian may be heading to the airport and wants to know whether there is still enough time to catch the cheaper airport bus rather than taking a faster but more expensive taxi. For this decision bus schedule times do not suffice and need to be enhanced with real-time information that answer questions like: What is my current position, what is the average walking speed on the high street, where was the bus last seen, and at which time will I and the bus arrive at a particular bus stop? The pedestrian's mobile device may not have the resources and data to answer all these questions and thus outsources the complex service request to surrounding mobile devices. For instance, the GPS-enabled sport gear carried by an oncoming jogger may provide the walking speed on the high street while a vehicle's navigation system calculates the estimated arrival time at the bus stop. The request is handled in a distributed way such that each device executes a service and forwards the request on to the next device (see Figure 1). The service providers have to be bound dynamically at runtime because assumptions about the provider availability cannot be made prior to issuing the service request.

III. Related work

Research in handling complex service requests at runtime may be grouped into service composition and service execution. The emergence of dynamic and ad hoc computing environments has led to decentralised approaches in both areas.

Decentralised service composition examines different ways of finding suitable service providers without relying on central coordination or repositories. Service providers may be bound to requests through probing or hop-by-hop selection. Probing approaches search service overlay networks for multiple high-quality paths by flooding the overlay network with probe messages. Probe-based binding algorithms do not allow for opportunistic service composition because they either require a completed composition before they are executable [4] or postpone final best path selection to the requester node [5], [6], [7]. Hop-by-hop provider selection determines the most suitable provider in each hop, discovers a single high-quality solution for a complex request and is thus more appropriate for immediate service execution. Existing solutions propose resource-efficient [8] and ad hoc [3] composition algorithms for service overlay networks. However, they either do not consider provider mobility [8] or do not investigate the possibility of executing a provider immediately after its selection [3].

Decentralised service execution addresses the scalability issues of central orchestration engines and investigates how to transfer control among multiple executing entities. A centralised process description has to be partitioned into fragments to be enacted by multiple decentralised engines. One approach to achieve this creates sub-processes based on dependency tables that store data and control dependencies of the complex request [9]. In this paper we investigate requests with sequentially connected services. Advanced dependency management is not required because each service only has a single dependency, namely the previous service, and process fragmentation is straight forward. Approaches to distributed coordination among service providers are based on direct

interaction [10], [11] or use shared tuplespaces [12], [13]. The advantage of shared tuplespaces is that communicating parties do not have to be present at the same time, as they anonymously exchange data and control via a piece of shared memory. However, shared tuplespaces consume high resources and suffer from efficiency and scalability issues due to global synchronisation requirements [14]. Continuation passing [10] and migration of process descriptions [11] use direct interaction and are similar to our approach of transferring control among service providers. Continuation passing allocates control hop-by-hop and activates subsequent service providers by passing the remainder of the execution along with asynchronous messages. However, the concept is based on activity pairing and requires extra initialisation of participating providers which increases the communication overhead. The closest work to our approach is the migration of a process description among execution engines [11]. It allows for selecting a new engine after an activity has been completed. However, while Zaplata et al. [11] analyse the extent to which XPDL [15] and WS-BPEL [16] process descriptions can be migrated, our objective in this paper is to quantify the benefit of opportunistic service composition in mobile ad hoc environments.

Service composition can be modelled as an AI planning problem where the objective is to create a service composite automatically by planning the execution order of the constituent services. Recent work has proposed a distributed planning approach for Web service compositions [17]. A global plan is devised based on partial plans that multiple service agents create locally. However, in dynamic environments, changes in the system may render an initial plan invalid. A declarative approach to Web service composition [18] interrupts the execution of a composite and re-plans after a failure has occurred. In this paper, we assume the complex service request is phrased as an abstract composite that already contains the execution order and description of the required services. We do not employ an AI planner. However, if binding service providers and transforming an abstract composite into an executable composite is considered planning, then opportunistic service composition corresponds to the notion of continual planning [19]. In contrast to [18], our approach plans only one step ahead, executes the partial plan, and resumes planning. Our goal is to investigate whether this interleaved service composition and execution approach [20] can be applied in distributed mobile ad hoc environments to reduce the failure probability of a service composite.

