[[1]](#footnote-1)

Localized Algorithm for Service Composition in Pervasive Computing Environments

(by Joanna, Jiannong Cao, Ma Chao, Steven Lai, Kong Jun Jun )

***Abstract—* In this research we significantly decrease the communication cost during service composition in highly dynamic large-scale environments, while satisfying the composition locality. We achieve this by avoiding collecting global information in service composition process, which is common for existing works. Moreover, in our approach, nodes and links are mapped in subsequent stage. We propose a novel localized service composition algorithm (LaSeC), in which the nodes collaborate only with their local neighbors to compose the service specified by a user. To reduce redundant broadcast, we modify MPR based technique in two respects. First, we propose a new forwarding node selection rule to prioritize nodes based on their possible contribution to the final solution. Second, we allow re-use of selected forwarding nodes by other nodes broadcasting for composition of the same composed service. We demonstrate feasibility of our approach in prototype implementation and conduct extensive simulations to study the performance of the proposed LaSeC compared with existing pull-based centralized techniques. Simulation results show that our technique decreases message cost and response delay. Based on collected results we discuss performance of our approach in terms of scalability, composition locality and coping with dynamic environments.**

Keywords. pervasive computing environment, service composition, localized algorithms.

# INTRODUCTION

Real world objects in the peoples environments are enhanced with additional capabilities. For example objects at home can serve as interfaces for controlling robots [1], [2], mirrors can help with shopping [3] and wearable computing enables trend of computing on the body [4], [5], [6]. Pervasive computing environment consist of physical devices, distributed in the physical environment and equipped with computing and communicating capabilities, which are implemented as services and exposed to the potential requestors. How to compose these services to better fulfill user request becomes a complex problem. Figure 1 shows service composition application in pervasive computing environment. Consider that user specifies the requirement of watching the movie, for which following services must be satisfied: file with the movie, movie player, device to output sound of the movie and device to output its display. During runtime usually multiple instances of requested functionalities can be found. And apart from the services specified in the requirement, other services exist in the environment as well.

The system model of the proposed approach is illustrated in Figure 2. Communication layer consists of a finite collection of service providers, which can communicate with each other to satisfy user specified requests. Request layer shows the user-specified service description, which can be accessed by the service providers in the communication layer. We assume that all service providers in the environment can access to a whiteboard showing the user-specified service description, which is a set of component functionalities together with composition relationships. Our problem is to construct the composite service in a localized manner, through collaboration between service providers. We transform this problem to subgraph isomorphism problem and prove that this problem is NP-complete. We also define a localized version of the subgraph isomorphism problem.

|  |  |
| --- | --- |
|  |  |
| 1. Service composition application. | 1. Localized service composition system |

Unlike previous algorithms that rely on global knowledge, in our algorithm each device only maintains local state information about its physical neighbors. By enabling localized interactions we achieve the solution which is highly scalable. We reduce redundant broadcast by employing forwarding node selection techniques. Moreover, in our approach, nodes and links are mapped in subsequent stage, which allows us to cope with dynamicity of the environments. Another important feature of our approach is high degree of composition locality. We validate our approach both through prototype implementation and simulation results.

The rest of the paper is organized as follows. In Section 2, we present the related work. In Section 3, we describe our problem formulation. Next, in Section 4, we describe the design of the proposed algorithms and its prototype implementation in Section 5. Finally, we describe the evaluation of the proposed approach in Section 6, and conclude the paper in Section 7.

# Related Work

To the best of our knowledge, thus far there is no research work that achieves the same objectives as ours. However, many related studies do exist.

In the area of service composition in pervasive computing many works [7-12] propose centralized solutions, where user specified service description is submitted to the centralized directory and the directory makes a decision on the list of services that should be returned to the requestors. This is impractical in dynamic environments, where it is required to frequently update the central entities, resulting in large system overhead. Moreover, central entity becomes a single point of failure. Some works exploiting distributed approach have already been proposed. For example, in [15] a hierarchical directory has been proposed. The resource-poor devices depend on resource-rich devices to support service discovery and composition. However, this approach is not fully decentralized. There exist nodes that perform the task of service discovery and composition for other nodes. There are also fully distributed approaches to service composition. In [13] a composite service is represented as a task-graph and sub trees of the graph are computed in a distributed manner. However this approach assumes that service requestor is one of the services and relies on this node to coordinate service composition. The same domain of the problem was studied in [14]. It is different from [13] in terms of the way of electing coordinator. In these approaches, although service discovery is performed in the distributed manner, composition process still relies on a coordinator assigned for performing task of combining and invoking services.

