Towards In-network Aggregation for People-Centric Sensing

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. Technological advances in the smartphone sector give rise to people-centric sensing that uses the sensing capabilities of mobile devices and the movement of their human carriers to satisfy the ever increasing demand for context information. The quick adoption of such pervasive and mobile services, however, increases the number of contributors, strains the device-to-server connections, and challenges the system's scalability. Strategies that postpone load balancing to fixed infrastructure nodes miss the potential of mobile devices interconnecting to preprocess sensor data. This paper explores opportunistic service composition to coordinate in-network aggregation among autonomous mobile data providers. The composition protocol defers interaction with peers to the latest possible moment to accommodate for the dynamics in the operating environment. In simulations such an approach achieves a higher composition success ratio at similar or less delay and communication effort than an existing conventional composition solution.

1 Introduction

Mobile devices have evolved from special purpose equipment to smart entities that can sense their environment and communicate with servers on the Internet. People-centric sensing projects, as presented in [8], use these technological advances and the movement of human carriers to improve the micro- and macroscopic view on today's and future cities. Bubble-sensing [10], for example, affixes a sensing task to a certain location and entrusts mobile devices in vicinity to upload sensor data to a server as a basis for deriving higher level context-information.

However, for sensing tasks that require multiple independent readings from different sensors, the number of individual data uploads increases. In addition, the growing demand for context information creates many of such complex sensing tasks that further strain the device-to-server connection. Targeting the scalability issues of such centralised architectures, research has recently explored ways to balance the load for data processing [2,12].

* Partially supported by Science Foundation Ireland grant 03/CE2/I303_1 to Lero - the Irish Software Engineering Research Centre (http://www.lero.ie).

K. Zheng, M. Li, and H. Jiang (Eds.): MOBIQUITOUS 2012, LNICST 120, pp. 115–126, 2013.

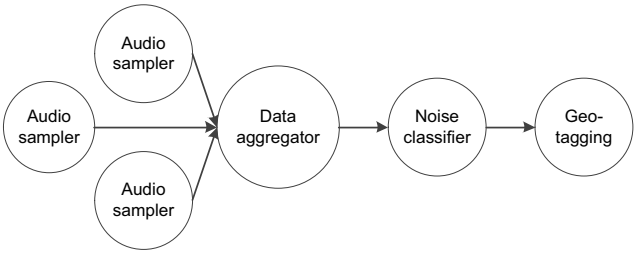


Fig. 1. In-network aggregation request phrased as service composite

In line with these efforts, this work investigates the application of service-oriented principles, namely dynamic service composition, to support in-network aggregation among mobile devices. In-network aggregation is well-studied in wireless sensor networks [1], however, people-centric sensing introduces participation autonomy and mobility that, traditionally not part of those networks, changes the system topology frequently. Dynamic service composition creates new value-added services from existing ones by discovering, allocating, and invoking service providers only at runtime to accommodate for the dynamics in the operating environment. Imagine, for example, a cloud service that provides a noise map of the city. For each geographic area it requires multiple independent audio samples to improve its quality. Instead of transmitting small data packets via multiple individual connections, co-located mobile devices collaborate to aggregate their recordings, add noise classifications and geo-tags, and upload one big data packet from one device. The cloud service posts its complex query and leaves the management of such a service composite (Figure 1) entirely to the ad hoc network of mobile nodes. This way, the query flexibly adjusts to currently available providers, their interconnections and localised knowledge about classification patterns.

This paper proposes opportunistic service composition, a composition protocol that defers all interactions with peers to the latest possible moment. In contrast to our previous work [3,4], it explores service composition from the perspective of autonomous service providers and how on-demand discovery, just in time release, and observation of the composition progress enables them to cooperatively control their availability. Focusing on the composition management within the ad hoc network, the paper leaves the integration with the cloud service to future work. With the analysis of dynamic service composition in mobile ad hoc environments, we hope to contribute to a better understanding of how complex tasks, such as in-network data aggregation, can be coordinated among autonomous and transiently available providers of sensor data.