IV. Opportunistic service composition

Mobile ad hoc computing environments require a fully decentralised peer-to-peer-like way of composing and executing complex service requests. As in other approaches [8], [10], the peers in our model operate on only a part of the request and send the remainder to the next peer. This

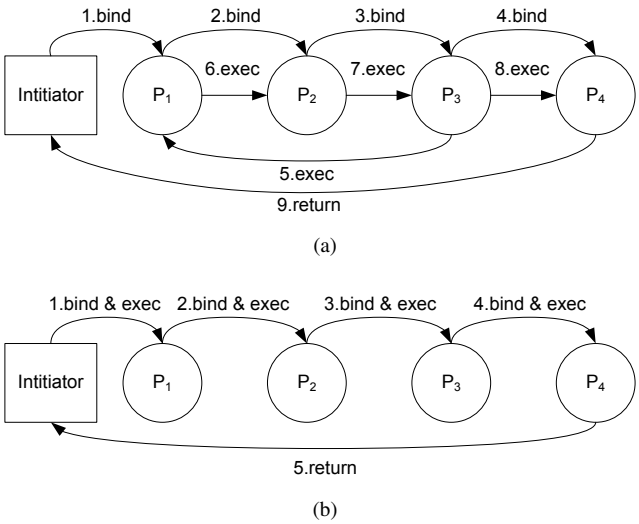

Figure 2. Service composition (a) baseline and (b) opportunistic

partial request processing involves two major tasks: binding a service provider address to a required sub-service and executing a sub-service based on the output of the previous sub-services.

We model complex service requests as composites of service types and focus in this paper on sequential composites. Service types are identifiers for abstract service descriptions. They have the advantage that the size of binding and execution messages stays small because detailed service requirements are implicitly specified by the identifier. This requires a common notion of possible service types and what they represent. During service binding the composite changes as particular service providers P_i get assigned to service types S_j. The Backus-Naur form of composites is:

$$\begin{aligned} < type > &::= S_1|...|S_n \\ < provider > &::= P_1|...|P_m \\ < composite > &::=< type > | \\ &\quad < type >:< provider > | \\ &\quad seq(< composite >, < composite >) \end{aligned}$$

Traditionally, the execution of a composite starts only after all constituent services have been bound to a provider. In a decentralised environment this implies that a request is sent around two times: first to bind appropriate service providers, then to invoke them (see Figure 2a). The consecutive order of binding and executing a service has two drawbacks: First, assigned providers may no longer be available by the time they are invoked and the composition fails. Second, peers communicate at least twice over the network to process the complex service request. This affects the life-time of wireless, battery-powered devices because network communication is one of the major causes for fast energy depletion [21].

The design objective is to reduce the composite's exposure to failures inherent in mobile computing environments. A way to achieve this is by reducing the number of interactions between the service providers during service binding and execution. The novel composition model seizes the opportunity of executing a service right away after it has been bound. The rationale behind this is: if a potential service provider has replied to a discovery request, it is likely to be still available when the execution request is sent immediately after receiving the provider's discovery reply. The opportunistic composition model thus combines the binding and execution phase by carrying out a constituent service while searching a provider for the next sub-service. It immediately executes partially-composed service requests without binding all sub-services in advance (see Figure 2b).

Service binding in dynamic environments is a non-trivial task because service providers are a priory unknown and have to be discovered at runtime. Dynamic service discovery in mobile ad hoc environments may be classified into directory-based, and directory-less approaches [22]. We use directory-less reactive discovery (as opposed to proactive service advertisement) and trade-off less network traffic with longer binding delays. The node, which is currently in binding control, issues a discovery request for the next service to bind and waits for potential service providers to respond. If multiple providers are available that offer the same functional service, we apply proximity-based provider selection. The provider, whose discovery response is received first, is automatically bound. This discovery and selection approach is sufficient to provide us with the means to study opportunistic service composition focusing on integrating the execution phase into the binding phase. Other work on, for example, distributed trust-aware service selection [23] and fair reputation propagation [24], as a way to establish trust, could also be used with our asynchronous composite execution strategy.