Our problem may be regarded as similar to embedding problem, which is concerned with mapping a new virtual network (VN), with constraints on the virtual nodes and links, on to specific physical nodes and links in the substrate network [16], [17], [18]. Composed service graph can be seen as virtual network, where virtual nodes are required functionalities and virtual links are required relationships. Communication network in pervasive computing can be seen as substrate network with substrate nodes being service providers and substrate links – communication links between service providers. Therefore virtual network embedding can be seen as service composition overlay (mapping a service composition graph, with constraints on the nodes and links, on to specific physical nodes and links in the communication network). There are two categories of algorithms solving this problem. In two stage approach [16], [17], there are separate stages for node and link assignment. In single stage [18], nodes and links are mapped in subsequent stage. Single stage approach achieves faster recovery from a bad mapping decision – this is because algorithm does not need to re-map all nodes and links from the beginning, it can backtrack to the last valid decision. In all of surveyed works the VN mapping algorithm is carried out in a centralized manner and a central entity is responsible for receiving VN requests from users and for selecting and assigning a set of virtual nodes to a set of substrate nodes. We have already pointed out the limitations of centralized approach presented on scalability, efficiency and resiliency, especially in a highly dynamic and changing environment (e.g. node/link failures, node mobility, etc). Although none of these works could be used directly for our approach, we were inspired by the idea of single stage mapping.

Works in both areas of research, service composition and virtual embedding share common feature, which is collecting global information. This is unsuitable for highly dynamic and large scale environments. To remedy this problem, we have opted to adopt a localized approach. Localized algorithms have been applied in wireless and mobile ad hoc networks to enable such tasks as target monitoring [19], service discovery[20], and topology management [21]. We have learned that localized approach can significantly decrease the communication cost as well as response delay. However, we are unaware of any localized algorithms for service composition in pervasive computing environments.

Localized approach to service composition requires nodes to broadcast their information in order to find collaborating nodes. There have been localized algorithms proposed for broadcasting in ad hoc networks, in which a small set of forward nodes is selected to reduce broadcast redundancy problem. The forward node set forms a connected dominating set (CDS). Summary of such methods is presented in [22]. In neighbor-designating methods the forwarding status of each node is determined by its neighbors. The source node selects a subset of its 1-hop neighbors as forward nodes to cover its 2-hop neighbors. Each forward node in turn designates its own forward node list. These methods consider broadcasting from one source to whole network. Requirements on localized service composition are different. Information needs to reach only selected nodes, and there may be many sources of the information. Moreover, additional, application oriented requirements have to be considered.

# Problem Specification

## Problem Illustration

Our system consists of a finite collection P of nodes communicating by means of messages. An entity corresponds to a service provider in pervasive computing environment. Nodes communicate with other nodes to achieve a common goal, which is to satisfy given request. Node communicates by transmitting messages to and receiving messages from other nodes. Communication links are bidirectional. In our system model node can communicate directly only with a subset of the other nodes. We denote by the set of entities to which node x can transmit and receive a message directly; we shall call such set of nodes the physical neighbors of node x.

The physical neighborhood relationship defines a graph, where  is the set of vertices and is the set of edges; the vertices correspond to entities, and  if and only if the node y is a physical neighbor of the node x. The graph  describes the communication topology of the underlying network. In summary, node can only receive messages from and send messages to its neighbors and it can distinguish between its neighbors.

The concept of k-neigbourhood is shown in Figure 3. The k-neighborhood relationship defines a graph, where  is the set of vertices at distance *k* or less from vi.

|  |
| --- |
|  |
| 1. Definition of neighbourhood. |

Let’s consider example service request shown in Figure 4. It can be represented by a graph: **

Furthermore, let’s consider environment to which such request was submitted.

**1-neighbourhood case**. The example environment for 1-neigbourhood is shown in Figure 5. Nodes are connected by communication links and for simplicity each node is described by service type, denoted with a letter of alphabet. We notice that in this environment, request from Figure 4 can be satisfied by four different compositions. For example, the bottom left service is very dense, all nodes are one hop away from node providing service i. In contrast, in next composed service, two F and H nodes are 6 hops away from each other. However both of these composed services are valid solutions to the request from Figure 4.

|  |  |
| --- | --- |
|  |  |
| 1. Example service request | 1. Service environment with possible service graphs |

**k-neighbourhood case.** In this case, request can be satisfied by nodes that are in at most k-hop distance from each other. The example environment for 3-neigbourhood is shown in Figure 6. Request from Figure 4 can be satisfied partially by nodes G and C, which are in 3-hop distance from each other.

|  |
| --- |
|  |
| 1. k-neighbourhood service graphs. |

## Problem Description

We study the problem of a localized approach to service composition. Given a set of nodes each providing certain services, the nodes are interconnected and can communicate with each other via wireless network, assume a user specified requested composite service with specified types of requested services, relationships between them and requested quality. Our problem is to design a localized protocol for the nodes to collaborate in the composition process such that the communication cost is minimised and the composed service satisfies the specified quality.