2 Related Work

Scalability issues of centralised, single-server systems have led to alternative architectures that support load balancing in people-centric sensing. In G-sense [12],

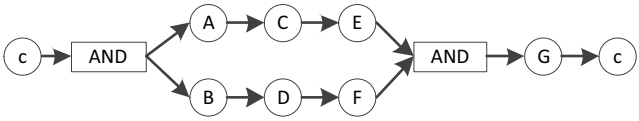
Internet servers create a peer-to-peer sensing overlay to manage the distributed collection and aggregation of sensor data. An alternative approach [2] leverages existing cluster or cloud infrastructure to evaluate different tree-based aggregation strategies. Sensor and context cloudlets [9] use computers with Internet connection, deployed on local business premises and in the vicinity of mobile device to pool sensory data and to provide data fusion and context reasoning. Mobile devices in these solutions are sole data sources that connect to fixed infrastructure to trigger data processing there. In contrast, this work investigates the possibility of mobile devices interconnecting to process complex data queries within their ad hoc network.

CountTorrent [6] devises aggregation techniques for sensor networks whose topology, in contrast to traditional wireless sensor networks, is continuously evolving. While efficient and accurate, the solution assumes simple queries that include all currently available nodes. Queries phrased as abstract service composites, on the other hand, allow for more complex and selective specifications. However, their enactment, in particular the discovery and allocation of resources, remains challenging in mobile ad hoc environments [9].

The willingness of service providers to participate in sensing compositions depends on local resources, their current load, and individual objectives. Probing techniques [5,11], primarily used for optimal provider selection, enable service providers to actively commit to a composition rather than allocating them without their confirmation and risking their later refusal. Alternatively, proactive discovery solutions explicitly ask for participation [14]. Both such approaches, however, require additional communication on top of establishing knowledge about available services. Binding and releasing services just in time is crucial to maintain good service availability and to mitigate overload situations. Composition approaches that start the execution only after the selection is complete [15][14][5][16], allocate all required services even those that do not get executed due to conditional paths or premature termination. They block providers unnecessarily and possibly make them unavailable for other compositions. Timeouts are a way to release those providers [5] but are difficult to configure. Further, the time a service must be blocked depends on its position in the request because it must wait on its predecessors to execute. Execution solutions that allow for interleaving selection and execution [17][18], as proposed in this work, do not target mobile ad hoc environments and the need for reducing communication over an unreliable network.

3 On-Demand and Just in Time Service Composition

Mobile ad hoc networks come with their own set of challenges, namely unreliable communication links, autonomous providers, and loalised service bottlenecks. Wireless communication in mobile environments is prone to interference, collision, and topology changes that can cause the composition to fail. Reducing network traffic is key to utilise scarce bandwidth effectively. Further, potential providers may hesitate to announce their services if they have little knowledge



Merge service: S_G
Synchronisation partners: S_E, S_F
Service router for sync partners: S_A, S_B, S_C, S_D

Fig. 2. Composition request with parallel execution paths

about the composition and the extent to which they will be involved. One could argue that service providers are still in full control after they have announced themselves because they can always refuse to deliver their services or simply drop out. A composition, however, will notice such unexpected behaviour only after it invokes the service and used up scarce bandwidth. In addition, compositions seek providers that are in close proximity to satisfy locality needs of their clients. Where there is high demand in one location, however, compositions compete for the same services because the resource-constraints of mobile devices limit the number of requests they can process simultaneously. A complex service request can be modelled as an abstract directed graph that defines the nature and order of required services. A request for in-network aggregation may contain parallel execution paths that deliver data from different sensors and merge for aggregation before the result reaches the client c (cp. Figure 2). For the request to become executable, suitable service providers must be discovered and selected.

3.1 Design Decisions

The service composition remains more flexible towards changes if it postpones the discovery and allocation of service providers to the time when the composite is actually invoked because then it has the most up to date view of available services. Further, due to the lack of a centralised composition entity with global system view, we choose a decentralised approach similar to a product line in which service providers receive a composite description, execute part of it, and forward it to the next provider. In the proposed composition protocol, assigned service providers act in a decentralised interleaved manner: They search, allocate, and invoke their successors after they executed their own service[1]. Such hop-by-hop processing pursues only paths that are actually required and reduces the time a provider is blocked. The composition request contains the entire composite including its current status to facilitate a provider's decision whether to block local resources and to announce its participation. Despite its high effort for finding distant services, we choose request-based service discovery because it creates network traffic only on demand, localises the use of bandwidth to the

[1] Typically, all providers get allocated before the execution of the composite starts.

area where the request is issued, and permits providers to voice their commitment selectively per composition request. Blocking local resources already when responding to a discovery request avoids overload but at the same time locks more providers than needed because only one of them will get the service assigned. For this reason, all known candidates receive a release message as soon as an allocation decision has been made to free them just in time for other requests. In wireless multihop networks, messages rely on being forwarded until they reach their destination and can be received by anyone who is in range of the forwarding node. Service providers thus receive messages for which they may not be the primary addressee. Instead of dropping these messages unseen, they infer what progress a composition has made and whether they should act. Listening to by-passing traffic allows for proactive behaviour and complementary to targeted messaging may reduce the composite's communication overhead.