V. Experimental setup

This paper studies decentralized service composition in mobile ad hoc networks. In particular, it investigates the impact of opportunistic service composition on the success ratio, response time, and communication effort. A baseline approach implements the consecutive composition strategy of first binding all required services and thereafter executing them. It serves as a reference to calculate the improvement of the opportunistic approach. We implemented both approaches independent from any process execution language. This allows for sufficient detail for the following analysis and the results may be applicable to any concrete process execution language e.g. WS-BPEL [16] or BSPL [25]. We test both approaches with multiple scenarios. Each scenario is defined by a certain composition length and speed range. The composition length is the number of sequentially connected services. The speed range reflects the movement of pedestrians, cyclist, or motorized vehicles in a city centre. The random components of a scenario include the provider's initial placement, mobility, and service execution time. Both approaches are tested with the same scenario configuration, in particular with the same values for the controlled and random variables. Each combination of a composition length and a speed range is simulated 1000 times for each approach. After 2 minutes the simulation terminates which represents an application layer time-out for requests that have not been returned yet. We implemented a no-repeat strategy to ensure decentralized binding. Both composition algorithms have to bind unique service provider such that the request does not contain the same provider for different services. Table I shows further details about the simulation configuration.

Table I
Simulation configuration

General	
Simulator	Jist/SWANS Ulm [26]
Mobility model	Random Waypoint
Field (m^2)	1000x1000
Radio range (m)	250
Providers	16
Clients	1
Simulation duration (min)	2
Random	
Node placement	
Node movement	
Service execution time (ms)	10-100
Controlled	
Composition length	4-7
Composition mode	baseline, opportunistic
Node speed (m/s)	1-2, 2-8, 8-13, 1-13

VI. Results

In the following we analyse the improvement of the opportunistic over the baseline approach in terms of composition success ratio, response time, and communication effort in mobile computing environments.

A. Composition success ratio

Composition success ratio is the number of successfully executed and returned compositions versus the total number of composition requests. In the simulation study, the composition success ratio varies from 24 to 79 percent. The main reason for not achieving 100 percent success is the failure of discovering suitable service providers that are in the range of the requester.

Generally, the success ratio decreases the more services have to be bound. However, Figure 3 shows that the opportunistic approach is more successful in completing a composition across all composition lengths and speed ranges than the baseline. In particular, the percentage increase of opportunistic service composition over the baseline rises the more services have to be bound and executed. This suggests that requests with several sub-services are more likely to

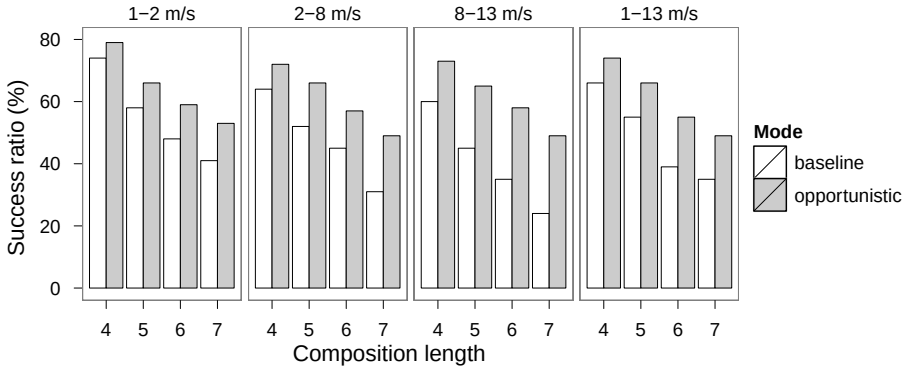

Figure 3. The composition success ratio is greater for opportunistic service composition than for the baseline approach.

succeed in dynamic environments if services are executed immediately.