We model this problem as follows. First we model the communication topology as a directed attributed communication graph (CG) with nodes representing service providers and edges representing communication links between them, and attributes represent service type and its quality. Service providers know only k-neighbourhood of the directed attributed graph. Requested composite service is modelled as a directed attributed service composition graph (SCG), where nodes represent requested services, edges represent relationships between services and attributes represent functionality of the service and minimum requested quality. We can see that in order to satisfy the request, we need to map each vertex and edge in service composition graph to exactly one vertex and edge in communication graph. Such a problem is known as a sub graph isomorphism problem. Since our problem is localized, we define a localized version of sub graph isomorphism problem:

For a node *i* in the network, given k-neighbourhood of the node *i* in communication graph CG and service composition graph SCG our problem is to design a localized protocol for each node *i* CG to determine a set of outgoing and incoming edges from/to *i* to be maintained such that:



* there is a one to one mapping between edges maintained by *i* and edges from/to the node *j* in SCG, where *j* has the same functionality as *i* and quality of *i* is higher than *j*,
* the number of edges maintained by *i* is the same as the number of edges from/to the node *j* in SCG, where *j* has the same functionality as *i* and quality of *i* is higher than *j*.

## Problem Formulation

Let be a directed attributed graph with the vertex set and the edge set and let , be a directed attributed graph with the vertex set and the edge set . Let be an attribute set. Let .

For each node determine a set of edges to be maintained such that

and

## Complexity Analysis

Given a SCP instance *I*, find an overlay *O* of the communication topology graph *GP*, compatible with the service request graph *GR*. This problem is NP-complete. We prove its NP-completeness by showing that it is a generalization of the sub-graph isomorphism problem (SGI), a NP-complete problem [23], which is defined as follows:

INSTANCE: Graphs *G=(V1, E1), H=(V2,E2)*

QUESTION: Does G contain a sub-graph isomorphic to H?

By setting *GP=G, GR=H*, SGI can be reduced to SCP, while there still exist differences between SGI and SCP:

* *G* and *H* are undirected, unlabeled and non-attributed graphs, while *GP* and *GR* are directed, labeled, and attributed graphs;
* SGI aims to find a sub-graph of *G*, while SCP aims to find a overlay of *GP*;

For the first different point, according to [24][25], the variant of SCI for directed graphs is still NP-complete. [26] has also proved that labeled SGI problem contains unlabeled SGI problem by setting the label set size to be one, and consequently, the labeled SGI problem is still a NP-complete problem. Similarly, we can prove that non-attributed graph is an extreme case of attributed graph, where the attribute size is zero. That is to say, a directed, labeled, and attributed SGI problem is still NP-complete. Furthermore, we can show that a sub-graph is a special case of a overlay by restricting that the edges of the overlay *Eo ⊆ Er*. In summary, the SCP problem is a generalization of the SGI problem, and thereby it is NP-complete.

# localized Service Composition Algorithm

In this section, we describe our LaSeC algorithm that enables the devices to cooperatively construct the requested service. When constructing the desired overlay graph, device candidates decide with whom to cooperate only based on the information of the devices within its k-neighborhood. We propose a protocol for the nodes to collaborate in service composition process. The main operation the protocol is broadcast when some nodes try to find other nodes to cooperate in service composition process. To reduce redundant broadcast, we modify MPR based technique [27] in two respects. First, we propose a new forwarding node selection rule to prioritize nodes based on their possible contribution to the final solution. Second, we allow re-use of selected forwarding nodes by other nodes broadcasting for the collaboration in the same user request.

## Forwarding Node Selection

While executing our algorithm, each node *i* exchanges messages listed in Table 1 and maintains necessary data structures listed in Table 2.

**Table 1.** Message Types in Forwarding Node Selection

|  |  |
| --- | --- |
| **Message** | **Purpose** |
| *Req(id, service type, if FNS)* | Node *i* requests from 1-hop neighbors their id and service type and if they were chosen as forwarding node by other broadcasting nodes |
| *Req(1-hop\_n)* | Node *i* requests from 1-hop neighbors their set of all 1-hop neighbors |

Localized algorithm for Forwarding Node Selection is summarized in Algorithm 1. At the beginning node *i* collects id and neighborhood information from its 1-hop neighbors. Upon receiving information about 1-hop neighbors of its 1-hop neighbors, node *i* builds information about its 2-hop neighborhood. The information includes service type of neighbors and if they already have been nominated by their neighbors to be in their Forwarding Node Set (*FNS*). Node *i* selects a subset *N* of 1-hop neighbors that can provide any service type in requested SCG and puts this nodes to *FNS* and marks which 2-hop neighbors are connected. Next, from remaining unconnected 2-hop neighbors, node *i* selects a subset *N* of 2-hop neighbors that are connected with node *i* only through one 1-hop neighbor.

