LASEC: A Localized Approach to Service Composition in Pervasive Computing Environments

Joanna Siebert, Jiannong Cao, Senior Member, IEEE, Steven Lai, Peng Guo, and Weiping Zhu

Abstract—Pervasive computing environments (PvCE) are embedded with interconnected smart devices which provide users with services desired. To meet requirements of users, smart devices with different kinds of functions may need to be associated together to provide the service described in the user requirement, which is called service composition. As the service composition environment may be dynamic and large scale, centralized service composition algorithm is usually inefficient due to message cost. On the other hand, a decentralized approach, which employs pre-determined coordinators to search and compose service, may have high cost as well, if the coordinators are far from those devices that should be composed. In this paper, we discuss a localized approach for service composition based on the UIO model we have proposed earlier. UIO (Ubiquitous Interacting Object) is an abstraction of physical devices in PvCE with ability to find and collaborate with other devices through exposing their capabilities as services. In our localized service composition algorithm (LASEC) UIOs collaborate with each other in a bottom-up, localized manner to compose required service without requiring global knowledge. The main challenge in LASEC is how to constraint the blind and random composition among the UIOs that could not form the requested service so as to guarantee all the services specified in the request will be provided. To solve this problem, we propose a novel mechanism called Alien-information-based Acknowledging (A-Ack), in which a UIO decide on collaborating with another UIO only after obtaining some additional information from the collaboration candidate. Specifically this information refers to ability of given UIO to compose another part of the service. Proposed LASEC is message-efficient and quality-guaranteed. Extensive simulations of LASEC as well as existing decentralized and pull-based centralized algorithms have been conducted. The results show the relatively low communication cost and composition time of LASEC. Moreover, we demonstrate feasibility of our approach with a prototype implementation.

Index Terms—Pervasive computing environment, Ubiquitous interacting objects, Service Composition, Localized algorithm.

1 Introduction

The widespread deployment of pervasive computing devices is transforming the physical world into a pervasive computing environment (PvCE). Devices with sensing, memory, computing and communication capabilities are immersed into our living environments. 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]. PvCE becomes a medium that provides user with all the functionality he needs to satisfy his requirements. It is built on physical objects in the real world with embedded computing devices interconnected according to an underlying communication network.

We define the physical objects having intelligent abilities of pervasive computing as Ubiquitous Interacting Objects (UIO) [28]. UIOs are supposed to be able to intelligently collaborate with each other. For this purpose objects can be augmented with sensing, processing actuating or communication devices. These capabilities are implemented as services and exposed to the potential requestors. Through services users can interact with PvCE.

UIOs can provide hardware or software services. Examples of software services can be weather forecast, stock quotes, and language translation. Services provided by the hardware UIOs can be for example controlling a temperature in the room or printing a document. UIOs that expose their functionalities in form of services are called service providers, and UIOs that need some functionality are called service requestors. Requestors of services can be users, other objects or middleware that facilitates interaction in PvCE. Service providers in PvCE are characterized by their heterogeneity and dynamicity.

Service composition mechanisms are necessary when no single service provider can provide a requested service and several service providers have to collaborate. Fig. 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. According to requirements, we need to select one instance for each service type.

Several approaches for selecting service providers

J. Siebert, J. Cao, S. Lai, P. Guo and W. Zhu are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. E-mail: csjsiebert@comp.polyu.edu.hk.

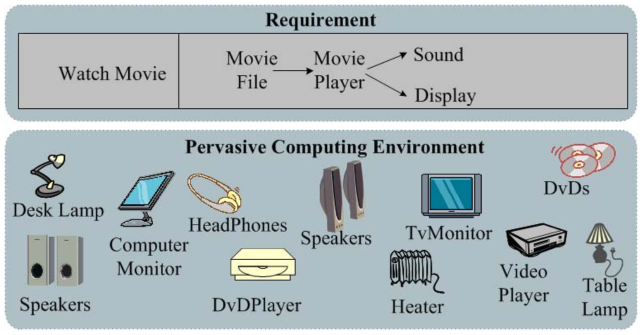


Fig. 1 Service composition application. For specified required functions multiple devices in the environment are available.