Analysing the data set across different speed ranges for the same service length also shows a raise in the percentage increase as the gap between the opportunistic and the baseline gets wider. It culminates when the request contains seven sub-services and potential service providers move between 8 and 13 meter per second. Then the success ratio doubles from the baseline with 24 percent to the opportunistic approach with 49 percent. This is an increase of the success ratio by 104 percent and indicates that opportunistic service composition may be well suited for motorised service providers driving at city speed. In addition, the opportunistic approach is less affected by different speeds because its success ratio for the same service length varies less over the different speed ranges than the baseline's ratio. Thus, opportunistic service composition may handle heterogeneous traffic including pedestrians and vehicles better than the baseline.

Overall, across all composition length and mixed traffic with speeds from 1 to 13 meter per second, the opportunistic approach is more successful than the baseline and raises the success ratio by 28 percent. The analysis of the failed baseline compositions revealed that with increasing node speeds compositions are more likely to fail after service execution has started. This supports the argument that in mobile environments bound service providers may no longer be available by the time they are invoked, if the binding phase has to be completed first before the execution phase can start.

In the following we focus on observations that have been successful in both approaches to investigate the effect of the composition mode on the response time and communication effort.

B. Response time

Response time is the duration from issuing a service request to receiving the result of the executed service composition. In the simulation study the mean response time varies between 3.0 and 8.8 seconds for different scenario setups (see Figure 4). As expected, the response time increases with longer requests because more interactions between providers is necessary to bind and execute the composition. Overall, opportunistic service execution returns a composition result quicker than the baseline.

For walking speeds of 1 to 2 meter per second the difference increases, the more services a request contains. For example, to compose four services the baseline needs 4.7 seconds while the opportunistic approach needs only 3 seconds and is 1.7 seconds faster than the baseline. For seven services the opportunistic approach is 3.3 seconds faster because it needs 4.8 seconds for completing the composition while the baseline needs 8.1 seconds. As an aside, the response time for nodes moving between 8 and 13 meter per second is shorter compared to that of slower moving nodes. The reasons for this are not yet clear.

For mixed traffic speeds ranging from 1 to 13 meter per second and across all composition length, opportunistic service composition returns a composition result with 4.9 seconds on average 2.4 seconds faster than the baseline approach which needs 7.3 seconds. This means that the opportunistic approach decreases the response time by 33 percent.

Faster response times are beneficial for two reasons: First, bound service providers are released earlier. This shortens the time a provider has to stay available for the composition to complete successfully. Second, a shorter response time raises the quality of service and will be appreciated by service consumers that are in a hurry and need information instantly.

C. Communication effort

Communication effort is the number of messages that have to be exchanged in order to satisfy a composition request. We focus the analysis on application layer messages, namely for service discovery, binding, and invocation, as well as on routing layer messages required to find routes to the messages' destination. In particular, we analyse the

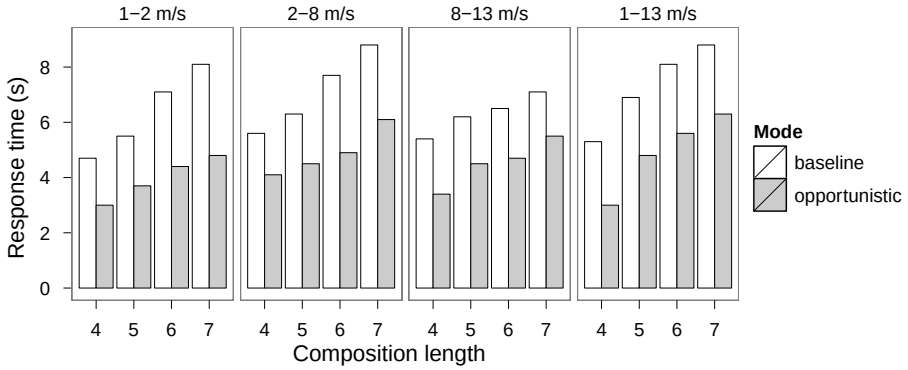

Figure 4. The average response time is generally shorter for the opportunistic approach than for the baseline.