**Table 2.** Data Structures in Forwarding Node Selection

|  |  |
| --- | --- |
| **Variable** | **Meaning** |
| *id\_i* | Unique id if node *i* |
| *id\_i(1-hop\_n)* | Set of id’s of 1-hop neighbors of node i |
| *1-hop\_n* | Set of all 1-hop neighbors of node *i* |
| *2-hop\_n* | Set of all 2-hop neighbors of node *i* |
| *N* | Set of 2-hop neighbors that are connected with node *i* only through one 1-hop neighbor. |
| *Not\_Covered* | Set of 2-hop neighbors that are not connected with node *i* through node from *FNS* |
| *FNS* | Set of 1-hop neighbors that connect node *i* with its all 2-hop neighbors |

If node *k* is the only 1-hop neighbor that connects node *i* with the 2-hop neighbor from set *N*, node *i* puts node *k* in the *FNS*. Next, node *i* examines if all 2-hop neighbors are connected. Node *i* selects the 1-hop neighbor that connects the largest number of two-hop neighbors that are not yet connected. If necessary, node *i* repeats this step until all 2-hop neighbors are connected. When node *i* broadcast messages to its 1-hop neighbors, it also informs those that are in *FNS*.

|  |
| --- |
| **Algorithm 1**  Forwarding Node Selection |
| ***INPUT:*** *1-hop\_n*,  *id\_i* ,  ***OUTPUT:*** *FNS*  // The code executed by broadcasting node *i*  (1) Send *Req(id, service type, if FNS)* to *1-hop\_n ;*  (2)Send *Req(1-hop\_n)* to *1-hop\_n;*  (3) if (node *k* is the 1-hop neighbor that provides service type specified in SCG) *FNS*←node *k*;  (4) *Not\_Covered* ← 2-hop neighbors that are not connected with node *i* through node from *FNS*  (5) *N* ← nodes from *Not\_Covered* that are connected with node *i* only through one 1-hop neighbor;  (6) if (node *k* is the only 1-hop neighbor that connects node *i* with the 2-hop neighbor from set *N*) *FNS*←node *k*;  (6) remove *N* from *Not\_Covered*  (7) repeat if (node *k* connects largest number of nodes from *Not\_Covered*) *FNS*←node *k*; until *Not\_Covered* = 0)  } |

## Cooperative Overlay Construction

In this section, we describe our protocol that enables the devices to cooperatively construct the requested service.

### Illustration of the Algorithm

Cooperative Overlay Construction consists of following components: (i) local request creation, (ii) local request unfolding/folding. Local request creation facilitates which services are required to be obtained from which node. Local request unfolding/folding is based on rules indicating which nodes are best to contact next and which nodes should not be considered further, since information that they have is irrelevant for the final solution.

***Component I: Local request creation.*** Each node receives the user request and identifies if it can provide any service specified in the request with the specified quality. If the answer is positive, the node becomes candidate node and identifies what local requests it should create. Each node creates local requests only for its children. Children of a given node are such nodes which can provide service type that in the request consumes the output of the service type provided by the given node. Example is shown in Figure 7. For such a requirement, each candidate node which provides service A has one child (B) and needs to create one local request for service type B. In the same example each candidate node that provides service B has two children, D and E, and consequently need to create two local requests, one for each of its children. Candidate nodes providing services E and F have no children for this request; therefore they will create no local requests.

***Component II: Local request unfolding/folding.*** Nodes who have no parent (such as node A in the example shown in Figure 7) unfold the local request to their 1 hop neighbors. At the same time broadcasting nodes use our Forwarding Node Selection algorithm to inform which 1-hop neighbors should re-broadcast the local request. This broadcast is repeated in k-neighborhood of initiating node. If local request arrives to the node which is intended receiver, the receiver determines if it should unfold its own local request or fold the received request.

To understand the concept of local request unfolding/folding let’s consider request A-D-E-F and environment in Figure 8.

|  |  |
| --- | --- |
|  |  |
| 1. Parent/child relationship in the request | 1. Service providers in the environment. |

For example, node A80 will unfold the local request for D in three directions: D50, D30 and D07. However D50 will never fold this local request, because it cannot satisfy finding E in its neighborhood. D07 will also not fold this local request, due to absence of I in its neighborhood. As for D30 it will unfold its local request for I and E. E11 and E05 in the neighborhood of D30 will unfold local request for F. Next, F30 will fold to E11 and E05, upon which E11 and E05 will fold to D30 and eventually D30 will fold to A80. In each step the unfolding/folding node will perform local computation to determine when and in which direction to unfold/fold.