have been proposed, which are mainly partitioned into two categories, i.e., the centralized approaches and the decentralized approaches. In centralized approaches, a coordinator is usually assumed in the environment with knowing the global knowledge of all the service providers and maintaining a centralized directory about the providers. To avoid central point of failure as well as to better support dynamic environments, some decentralized approaches have been proposed, which do not require a determined and fixed coordinator in the environment before the service composition request. Instead, a temporary coordinator or a predetermined device is assigned for the current service composition, which will collect the current information of service providers in the environment to compose the service. Though information is obtained in a decentralized way, the service composition is still performed by the assigned coordinator, similarly to the centralized approach. This will lead to unpredictable message cost and composition delay since inappropriate coordinator selection may incur large searching range, thus leading to high message cost.

In this paper, we study the localized approach to composition problem, where service providers can automatically collaborate with each others without any global knowledge. We first formulate the problem as the subgraph isomorphism problem which has been proved to be NP-complete. To find an efficient heuristic approach for the problem, we model the process of localized composition in a graph and propose a message-efficient localized service composition (LASEC) algorithm. The key idea of LASEC is to facilitate service providers to locally but orderly collaborate with each other according to the topology feature in the required services' relationship. To implement it, a novel *Alien-information-based Acknowledging A-Ack* mechanism is specially designed. We prove that each service provider with LASEC sends no more than two messages during the composition, and the service providers can compose the highest-quality service when given the constraint that each provider sends message within its local k-hop area. In addition, if considering different service's weight in each composition step, the proposed LASEC can easily be combined with the decentralized approach in [29], which focuses on computing services' weight in the QoS of composition. Extensive simulations for LASEC as well as existing decentralized and pull-based centralized approaches have been conducted, showing the relatively low communication cost and composition time of LASEC. In addition, we demonstrate feasibility of our approach with a prototype implementation.

Contributions of the paper are summarized as follows.

- For the first time, we propose a localized service composition algorithm for service providers to collaboratively compose the required service.
- We design a novel mechanism called *A-Ack* for service providers to exchange messages, with which each service only needs to send no more than two messages during the composition.
- We give the upper bound of the running time of the proposed localized composition algorithm, and prove that service providers with the proposed algorithm can compose the highest-quality service when given a constraint that service providers send messages within their local k-hop area.
- We implement the proposed LASEC algorithm on a prototype system, demonstrating the effectiveness of the algorithm in practice.

The rest of the paper is organized as follows. In Section 2, the related works are presented. In Section 3, we describe the system model and formulate the problem. Then, we introduce the design of the proposed algorithm and its theoretical analysis. Performance evaluation of the proposed algorithm, including simulations and experiments, is made in Section 5, followed by the conclusions in Section 6.

2 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.

Different kinds of environments require different designs of service composition mechanisms. Many existing works exist for web services which are a method for utilizing services through world wide web [36]. In these works, service registry is an important component that allows service requestors to find service providers which previously published their information to it. Multimedia streaming is area that produced a number of distributed service composition approaches [29, 30, 31, 32, 33]. In distributed multimedia and stream processing domain, some multimedia content may be divisible into independently routable components, e.g. audio and video flows. As a result media content adaptation services may be linked in serial, parallel and hybrid configurations to form a directed, acyclic graph of composed services [34]. There are also works on service provisioning [39] and service composition [40] in opportunistic networks, which exploit an opportunistic contact between two mobile devices when they are within communication range of each other.