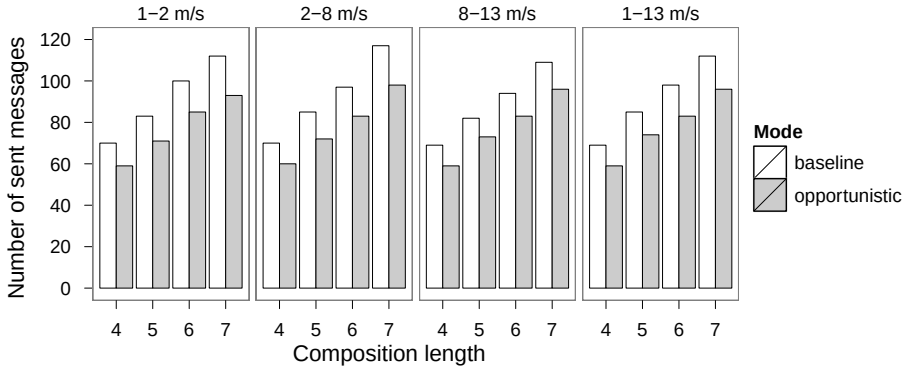

Figure 5. The average number of total messages sent is less for opportunistic composition than for the baseline.

mean number of sent messages because the number of sent messages is less affected by the sender's placement than the number of received messages. In the simulation results shown in Figure 5 the average total number of sent messages varies from 59 to 117 messages. In line with the improvements of the composition success and response time, the opportunistic approach requires less communication for completing a composition successfully than the baseline approach.

As expected, the difference in the number of sent messages increases for a certain speed range across different composition lengths because the opportunistic approach is designed with fewer interactions between providers than the baseline. For example in scenarios with speeds from 1 to 13 meter per second and requests lengths of four services, the baseline sends 69 messages while the opportunistic approach sends 10 messages less, only 59 messages. For seven services, the opportunistic approach sends 16 messages less as it requires only 96 messages while the baseline sends 112 messages.

Generally, the number of sent messages for one composition length and across different speed ranges stays similar. This indicates that the same routing effort is applied regardless at what speed the service providers move. We further analysed the ratio of routing request messages to the total number of sent messages (not depicted). In general, the opportunistic approach has a smaller ratio and sends less routing requests than the baseline. Routing requests are broadcasted in the network and each provider in range has to dedicate some of its battery power to forwarding the route request. The opportunistic approach sends less route requests and conserves the overall energy in the network longer than the baseline.

On average for speeds ranging from 1 to 13 meter per second and across all composition lengths, the opportunistic approach sends 78 messages while the baseline requires 91 messages. The average difference of 13 messages equals a reduction of the communication effort by 14 percent. Reducing communication overhead is beneficial in mobile networks for two reasons: First, less communication over the network exposes the composition less often to failures inherent in mobile environments. Second, wireless communication is the main energy consumer and with fewer interactions between providers the battery of mobile devices may last longer and pro-long the device's availability.

VII. Summary

The dynamic composition of complex service requests allows for flexible software that adapts to the actual service supply and runtime characteristics of its operating environment. New application domains such as smart cities will rely on dynamic service composition to make use of the computational power that is available in form of mobile devices carried by pedestrians, in vehicles and other city-wide locations. However, state-of-the-art approaches that compose first and execute thereafter do not suit dynamic, resource-constrained settings. They expose the composition more often to the failure inherent in mobile environments. For example, a provider may leave the network after it has been bound, causing the composition to fail.

We introduced opportunistic service composition, an optimised execution model for complex service requests. It allows for executing partially composed services in a decentralised fashion and reduces the composition's exposure to unreliable networks. We evaluated the proposed approach against a baseline in different simulation scenarios that varied in node mobility and request complexity. The opportunistic approach improves the composition success ratio in mobile environments because of the opportunistic execution of bound services and not at the expense of response time and communication effort. The results of the simulation study for mixed traffic and different request lengths show, that opportunistic service composition in comparison to a baseline incurs on average a:

- 28 percent increase of the composition success ratio,
- 33 percent decrease of the response time, and
- 14 percent decrease of the communication effort.