Local request unfolding/folding involves following steps:

*Step 1:* wait for local requests from all parents – If a node has multiple parents (such as node providing service F for the example request shown in Figure 3) it needs to wait until it receives at least one local request from each of its parents. It is possible that the node will receive more than one local request from the same parent type.

*Step 2:* unfold own local requests to all children – When a node received local request from all parents, it puts them away and sends out its own local request created in phase I. If a node has no children (such as node F and E) it goes directly to step 4.

*Step 3:* wait for response from all children – After sending local requests a node waits for response to each local request. While waiting for the response from children, new local requests from parents can arrive, which node accepts and put away with other local requests.

*Step 4:* fold to all parents – When response to local request was received from all children (or node has no children), node retrieves local requests that were put away, merges them with responses and sends to its all parents. At least one response should be received from each child, and at least one response should be sent to each parent. A node can use candidate ranking mechanism to limit the number of responses.

### Formal Description of the Algorithm

While executing our algorithm, each node *i* maintains necessary information about its state in the data structures listed in Table 3 and may exchange messages listed in Table 4.

**Table 3.** Data Structures in Decentralized Service Composition

|  |  |
| --- | --- |
| **Variable** | **Meaning** |
| *status* | Status of node *i* = {*WAITING, INITIATOR, CANDIDATE, ROUTING, DONE*} |
| *out\_edges* | Service types that succeed the type of node *i* in the requested service GR |
| *in\_edges* | Service types that precede the type of node *i* in the requested service GR |
| *sender* | Node ID from which *i* received a message |

**Table 4.** Message Types in Decentralized Service Composition

|  |  |
| --- | --- |
| **Message** | **Purpose** |
| *local\_request\_unfold* | Candidate node *i* announces local request to nodes which service type succeeds the type of node *i* in the requested service. |
| *local\_request\_fold* | Response to the local request. |

Unlike previous algorithms that rely on global knowledge, in our algorithm each device only maintains local state information about its physical neighbors. In particular, devices in our algorithm exchange a set of local requests locally to find the desired overlay graph. Importantly, our algorithm can minimize the number of involved nodes, and response time. It is summarized in Algorithm 1.

***Finding Candidate Solution.*** The purpose of this round is to find candidate overlay graphs by localized interactions between devices:

*Phase I.* At the beginning of Round I, the requested composite service  is submitted to each device in. Upon receiving, a device that can provide one of the requested functionalities identifies itself as a candidate. Devices performing functionality that has no preceding nodes in the requested graph identify themselves as initiators. Each initiator generates a message called local\_request\_unfold. Each local\_request\_unfold is set with its designated service type. Designated service type is set according to the node that succeeds the type of node i in the requested service.

|  |
| --- |
| **Algorithm 1 LaSeC:** Localized Service Composition |
| ***INPUT:*** , ,  ***OUTPUT:***  Find Candidate Solutions()  {  // The code executed by each node *i*  (0) *status* ←*WAITING*;  (1) Wait until received GR,  (2) if (*in\_edges* of *i* in GR is empty) *status* ←*INITATOR*;  (3) if (*in\_edgesr* of *i* in GR is not empty) *status* ←*CANDIDATE*;  /\* The code executed by each *INITIATOR* node \*/  (5) Send (*local\_request\_unfold*) to *out\_edges*  /\* The code executed by each *CANDIDATE* node \*/  (6) Wait until received *(local\_request\_unfold*);  (7) *in\_edges*= *in\_edges* –(*sender)*;  (8) if (*in\_edges* = 0) {Generate (*local\_request\_unfold*); send (*local\_request\_unfold*) to *out\_edges*; };  (9) else store (*local\_request\_unfold*);  (10) Wait until received *(local\_request\_fold*);  (11) if (merged *local\_request\_fold* = GR) become *DONE;*  (12) else send (*local\_request\_fold*) to *in\_edges;*  } |

*Phase II.* In the second phase, the local\_request\_unfold will be processed by physical neighbors. At each neighbor, an local\_request\_unfold collects local state information from the visited device. When an local\_request\_unfold arrives at a device that is a candidate for the designated service type of that local\_request\_unfold, the local\_request\_unfold is stopped and merged with local\_request\_unfold from other incoming edges. When there was at least one local\_request\_unfold coming from all incoming\_edges, the device generates new local\_request\_unfold, with new designated service. When local\_request\_unfold arrives at the service type that has no outgoing edge in the requested graph, it starts phase III.

*Phase III.* In the third phase, the device without outgoing edges returns local\_request\_fold to the received local\_request\_unfold to its initiators. When any device receives the local\_request\_fold from different devices, it merges them. When merging results with a complete candidate request graph phase III is completed.