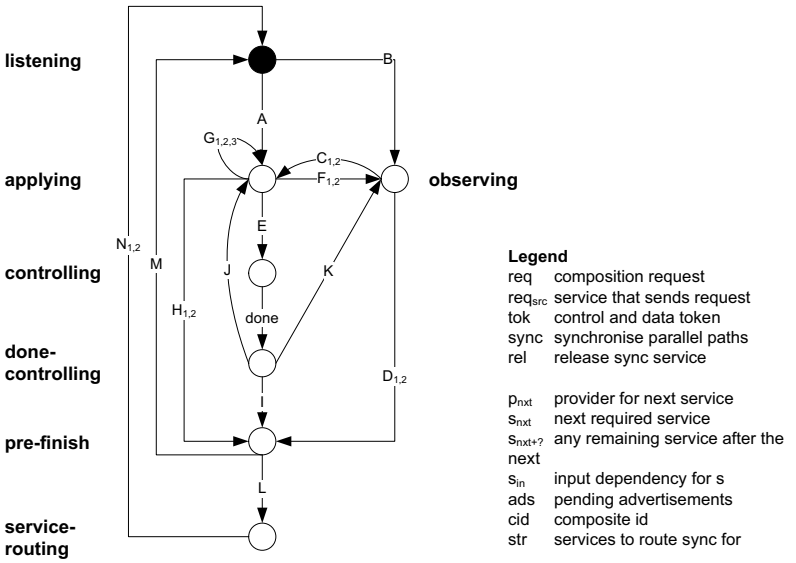
3.2 Protocol

The protocol for opportunistic service composition runs on each service provider and is modelled as a finite state machine (cp. Figure 3). The following description refers to providers that have only enough resources to handle one composition at a time. Transitions between protocol states occur upon the arrival of a composition message. A composite request contains the composite description and shows with the service that is required next which parts of the request still need to be allocated and executed. A token message transfers the composition control and indicates which service and corresponding provider execute next.

Each provider is initially in the listening state. If a listener receives a composition request and offers a service that is immediately required, it responds with an advertisement and changes its state to applying (A). Otherwise, if it provides any other required service, the provider remains silent and switches to observing (B). Checking for local objectives is not explicitly modelled, however, the protocol defines state transitions only on basis of a positive attitude towards participating in a composition.

An observing provider infers from an arriving token message and the request message heard earlier whether to announce its services to the token's primary recipient (C_1) or to transition to pre-finish (D_1). The choice depends on the conditions stated in Table 3b. Upon receiving a composition request it either keeps observing (D_2) or switches composites and applies for a different one (C_2).

A provider in applying state derives from a token whether it got selected to execute a service (E). If not, the applicant can decide to immediately apply for the next required service (G_1), observe the composite further to issue a new application at a later stage (F_1), or to finish its participation (H_1). In terms of a composite request, applicants generally do not respond except from two cases: First, if the request searches for a service for which the provider has already applied, then the provider resends the ad as the earlier one must have gotten lost (G_2). Second, the composition has made progress and the request reveals that all applications of the applicant are obsolete. then the applicant is free to apply (G_3), observe (F_2), or pre-finish (H_2).