The analysis in this paper focused on requests that contain sequentially connected sub-services. In future work we will study the impact of opportunistic service composition on more complex control logic such as conditional, parallel, and iterative paths.

VIII. Discussion

Until now we discussed the problem of high composition failure probability in mobile environments and how the opportunistic approach reduces it. However, other challenges remain and are partly intrinsic to decentralised composition approaches in general and partly emerge because of the opportunistic approach.

The challenges for decentralised composition in general concern resource-constraints and privacy issues. Composition participants need sufficient capabilities to perform service-related tasks e.g., resolving complex mismatches, reasoning about the best provider, verifying the partial composite, and executing the control logic. This creates tension between resource-constrained mobile devices and the demand for increased device intelligence. Dynamic service composition in mobile computing needs comprehensive but light-weight concepts that integrate resource-poor devices. Further, decentralisation and hop-by-hop processing raises major privacy concerns because the control and data flow is visible to assigned providers. Sensitive information such as personal data, identities, and even the nature of some sub-services need to be protected while peers still have effective means to search and evaluate subsequent service providers.

A challenge arising from opportunistic service composition is identifying suitable failure recovery for partially executed compositions. Although the opportunistic approach improves the composition success ratio, it cannot prevent empty search results, short-notice provider drop-out, or requirement mismatches. In mobile ad hoc environments service provision and the success of a request cannot be guaranteed. If a service provider is not available, the partially executed composition remains in inconsistent state. Recovery strategies are indispensable and may be complemented by the proposed model. However, opportunistic service composition may be insufficient for transactional services that require a roll-back strategy in the event of failure. The cost of executing the reverse service may outweigh the benefits of the proposed approach. On the other hand, mobile ad hoc domains such as a smart city provide stateless services to share data and device capabilities. Returning the result of an incomplete composition of stateless services may be more beneficial to the consumer than not receiving any result at all. Alternatively, opportunistic and carefully-planned service composition may co-exist for requests that contain stateless and transactional services.

Acknowledgment

This work was supported, in part, by Science Foundation Ireland grant 03/CE2/I303_1 to Lero - the Irish Software Engineering Research Centre (www.lero.ie). Thanks also to Serena Fritsch and Razvan Popescu for commenting on earlier drafts.

References

[1] Valeria Cardellini, Emiliano Casalicchio, Vincenzo Grassi, Francesco Lo Presti, and Raffaela Mirandola. QoS-driven runtime adaptation of service oriented architectures. In *European Software Engineering Conference and The Foundations of Software Engineering (ESEC/FSE)*, pages 131–140. ACM, 2009.

[2] Vivian Prinz, Florian Fuchs, Peter Ruppel, Christoph Gerdes, and Alan Southall. Adaptive and fault-tolerant service composition in peer-to-peer systems. In *International Conference on Distributed Applications and Interoperable Systems (DAIS)*, pages 30–43. Springer, 2008.

[3] S. Kalasapur, M. Kumar, and B.A. Shirazi. Dynamic service composition in pervasive computing. *Transactions on Parallel and Distributed Systems*, 18(7):907–918, 2007.

[4] Zhenghui Wang, Tianyin Xu, Zhuzhong Qian, and Sanglu Lu. A parameter-based scheme for service composition in pervasive computing environment. In *International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*, pages 543–548. IEEE, 2009.

[5] Xiaohui Gu, K. Nahrstedt, and Bin Yu. Spidernet: An integrated peer-to-peer service composition framework. In *International Symposium on High performance Distributed Computing (HPDC)*, pages 110–119. IEEE, 2004.

[6] Eunjeong Park and Heonshik Shin. Reconfigurable service composition and categorization for power-aware mobile computing. *Transactions on Parallel and Distributed Systems*, 19(11):1553–1564, 2008.