# Prototype Implementation

## System Architecture

We develop a prototype for our service composition approach to study its performance in the real environments. As shown in Figure 9-11, we used 40 MicaZ nodes with a MIB600 gateway for the hardware and deployed them in an indoor environment.

|  |  |  |
| --- | --- | --- |
|  | IMG_1035 | PHM037 |
| 1. MicaZ node | 1. MIB600 Gateway | 1. Part of the Testbed |

The system architecture for the prototype is shown in Figure 12. In order to support flexible interaction between the application programmer and our service composition middleware, our prototype mainly consists of two layers. On the top layer, the application user will provide an XML file which defines service composition. Then the file will be parsed and packetized for dissemination into the pervasive computing environment. On the bottom layer, after the devices receive the service specification they will start executing the service composition algorithm and notify the user when the composition process is complete.

## Experiments

We conducted experiments based on our prototype. We pre-defined a number of services for each sensor node in our testbed.

|  |
| --- |
|  |
| 1. Composition locality. |

# Performance Evaluation

## Simulation

We have carried out extensive simulations to evaluate the performance of our algorithm. Moreover, to show the advantage of our algorithm (LASEC), we have also implement a pull-based model of the centralized algorithm for service composition (CEN) [11], and have compared its results with ours. Below, we give a brief description of the implemented algorithms.

***LASeC. Localized service composition algorithm*:** Our localized service composition algorithm has been applied to the network with implemented forwarding node selection protocol.

***CEN. A centralized service composition algorithm:*** In this algorithm, we provide a dedicated service directory. Initially there are no services registered with the directory. When a user specifies a composite service, it is submitted to the directory in the form of a graph. Upon receiving the request, the directory pulls from the environment the information related to service types specified in composition request. The pull operation is supported by broadcast.

In our simulations, we consider a grid network of N nodes which are uniformly deployed. The total number of nodes in the network is varied to examine the effect of system scale on the performance. Also we varied the complexity of the request. A service type is randomly assigned to each node. In each network we set the total number of service type so that the density of service types (ds=N/ns) is comparable. The simulation parameters are listed in Table 5.

**Table5.** Simulation Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameters** | **Values** | | |
| Number of nodes, (N) | 64 | 81 | 100 |
| Number of service types (ns) | 10 | 13 | 17 |
| Service density | 6, 7.5, 9, 12.5 | | |
| Request complexity | 3, 4, 5, 6 | | |
| Territory scale (m2) | 200 | | |
| Transmission radio range (m) | 40 | | |

We evaluated the performance of the algorithms using the following metrics:

*Message overhead:*  the number of messages generated by the composition process. We do not consider injecting request as the part of composition process. We start measuring messages that are generated after the request was specified.

*Response delay:* the interval between the time a request is received by the system and the time corresponding reply is returned by the system.

Below we present our simulation results with analysis. We have simulated our LASEC algorithm as well as CEN approach. Each point is obtained by averaging 100 runs.

## Performance Analysis

### Scalability

*Scalability with regards to the network size*

The number of messages generated by our service composition protocol according to the size of the network is presented in Figure 13. It considers the number of messages necessary to find first response. Our LASEC service composition mechanisms require less messages than CEN for finding first result. This is because CEN needs to pull the request from the network, which includes generating and forwarding messages for sending the request and receiving the response, while LASEC takes advantage of localized interactions. Moreover, we observe that LASEC scales very well with the growing size of the network while CEN imposes larger cost when the network grows. Similar results can be observed for the time necessary to find first answer, according to the size of the network, which is presented in Figure 16. As can be seen from the results, our LASEC algorithm performs better than CEN. Since in CEN messages are flooded in the network, the collision effect causes increase in delay. LASEC experiences lower rate of collisions, due to forwarding node selection. Similar as with message cost, LASEC scales beter than CEN.

*Scalability with regards to the request complexity*

Figure 14 gives the number of messages generated by our service composition protocol according to the size of the request. It shows that number of messages increases with the complexity of the request. LASEC algorithm takes advantage of the localized approach and due to forwarding the messages only on the SCB. LASEC avoids flooding the network with messages. However, we observe, that along with the growth of request complexity, LASEC generates more messages, while CEN is more scalable. Since CEN collects global information, it collects the same amount of information for different request complexity, while LASEC cost depends on the complexity result. Figure 17 gives the time necessary to find first answer, according to the size of the request. Our LASEC service composition mechanism performs better than CEN for finding first result. This is possible due to localized nature of discovery of services to be composed. If service providers are close to each other they can compose service faster that it takes for the central entity to collect information from the environment. Moreover, both CEN and LASEC perform similar in terms of scalability. CEN requires longer time for more complex requests. Although the time for collecting the information from the environment is similar for different requests, however CEN will need additional time for evaluating the information and making decision.