(a)

State	Message	Condition	Transition
listening	req	offer(s_{nxt})	A
	req	$\neg$offer(s_{nxt}) $\wedge$ offer($s_{nxt+?}$)	B
observing	tok	$cid_{tok} == cid_{local} \wedge$ offer(s_{nxt})	C_1
	tok	$cid_{tok} == cid_{local} \wedge \neg$offer($s_{nxt}$) $\wedge \neg$offer($s_{nxt+?}$)	D_1
	req	$cid_{req} == cid_{local} \wedge \neg$offer($s_{nxt}$) $\wedge \neg$offer($s_{nxt+?}$)	D_2
	req	offer(s_{nxt})	C_2
applying	tok	$cid_{tok} == cid_{local} \wedge$ this$==p_{nxt}$	E
	tok	$cid_{tok} == cid_{local} \wedge \neg$(this$==p_{nxt}$) $\wedge$ ads$==\emptyset \wedge$ $\neg$offer(s_{nxt+1}) $\wedge$ offer($s_{nxt+1+?}$)	F_1
	tok	$cid_{tok} == cid_{local} \wedge \neg$(this$==p_{nxt}$) $\wedge$ ads$==\emptyset \wedge$ offer(s_{nxt+1}) $\wedge$ inrange(p_{nxt})	G_1
	tok	$cid_{tok} == cid_{local} \wedge \neg$(this$==p_{nxt}$) $\wedge$ ads$==\emptyset \wedge$ $\neg$offer($s_{nxt+?}$)	H_1
	req	$cid_{tok} == cid_{local} \wedge \exists s : req_{src} == s_{in} \wedge$ s $\in$ ads	G_2
	req	ads$==\emptyset \wedge$ offer(s_{nxt})	G_3
	req	ads$==\emptyset \wedge \neg$offer(s_{nxt}) $\wedge$ offer($s_{nxt+?}$)	F_2
	req	ads$==\emptyset \wedge \neg$offer(s_{nxt}) $\wedge \neg$offer($s_{nxt+?}$)	H_2
done-controlling		$\neg$offer(s_{nxt}) $\wedge \neg$offer($s_{nxt+?}$)	I
		$\neg$offer(s_{nxt}) $\wedge$ offer(s_{nxt+1})	J
		$\neg$offer(s_{nxt}) $\wedge \neg$offer(s_{nxt+1}) $\wedge$ offer($s_{nxt+1+?}$)	K
pre-finish		$\neg$(str$==\emptyset$)	L
		str$==\emptyset$	M
service-routing	sync	str$==\emptyset$	N_1
	rel	str$==\emptyset$	N_2

(b)

Fig. 3. Protocol for on-demand and just in time service composition

In the controlling state the service provider executes its assigned service, searches for a successor and when both these tasks are completed hands over the composition control to its successor. Controllers generate composition requests, token messages, and release redundant resources. Once done with controlling, the provider knows the composite status first hand and has again the choice to apply (J), observe (K), or pre-finish (I).

Pre-finishing determines whether the provider is a service router. The role of a service router was introduced by our synchronisation protocol [4] to handle requests with parallel execution paths. Service routers use the structure of the composite and the observed allocation history to route synchronisation messages between mutually unknown synchronisation partners that must agree on a common provider for the merge service. For example in Figure 2, the provider for S_B is a service router for S_E and S_F to route their synchronisation messages and help them decide on a common provider for S_G. If the provider is not a service router, it returns to listening (M), otherwise to service-routing (L). A provider exits service routing on receiving a release or sync message that indicates all routing responsibilities have been completed (N_1, N_2).

In the protocol, observers, applicants, and service routers block local resources and must be released in time to increase service availability. Observers may unblock any time to respond to more urgent service demand in different composite requests (C_2). Applicants are free as soon as they have no pending ads and are not assigned to provide a service. The list of pending ads is updated each time a token message or composite request signals that the relevant service has been allocated elsewhere. The allocator sends the token to all known advertisers. If the unreliable network looses ads, the allocator is not aware of these blocked candidates and does not release them. By-passing composition messages may indicate the progress of the composition. If this fails, too, a timeout is the last resort to unblock these resources. Service routers analyse all synchronisation messages which they forward to update and reduce their services-to-route list. Synchronisation messages, however, may find shortcuts in the network and do not follow the exact composite structure to reach their destination. In this case, a service router sends a release message back in its branch to enable other service routers to unblock.