[7] Farshad A. Samimi and Philip K. McKinley. Dynamis: Dynamic overlay service composition for distributed stream processing. In *International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pages 881–886, 2008.

[8] Mea Wang, Baochun Li, and Zongpeng Li. sFlow: Towards resource-efficient and agile service federation in service overlay networks. In *International Conference on Distributed Computing Systems (ICDCS)*, pages 628–635. IEEE, 2004.

[9] W. Fdhila, U. Yildiz, and C. Godart. A flexible approach for automatic process decentralization using dependency tables. In *International Conference on Web Services (ICWS)*, pages 847 –855. IEEE, 2009.

[10] Weihai Yu. Scalable services orchestration with continuation-passing messaging. In *International Conference on Intensive Applications and Services (INTENSIVE)*, pages 59–64. IEEE, 2009.

[11] Sonja Zaplata, Kristof Hamann, Kristian Kottke, and Winfried Lamersdorf. Flexible execution of distributed business processes based on process instance migration. *Journal of Systems Integration*, 1(3):3–16, 2010.

[12] D. Martin, D. Wutke, and F. Leymann. A novel approach to decentralized workflow enactment. In *International IEEE Enterprise Distributed Object Computing Conference (EDOC)*, pages 127–136. IEEE, 2008.

[13] H. Fernández, T. Priol, and C. Tedeschi. Decentralized approach for execution of composite web services using the chemical paradigm. In *International Conference on Web Services (ICWS)*, pages 139 –146. IEEE, 2010.

[14] Hassan Artail, Rula Antoun, and Kassem Fawaz. CRUST: Implementation of clustering and routing functions for mobile ad hoc networks using reactive tuple-spaces. *Ad Hoc Networks*, 7(6):1064 – 1081, 2009.

[15] WFMC. Workflow management coalition workflow standard: Workflow process definition interface – XML process definition language (XPDL). Technical Report WFMC-TC-1025, Workflow Management Coalition, 2002.

[16] OASIS. Web service business process execution language version 2.0. Technical report, OASIS, 2007.

[17] M. El Falou, M. Bouzid, A.-I. Mouaddib, and T. Vidal. A distributed planning approach for web services composition. In *International Conference on Web Services (ICWS)*, pages 337 –344. IEEE, 2010.

[18] Therani Madhusudan and N. Uttamsingh. A declarative approach to composing web services in dynamic environments. *Decis. Support Syst.*, 41:325–357, 2006.

[19] M.E. desJardins, E.H. Durfee, C.L. Ortiz, and M.J. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, 20(4):13–22, 1999.

[20] Vikas Agarwal, Girish Chafle, Sumit Mittal, and Biplav Srivastava. Understanding approaches for web service composition and execution. In *Proceedings of the 1st Bangalore Annual Compute Conference*, pages 1:1–1:8. ACM, 2008.

[21] Christin Groba and Siobhán Clarke. Web services on embedded systems - A performance study. In *International Workshop on the Web of Things (WoT) at PERCOM*, pages 726–731. IEEE, 2010.

[22] C.N. Ververidis and G.C. Polyzos. Service discovery for mobile ad hoc networks: a survey of issues and techniques. *Communications Surveys Tutorials, IEEE*, 10(3):30 –45, 2008.

[23] Chung-Wei Hang and Munindar P. Singh. Trustworthy service selection and composition. *ACM Transactions on Autonomous and Adaptive Systems*, 6(1):5:1–5:17, February 2011.

[24] S. Nepal, Z. Malik, and A. Bouguettaya. Reputation propagation in composite services. In *International Conference on Web Services (ICWS)*, pages 295 –302. IEEE, 2009.

[25] Munindar P Singh. Information-Driven Interaction-Oriented Programming : BSPL, the Blindingly Simple Protocol Language. In *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 2–6, 2011.

[26] JiST/SWANS Edition of Ulm University. http://vanet.info/jist-swans.