Figures 15 and 18 show the time and message overhead necessary to find optimum answer, according to the service density in the environment. In this simulations we have explored localized characteristics of our approach and allowed nodes to forward message to 2 hop neighbours only. We observe that response delay as well as number of messages increase with service density. Our LASEC service composition mechanism has smaller overhead than CEN for finding optimum result.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1. Impact of network size on message overhead. | 1. Impact of request size on message overhead. | 1. Impact of service density on message overhead. |
|  |  |  |
| 1. Impact of network size on response delay. | 1. Impact of request complexity on response delay. | 1. Impact of service density on response delay. |

### Composition locality

Figure 19 shows the result of our simulation. For the request of 6 types of services two possible compositions are shown.

|  |
| --- |
|  |
| 1. Composition locality. |

In pervasive computing environments composition locality refers to the distance between the services used in composition and it is required that it should be as small as possible. Composition locality is an important factor when the tasks include services which need to interact with each other and if they are embedded on different nodes in the network. To minimize such cost of composition all atomic services in the composed service need to be located as close to each other as possible. Through simulation we have observed that LASEC tends to select compositions with high degree of composition locality.

### Coping with Environment Dynamicity

Dynamicity of the environments is one of the major challenges in pervasive computing. Service providers may leave environments due to mobility, unexpected power off or failures. In a pervasive computing environment, even when once the initial composition is identified and a service provisioning is established, the dynamicity of the environment may require change of the composed solution. For example, the user with his mobile phone may walk out of a room while user’s mobile phone was providing a service in collaboration with other devices in this room. Or, user’s mobile phone may be no longer available due to power limitations. In scenarios of one of atomic services in a composed service becoming suddenly unavailable, the challenge is to recompose the service as quickly as possible considering new state of the environment.

With our localized approach, it is possible to ensure that the request can be recomposed with minimal overhead. In extreme cases of dynamic (mostly theoretical) the composed service may need to be completely recomposed. Typically, however, it is sufficient to recompose the service partially, maintaining those services identified previously which are still available and replacing only the service that left the environment.

The result of a successful composition is a set of service providers S, where each service provider is responsible to provide atomic functionality requested by user in the request graph GR. Therefore, each node in the graph GR corresponds to one service provider in S. If one of the service providers from S becomes unavailable, then corresponding node in the GR must be recomposed. Based on our localized approach, services responsible for new composition, are only immediate neighbors of the unavailable service provider: that means service providers that corresponds to the nodes in GR which share links with the node corresponding to unavailable service provider. This concept is shown in Figure 22 and Figure 23. In original composition following set of service providers has been identified {F1, E1, D1, I1, A1, B1, C1, H1}. Due to sudden unavailability of service provider B1, the service cannot be fully composed. However we see, that only path between service providers A1 and C1 has been affected – therefore only this path requires repair. Service provider A1 will initiate our localized algorithm, including all nodes apart from that of service of type D}. Recomposition completes after finding such service provider of type B, which is both neighbor of service provider A1 and C1. This is because remaining part of composition does not need to be recomposed. Service provider C1 maintains links leading to successful composition. Through simulations, pictured in Figure 20 and 21, we have observed, that recomposition cost is stable and similar to that of composing simple requests.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 1. Message overhead due to recomposition | 1. Response delay due to recomposition. | 1. Service provider B1 becomes unavailable | 1. Service provider B2 serves as a replacement |

# Conclusion

This paper presents a localized approach to service composition. Service providers identify what service type they need to interlock with through their outputs and then search for appropriate service providers with matching input types. Our idea is to grow sections of composed service from available services. In the initial stages of search it is only some pieces that are used to construct service sections. At later stages these sections are joined together with other sections to gradually construct the completed solution. For the purpose of reducing redundant broadcast we employ forwarding node selection techniques. We have implemented prototype of LaSeC and performed simulations to evaluate its performance. Our evaluation has shown that compared with existing pull-based centralized techniques our algorithm is more scalable, achieves higher degree of composition locality and copes better with dynamic environments.