3.3 Implementation

We implemented the opportunistic composition protocol on the discrete event simulator Jist/SWANS Ulm edition[2] by extending `ducks.driver.GenericNode` with a composite initiator and a composite provider. Both these types of composition entities require messages to be observable such that they receive and analyse any message issued in their transmission range. Lower layer network protocols typically discard messages if the node is not the primary addressee and let only broadcasts pass. Ordinary broadcasts, however, are not recovered and solely used imply high composite failure in unreliable networks. As a solution the

[2] `http://vanet.info/jist-swans/download.html`

protocol sends all composition messages as *directed broadcasts*, i.e., broadcasts with a primary addressee who acknowledges the receipt to ensure recovery and which can be received without acknowledgement by any other node in range. Further, a composition message may have multiple recipients. For example, a token message must be delivered to the next controller and all redundant applicants to unblock them. Instead of sending multiple messages, the directed broadcast includes a list of primary addressees and specifies for each destination the next hop. This way a single message is sufficient to inform all next hops. The management of directed broadcasts currently resides on the simulator's application layer to ensure the network layer passes messages directly to the MAC layer without consulting the routing protocols implemented in the simulator.

4 Simulation-Based Evaluation

The following study explores the impact of on-demand and just in time composition on the success and communication effort of complex service requests in multi-client scenarios.

4.1 Experimental Setup

The Jist/SWANS simulator configuration (Table 1) reflects a medium dense network of walking service providers. This setup was chosen based on the analysis in [7] to ensure zero percent partition and short routes (three hops on average) to reduce the effect of the particular routing protocol. The ratio of clients determines the number of composite requests that, scattered in the network, are issued at the same time. The study uses CiAN*, an adapted version of CiAN [15], as baseline that dispatches all messages with AODV [13] instead of the original publish-subscribe mechanism because of its otherwise higher message overhead. Just like the original, CiAN* implements ad-based discovery, client-controlled all-at-once service allocation, and decentralised service invocation. Opportunistic service composition implements on-demand request-based discovery with situational advertising, interleaved hop-by-hop allocation and invocation of services, and directed broadcasting. Any options for route repair and collision avoidance are disabled to analyse the failure probability of service composition prior to any recovery measures. Both composition models use the same non-standard description of the test composite in Figure 2 and select most recent neighbours first for required services. For each client ratio and composition model the simulation repeates 100 times with different randomised initial settings. Each such setting is used in both models. We test the most dynamic behaviour in which each required service must be bound to a unique provider. Autonomous node movement and the restriction to one composite at a time per node provides enough variance such that the willingness to participate is not modelled and assumed to be positive at all times.

Table 1. Simulation configuration

General		**Random**	
Simulation type	terminating	Node movement	Random Waypoint
Field ($m \times m$)	500×500	Node speed (m/s)	1-2
Radio range (m)	100	Service exec time (ms)	10-100
Nodes in total	150	**Controlled**	
MAC	802.11 non-promisc	Clients from total (%)	1, 5, 10, 15, 20

4.2 Results

Composition Success. A composition request is successful if all required services have executed and the client receives the final composition result.The left graph in figure 4 depicts the number of successful clients relative to the total number of clients in the simulation study. Opportunistic service composition achieves a higher success ratio than CiAN*, most notably, for five percent clients where the difference is 45 percent points. The success ratio degrades in both approaches as the demand for complex services increases. Opportunistic service composition grows more prone to service allocation failure because an increasing number of clients competes for the same service providers and soon reaches the maximum search radius unable to find unblocked resources. Blocking service routers contribute substantially to provider depletion in a search area. In the test composite (cp. Figure 2), service routers S_A, S_B, S_C, and S_D do not respond to other composites until they can be sure the synchronisation partners S_E and S_F know each other. In the worst case (when no by-passing traffic accidentally introduces parallel branch providers) all these six services block. CiAN* fails mainly due to provider overload since clients allocate services in isolation and providers handle only one composite at a time, silently dropping any other allocation. A service provider may resolve this conflict but only after it has detected the overload. Recovery strategies are out of scope and not implemented in either of the two approaches to study the original composition behaviour. Network failure due to collision and stale routes occur in both approaches, but at a lower percentage than their main failure source.

Response Time. The response time is the delay on the client from sending the first allocation or composition request to receiving the final result. The right graph in figure 4 depicts the average response time of compositions that completed successfully. CiAN* takes around four seconds to respond and is less affected by number of clients. The opportunistic approach responds in scenarios with one and five percent clients quicker than CiAN* because services and routing information is available in close proximity. Completing a composite quickly, releases bound service providers early and make them available for other requests. With an increase in clients, however, opportunistic composition faces service discovery delays as well as the same route finding problems as CiAN* such that both approaches respond after similar time.