When designing mechanisms for service composition in PvCE, issues of scalability and dynamicity must be taken into consideration. Both, environment and users are impacted by the scale and dynamicity of PvCE. In the area of service composition in pervasive computing many works [7-12] propose centralized solutions. Works on automatic service composition in recent environments, such cloud computing, also often base their work on centralized architectures [38]. This category of works depends on the existence of a centralized directory of services. Namely, they rely on one or more central entities to maintain the global service information in order to make composition decisions. Service requestors submit requirements to the directory and the directory makes a decision on the list of services that should be returned to the requestors. This approach works well for environment that are well managed and relatively stable. However, the assumption on centralized control becomes impractical in scenarios with dynamic arrivals and departures of service providers, which requires frequent updates of the central entities, resulting in large system overhead. Moreover, relying on central entities to maintain global knowledge leads to inability of the system to serve the request when the central manager is compromised.

In order to address deficiencies of centralized approaches, some works exploiting distributed approaches have already been proposed. As the first step towards decentralization of service composition they introduce distributed directory of services. 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.

Next category of distributed service composition approaches removes the need for a service directory and provides a fully distributed search for needed services. In [13] a hierarchical task-graph based approach to do service composition in ad hoc environments has been proposed. A composite service is represented as a task-graph and sub trees of the graph are computed in a distributed manner. This approach assumes that service requestor is one of the services and relies on this node to coordinate service composition. The coordinator uses global search across the whole network to do composition. The same domain of the problem was studied in [14]. It is different from [13] in terms of the way of electing coordinator. For each composite request, a coordinator is selected from within a set of nodes. The service requestor delegates the responsibility of composition to the elected coordinator.

The common feature of existing distributed service composition approaches is that while they perform discovery of services in the distributed way, they still require coordinator to perform service composition. However the support for the UIO-based service composition in large scale and dynamic pervasive environments is inadequate. A large number of portable and embedded devices in these environments work collaboratively in a purely ad hoc manner and without any central control. This necessitates a localized interaction model which will achieve scalable service composition minimizing both message costs and time response. To achieve this goal, we prefer a coordinator-less based service composition solution to a coordinator based one as the former appears to be more scalable than the latter. Even using dynamic coordinators (new coordinator for each request) leads to to expensive message broadcasts which is not practical for pervasive computing environments with large number of dynamically changing devices.

There are also works proposed for efficient composition meeting specified QoS requirements. For example, [29, 30, 32, 37] proposed a QoS-aware service composition framework. While [32] focus is on QoS assurance and service composition is achieved in centralized way, [29 and 30] proposed a a fully decentralized service composition framework. It is constructed as a service overlay network layered on top of existing Internet infrastructure. In fact, this makes this approach not feasible to implement in pervasive computing environments, especially in real-time environments: network communication has no temporal guarantees; there is no defined communication with the OS; and its QoS monitoring proposal is based on sending periodical probe frames that could be a source of nondeterminism. Besides, the execution time of their online composition algorithm is not bounded. In [37] authors further differentiate between the QoS of services themselves and the network and they propose a network-aware approach that handles the QoS of services and the QoS of the network independently.

Works in service composition share common feature, which is a globalized nature of algorithms. In a globalized algorithm, at least one node needs to maintain global information about the current task and about network status. This is unsuitable for highly dynamic and large scale environments. Gathering such global information creates huge communication overhead, especially with frequent changes of topology. Moreover, global knowledge is not always necessary, since in service composition for PvCE, it is beneficial when the distance between the services used in composition is as small as possible. To remedy the problem of maintaning global infromation, we have opted to adopt a localized approach, which naturally tends to select compositions with high degree of composition locality.

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

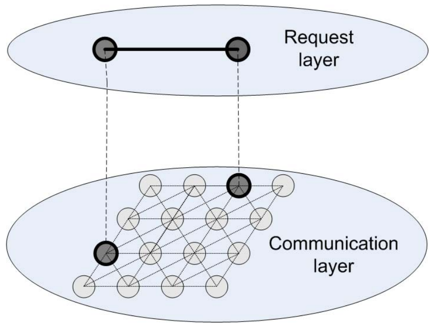


Fig. 2 Localized service composition system. Devices in the communication layer explore localized neighborhood to provide functions specified in the request layer.

of any localized algorithms for service composition in pervasive computing environments.