# References

1. Sakamoto, D., Honda, K., Inami, M., Igarashi, T.: Sketch and Run: A Stroke-based Interface for Home Robots. In: 27th International Conference on Human Factors in Computing Systems, pp. 197–200 (2009)
2. Shengdong Zhao, Koichi Nakamura, Kentaro Ishii, and Takeo Igarashi. 2009. Magic cards: a paper tag interface for implicit robot control. In Proceedings of the 27th international conference on Human factors in computing systems (CHI '09).
3. Maurice Chu, Brinda Dalal, Alan Walendowski, and Bo Begole. 2010. Countertop responsive mirror: supporting physical retail shopping for sellers, buyers and companions. In Proceedings of the 28th international conference on Human factors in computing systems (CHI '10).
4. Grace NGAI, Stephen C.F. Chan, Vincent T.Y. Ng, Joey C.Y. Cheung, Sam S.S. Choy, Winnie W.Y. Lau and Jason T.P. Tse. "i\*CATch: A Scalable, Plug-n-Play Wearable Computing Framework for Novices and Children". Proceedings of the Twenty-Seventh Annual SIGCHI Conference on Human Factors in Computing Systems (CHI 2010)
5. Lee, S. and Starner, T. 2010. BuzzWear: alert perception in wearable tactile displays on the wrist. In Proceedings of the 28th international Conference on Human Factors in Computing Systems (Atlanta, Georgia, USA, April 10 - 15, 2010). CHI '10.
6. Grace NGAI, Stephen C.F. Chan, Winnie W.Y. Lau and Joey C.Y. Cheung. "The TeeBoard: an Education-Friendly Construction Platform for E-Textiles and Wearable Computing" Proceedings of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems (CHI 2009) Boston, USA: April 2009
7. H. Pourreza and P. Graham. On the fly service composition for local interaction environments. In IEEE International Conference on Pervasive Computing and Communications Workshops, page 393. IEEE Computer Society, 2006.
8. S. B. Mokhtar, N. Georgantas, and V. Issarny. Cocoa: Conversation-based service composition in pervasive computing environments. Proceedings of the IEEE International Conference on Pervasive Services, 2006.
9. Z. Song, Y. Labrou, and R. Masuoka. Dynamic service discovery and management in task computing. Mobiquitous, 00:310-318, 2004.
10. M. Vallee, F. Ramparany, and L. Vercouter. Flexible composition of smart device services. In International Conference on Pervasive Systems and Computing (PSC05), pages 165-171. CSREA Press, 2005.
11. A.C. Huang and P. Steenkiste. A Flexible Architecture for Wide-Area Service Discovery, The Third IEEE Conference on Open Architectures and Network Programming (OPENARCH 2000), March 26-27, 2000.
12. A. Bottaro, J. Bourcier, C. Escofier, and P. Lalanda. Autonomic context-aware service composition. 2nd IEEE International Conference on Pervasive Services, 2007.
13. T. D.C. Little, B. Prithwish, K. Wang. A novel approach for execution of distributed tasks on mobile ad hoc networks. In IEEE WCNC. Orlando. Florida, 2002.
14. D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha. Service composition for mobile environments. Journal on Mobile Networking and Applications, Special Issue on Mobile Services, 10(4):435-451, January 2005.
15. S. Kalasapur, M. Kumar, B. Z. Shirazi. Dynamic Service Composition in Pervasive Computing. IEEE Transaction on Parallel and Distributed Systems, 2007.
16. M. Yu, Y. Yi, J. Rexford and M. Chiang, Rethinking virtual network embedding: substrate support for path splitting and migration, ACM SIGCOMM Computer Communication Review 38 (2) (2008), pp. 17–29.
17. Chowdhury, N.M.M.K., Rahman, M.R., Boutaba, R.: Virtual network embedding with coordinated node and link mapping. In: IEEE INFOCOM (2009)
18. Lischka, J., Karl, H.: A virtual network mapping algorithm based on subgraph isomorphism detection. In: Proceedings of ACM SIGCOMM VISA (2009)
19. GruiaCalinescu, A fast localized algorithm for scheduling sensors, Journal of Parallel and Distributed Computing, v.66 n.4, p.507-514, April 2006
20. XuLi, Nicola Santoro, Ivan Stojmenovic, "Localized Distance-Sensitive Service Discovery in Wireless Sensor and Actor Networks," IEEE Transactions on Computers, pp. 1275-1288, September, 2009
21. Wen-Zhan Song, Yu Wang, Xiang-Yang Li, and OphirFrieder, “Localized algorithms for energy efficient topology in wireless ad hoc networks," in 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc2004),
22. W. Lou and J. Wu, Localized Broadcasting in Mobile Ad Hoc Networks Using Neighbor Designation. in Mobile Computing Handbook, CRC Press, 2005.
23. H. Papadimitriou, and K. Steiglitz. Combinatorial optimization: algorithms and complexity. Dover Publication, 1998
24. M. R. Gary, and D. S. Johnson. Computer and Interactability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York, 1979.
25. Z. Ling. An Effective Approach for Solving Subgraph Isomorphism Problem. In Proceedings of the IASTED International Conference, 1996.
26. Santoro, N. Design and Analysis of Distributed Algorithms. John Wiley & Sons, Inc., Chichester, 2007.
27. A. Qayyum, L. Viennot, and A. Laouiti, “Multipoint Relaying for Flooding Broadcast Message in Mobile Wireless Networks,” Proc. Hawaii Int’l Conf. System Sciences, p. 298, Jan. 2002.

1. [↑](#footnote-ref-1)