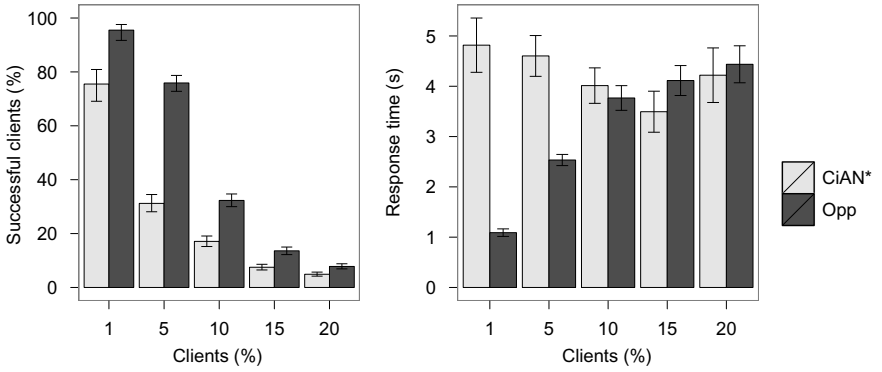


Fig. 4. Composition success and response time

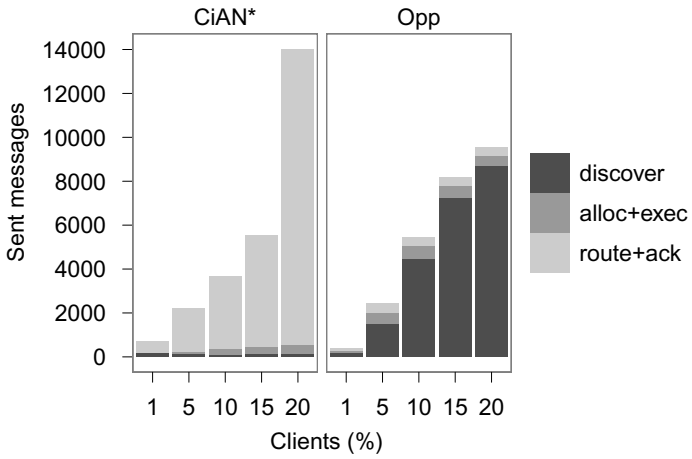


Fig. 5. Communication effort

Communication Effort. Composition and routing messages sent from the MAC layer determine the communication effort for handling complex service requests. These messages are counted until each of the started compositions either completes successfully or stops due to failure. Figure 5 shows the network load regardless of the final composition status. Starting with fewer messages, the opportunistic approach gradually exceeds CiAN*'s communication effort mainly because it completes more composite requests successfully than the baseline for which message counting stops early after a failure. In particular, the test cases with one and five percent clients indicate that opportunistic composition reduces network traffic. In contrast to CiAN*, routing layer messages in the opportunistic approach are rare demonstrating the benefit of directed broadcast. Service discovery messages dominate the message exchange and their increasing proportion with the increasing number of clients shows again the difficulty of finding unblocked service providers.

5 Discussion

The simulation study shows that opportunistic service composition is more successful at less or similar communication effort and delay than a conventional composition solution. In particular, for moderate client densities it outperforms the baseline. Stress testing the approach with many clients that issue the exact same request and the exact same time, the requests compete for the same service providers and extent their search for alternatives. Thereby they consume more time and bandwidth, and eventually fail due to provider depletion in the search area, making recovery strategies (that have not been considered in this study) indispensable for actual deployments.

The strength of opportunistic service composition is at the same time its weakness: Immediately executing partial composites reduces the impact of system changes but also leads to inconsistency if required services are not available in the network. The notion of standard services hosted on every mobile device would ease the problem as the existence of an ad hoc network would then imply the availability of such services. In comparison, composition approaches like the baseline that verify availability in advance, may fail nonetheless as the dynamics of the environment render earlier verifications invalid.

Opportunistic service composition is a promising approach toward in-network aggregation for participatory sensing as it overall shows an advantage over conventional composition approaches when it comes to managing complex service requests in unreliable networks. This work focused on how service composition could be used to coordinate the collective effort of autonomous mobile nodes to achieve a complex task. It did not touch on the technical realisation of ad hoc communication and radio broadcast with state-of-the-art smartphones as well as related security and routing issues. Future work will also have to investigate how the composition effort in an ad hoc network compares to the overhead of uploading sensor data individually. Each device may experience delays, losses, and energy costs when transferring small data packets to the distant cloud or to consecutive service providers.