3 Preliminaries

3.1 System Model and Assumption

We suppose a set of UIOs providing certain services in the PcVE form a communication network, where the UIOs can communicate with each other through delivering messages. The service request is a specification of some types of services as well as their connection relationship. We illustrate the system with a two-layer model as shown in Fig. 2. The communication layer consists of the UIOs providing different services and the communication links among them. The request layer consists of the services required and the relationships between them. The UIOs with services involved in the request need to find each other in the communication layer according to the relationship specified in the request layer.

We model the communication layer with a graph $G_c = (V_c, E_c)$, where the vertices represent UIOs and the edges represent the communication links. Also, we model the request layer with a graph $G_s = (V_s, E_s)$, where each vertex represents a particular service and the edges represent the relationship between the services. For convenience, we call "vertex" as "node" hereafter, and call that nodes providing the same service are of the same type. In addition, since each UIO in the communication layer corresponds to one service in the request layer, we call the node in G_s representing the service provided by node i in G_c as node i's corresponding node in G_s. Note that, usually multiple nodes in G_c may have the same corresponding node in G_s.

A detailed example for G_c and G_s is shown in Fig. 3. From Fig. 3 (a), G_s is a tree graph and $V_s = \{A, B, C, D, E, F, G, H, I\}$. In G_c in Fig. 3(b)(c), the alphabet letter on each node denotes the service type provided by the node (i.e., UIO). For simplicity, we use Z to denote the nodes which have no corresponding node in G_s. In Fig. 3(b), the nodes make compositions just with their neighbors. After the composition process, one composition result meets the request, while others fail. However, if the nodes search other nodes in their 2-hops distance to collaborate for the composition, more composition results will meet the request, as shown in Fig. 3(c).

We make the following assumptions for the localized service composition problem.

- We consider the tree case of G_s, which is the usual case of service request in practice.
- We assume that each node's searching range, within which the node searches other nodes to compose with, is constrained to be no mare than k-hop distance.

For convenience, we call node i which is in k-hop distance of node j as j's k-neighbor.

3.2 Problem Formulation

Based on the model described in the previous section, we formulate the service composition problem (SCP) as follows: Given the service request G_s for each node in G_c, our target is to design a localized algorithm for nodes in G_c to search other nodes in their local k-hop distance in G_c to form a connected sub-graph G_r isomorphous to G_s with minimum message cost. Since we propose a localized approach to the above problem, we introduce following constraints: (i) no node in G_c knows or collects the global information about G_c during running of the algorithm, each node only maintains local state information about its k-hop neighbours; ii) no node in G_c knows or collects the full information about current status of G_r during the running of the algorithm. We call the isomorphous sub-graph as *overlay graph* hereafter.

The SCP problem is NP-complete and 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 $G = (V_1, E_1)$, $H = (V_2, E_2)$

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

By setting $G_c = G$, $G_s = H$, 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 G_c and G_s are directed, labeled, and attributed graphs;
- SGI aims to find a sub-graph of G, while SCP aims to find a overlay of G_c;

For the first different point, according to [24][25], the variant of SGI 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 an overlay. In summary, the SCP problem is a

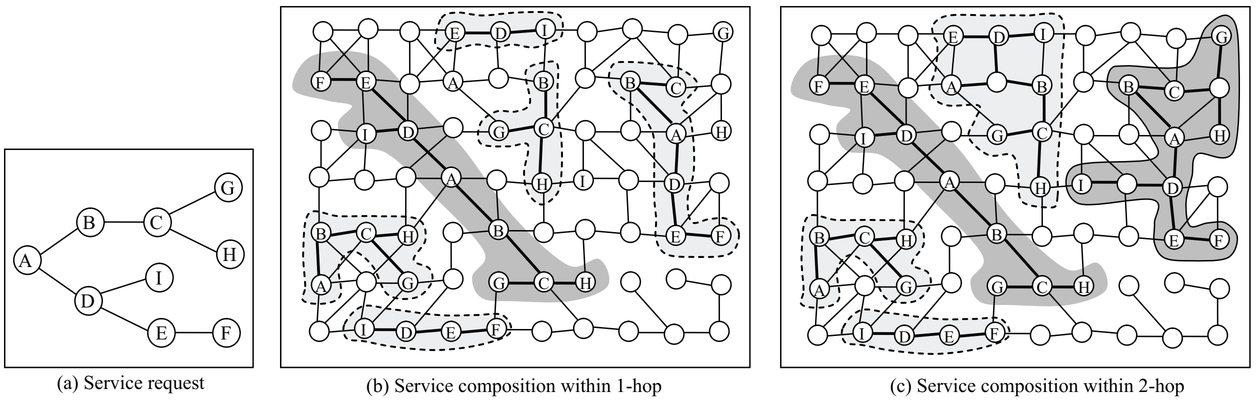

(a) Service request (b) Service composition within 1-hop (c) Service composition within 2-hop

Fig. 3 Example of localized service composition.

generalization of the SGI problem, and thereby it is NP-complete.

4 The Proposed LASEC Algorithm

Due to the NP-hardness of the problem SCP, a heuristic approach is needed to resolve it. In this section, we propose a localized service composition (LASEC) algorithm. The design of LASEC algorithm is inspired by the classic game of jigsaw puzzle. A jigsaw puzzle consists of a number of oddly shaped pieces with parts of picture on them. By fitting together interlocking pieces, a complete picture can be assembled. We expect to compose *sections* of *overlay graph* from available nodes and edges first and than join the *sections* into a completed *overlay graph*.

However, in jigsaw puzzle, the player usually has overall knowledge about the pieces, which helps to get to know which two pieces should be interlocked with each other. While, in localized composition, nodes composing the current *section* do not have the knowledge about other *sections* (or even about the current *section*). Hence, there are some challenges in the localized service composition:

- **Awareness**. How does a node know the current members of the *section* that it has participated in? How does a node know the *section* that it has participated in has already failed? Here, the failed *section* means the *section* that cannot further find other node in k-hop distance to compose with.
- **Judgement**. How does a node know which *section* is more possible to succeed the composition when the node is required to join? How to reduce or stop the useless composition which cannot be completed?
- **Assembly**. Since there is no coordinator in the network, how to guarantee the composition to converge into one integrated *overlay graph*. For instance, in Fig. 4, three nodes with different types in G_c are required to be composed according to G_s. When the nodes of type A request to compose with two nodes of type B and C, respectively, no composition is achieved although there are many possible *overlay graphs* in the network.

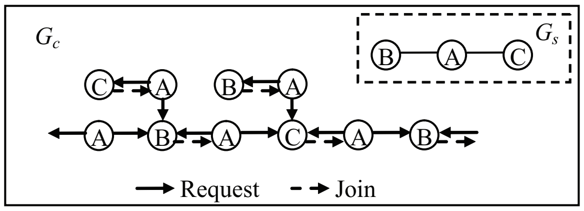


Fig. 4 The unassembled case in localized composition.

Considering these challenges, we design our proposed LASEC algorithm as follows.

4.1 Description of the Algorithm

For convenience, some notations are defined in Table 1. Meanwhile, the root in G_s in LASEC is the node to which the maximum hops in G_s is minimum. We apply the *Floyd-Warshall* algorithm [24] to find the root in our algorithm. The parent of a node in G_s is the neighboring node with smaller hop count to the root, and the child of a node in G_s is the neighboring node with larger hop count to the root. The leaf in G_s is the node without a child.

We suppose all nodes in G_c have obtained the information of G_s (i.e., the service request) before the service composition. This can be achieved simply by broadcasting it to the service providers. In addition, as described in [15], there is a literature proposing different methods for making request known to the service providers. For example request can be already embedded in the applications by the designers. Application can also have the ability to generate the requests based on the current context.