References

1. Fasolo, E., Rossi, M., Widmer, J., Zorzi, M.: In-network aggregation techniques for wireless sensor networks: a survey. IEEE Wireless Communications 14(2), 70–87 (2007)
2. Ferreira, H., Duarte, S., Preguiça, N., Navalho, D.: Scalable data processing for community sensing applications. In: Puiatti, A., Gu, T. (eds.) MobiQuitous 2011. LNICST, vol. 104, pp. 75–87. Springer, Heidelberg (2012)
3. Groba, C., Clarke, S.: Opportunistic composition of sequentially-connected services in mobile computing environments. In: International Conference on Web Services (ICWS), pp. 17–24. IEEE (2011)
4. Groba, C., Clarke, S.: Synchronising service compositions in dynamic ad hoc environments. In: International Conference on Mobile Services, pp. 56–63. IEEE (2012)

5. Gu, X., Nahrstedt, K., Yu, B.: Spidernet: An integrated peer-to-peer service composition framework. In: International Symposium on High performance Distributed Computing (HPDC), pp. 110–119. IEEE (2004)
6. Kamra, A., Misra, V., Rubenstein, D.: Counttorrent: ubiquitous access to query aggregates in dynamic and mobile sensor networks. In: International Conference on Embedded Networked Sensor Systems (SenSys), pp. 43–57. ACM (2007)
7. Kurkowski, S., Camp, T., Navidi, W.: Two standards for rigorous manet routing protocol evaluation. In: International Conference on Mobile Adhoc and Sensor Systems (MASS), pp. 256–266. IEEE (October 2006)
8. Lane, N., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., Campbell, A.: A survey of mobile phone sensing. IEEE Communications Magazine 48(9), 140–150 (2010)
9. Loke, S.W.: Supporting ubiquitous sensor-cloudlets and context-cloudlets: Programming compositions of context-aware systems for mobile users. Future Generation Computer Systems 28(4), 619–632 (2012)
10. Lu, H., Lane, N.D., Eisenman, S.B., Campbell, A.T.: Fast track article: Bubblesensing: Binding sensing tasks to the physical world. Pervasive and Mobile Computing 6(1), 58–71 (2010)
11. Park, E., Shin, H.: Recon gurable service composition and categorization for power-aware mobile computing. Transactions on Parallel and Distributed Systems 19(11), 1553–1564 (2008)
12. Perez, A., Labrador, M., Barbeau, S.: G-sense: a scalable architecture for global sensing and monitoring. IEEE Network 24(4), 57–64 (2010)
13. Perkins, C.E., Belding-Royer, E.M., Das, S.: Ad hoc on-demand distance vector (aodv) routing, `http://www.ietf.org/rfc/rfc3561.txt`
14. Prinz, V., Fuchs, F., Ruppel, P., Gerdes, C., Southall, A.: Adaptive and fault-tolerant service composition in peer-to-peer systems. In: Meier, R., Terzis, S. (eds.) DAIS 2008. LNCS, vol. 5053, pp. 30–43. Springer, Heidelberg (2008)
15. Sen, R., Roman, G.-C., Gill, C.: CiAN: A workflow engine for manets. In: Lea, D., Zavattaro, G. (eds.) COORDINATION 2008. LNCS, vol. 5052, pp. 280–295. Springer, Heidelberg (2008)
16. Wang, M., Li, B., Li, Z.: sflow: Towards resource-efficient and agile service federation in service overlay networks. In: International Conference on Distributed Computing Systems (ICDCS), pp. 628–635. IEEE (2004)
17. Yu, W.: Decentralized orchestration of BPEL processes with execution consistency. In: Li, Q., Feng, L., Pei, J., Wang, S.X., Zhou, X., Zhu, Q.-M. (eds.) APWeb/WAIM 2009. LNCS, vol. 5446, pp. 665–670. Springer, Heidelberg (2009)
18. Zaplata, S., Hamann, K., Kottke, K., Lamersdorf, W.: Flexible execution of distributed business processes based on process instance migration. Journal of Systems Integration 1(3), 3–16 (2010)