We define two types of messages as shown in Table 2. To guarantee that *active* nodes deliver messages within its k-hop distance, a time-to-live (TTL) variable is used in each message. In addition, we define two states for the nodes in G_c: $FREE$ (the default state) and $SELECTED$ (the state for nodes selected in an *overlay graph*).

We assume the quality of a composition is the sum of the qualities of all *active* nodes therein. The quality of each node may stand for the computation resource,

TABLE 1 Definitions of some notations.

Notation	Definition
Active node	The node in G_c with a corresponding node in G_s
Non-active node	The node in G_c without a corresponding node in G_s
Root node	The *active* node in G_c whose corresponding node in G_s is the root
Leaf node	The *active* node in G_c whose corresponding node in G_s is a leaf
G_s-neighbor	The *active* node in G_c with a neighboring corresponding node in G_s
G_s-parent	The *active* node in G_c with the parent corresponding node in G_s
G_s-child	The *active* node in G_c with the child corresponding node in G_s
G_s-descendant	The *active* node in G_c with the descendant corresponding node in G_s
$T(k)$	The upper bound of the time for message flooding in k-hop distance in G_c.
L	The maximum hops of nodes in G_s to the root.
m	The hops of the corresponding node in G_s to the root.
n	The maximum hops of the corresponding node in G_s to its descendant nodes.

TABLE 2 Definitions of messages.

Message type	Purpose of message	Information contained
Local_request message	Request *active* node to join in the composition	ID of the *active* node sending the message and the quality of current composition
Acknowledge message	Acknowledge the composition request	ID of the *active* node sending the message and the quality of current composition

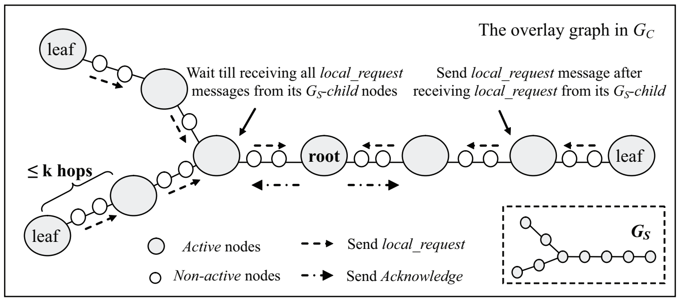

The overlay graph in G_C
leaf
leaf
leaf
root
Wait till receiving all *local_request* messages from its G_S-*child* nodes
Send *local_request* message after receiving *local_request* from its G_S-*child*
≤ k hops
Active nodes
Non-active nodes
Send *local_request*
Send *Acknowledge*
G_S

Fig. 5 The basic idea of LASEC.

bandwidth, etc. We define two basic composition operations in the localized composition.

- *Operation 1*: When an *active* node receives a *local_request* message (or an *acknowledge* message) sent from one of its G_s-neighbor nodes and decides to join in the composition, the *active* node will add its quality to the quality recorded in the message.
- *Operation 2*: when an *active* node receives each type of its G_s-child nodes' one *local_request* message, the *active* node will add all the qualities recorded in the messages as well as the quality of itself.

The basic idea of LASEC algorithm is illustrated in Fig. 5, where a novel mechanism called *Alien-information-based Acknowledging* (*A-Ack*) is designed. Different from the traditional acknowledgement mechanism where the receiver will acknowledge the sender if receiving a message, in *A-Ack* mechanism, the receiver will not acknowledge the sender until the receiver obtains some additional information (we call it alien information) from other nodes. This additional information relates to whether the receiver is able to collaborate with other necessary nodes or not. This mechanism allows us to guarantee the composition to converge into one integrated *overlay graph* without nodes having any global information. We introduce the *A-Ack* mechanism as follows.

- At the beginning, all *active* nodes identify whether they are *root* nodes, *leaf* nodes or others.
- Then, each *leaf* node in G_c initiates a *local_request* message and sends it to its G_s-parent nodes.
- When an *active* node receives the *local_request* message from its G_s-child, it will execute *operation 1*, and then send a new *local_request* message to its G_s-child nodes. Specially, if the *active* node's corresponding node in G_s has multiple child nodes, the *active* node will wait until it has received from each type of its G_s-child nodes' at least one *local_request* message. Then, the *active* node executes *operation 2* and sends a new *local_request* to its G_s-parent.
- When a *root* node has received from each type of its G_s-child nodes' at least one *local_request* message, the *overlay graph* is achieved. Then, the *root* node broadcasts an *acknowledge* message to all *active* nodes involved in the *overlay graph*.

Note that an *active* node may receive multiple *local_request* messages sent by different G_s-child nodes. For this case, the *active* node selects one message which is the message recording the highest quality of the current composition.

Based on the *A-Ack* mechanism, we design the LASEC algorithm for each node in G_c with **Algorithm** 1, 2, 3 and 4.

From the algorithm, we can see: 1) *active* nodes are composed orderly in G_c with the rule in the algorithm that *active* nodes send *local_request* messages only if they receive *local_request* from their G_s-child nodes, thus reducing many redundant and meaningless compositions in G_c; 2) each node records only the IDs of its selected G_s-neighbor nodes and the quality of current composition. The node does not need to forward all the IDs of nodes involved in the composition, hence making the message cost of the algorithm low.

Algorithm 1 The proposed LASEC algorithm

```
1:  $node\_state$=$FREE$
2:  Input: $G_s$
3:  Apply Floyd-Warshall algorithm
4:  Output: the root in $G_s$
5:  if the node's corresponding node $\in G_s$ then
6:      if the node's corresponding node in $G_s$ is the root then
7:          Execute Algorithm 2
8:      else if the node's corresponding node in $G_s$ has no child
        then
9:          Execute Algorithm 3
10:     else
11:         Execute Algorithm 4
12:     end if
13: else
14:     for $timer \leftarrow 0$ to $2L * T(k)$ do
15:         Keep receiving messages and forward the messages
16:     end for
17: end if
```

Algorithm 2 Algorithm for *root* nodes

```
1:  for $timer \leftarrow 0$ to $2L * T(k)$ do
2:      Keep receiving messages and help to forward messages
        whose destinations are not itself
3:  end for
4:  if receive each type of $G_s$-child's at least one local_request
    message then
5:      For each type of $G_s$-child, select one message recording
        the highest quality
6:      Broadcast an acknowledge message to all the $G_s$-child
        nodes recorded in the selected local_request messages
7:      $node\_state$=$SELECTED$
8:      OUTPUT: IDs of the $G_s$-child nodes recorded
9:  else
10:     $node\_state$=$FREE$
11: end if
```

4.2 Discussion of the Algorithm

The algorithm proposed in the previous section can guarantee the optimal service composition under the constraint that the communication between one node and its neighbours are within a number of k hops. However, if we relax this constraint the distributed way to achieve the global optimal may become quite costly. Here we discuss how to guide the service composition in such case. We suppose there is the knowledge about the range of benefit when a user requests a tree-based composite service. It means that each node and hence each tree rooted at a node in the service tree can have a benefit range estimate. Based on this knowledge and the processing condition in the real environment, when we face multiple combinations of nodes that all may be possible to be final service composition, we can select the ones whose sub-tree dominate the benefit of the whole service-tree and eliminate the ones whose benefit is definitely less than the others.

Algorithm 3 Algorithm for *leaf* nodes

```
1:  Send a local_request message to all its $G_s$-parent nodes
2:  for $timer \leftarrow 0$ to $(L + m) * T(k)$ do
3:      Keep receiving messages and help to forward messages
        whose destinations are not itself
4:  end for
5:  if receive acknowledge messages from its $G_s$-parent nodes
    then
6:      record IDs of the $G_s$-parent nodes sending the acknowl-
        edge messages
7:      $node\_state$=$SELECTED$
8:      OUTPUT: IDs of the $G_s$-parent nodes recorded
9:  else
10:     $node\_state$=$FREE$
11: end if
```

Algorithm 4 Algorithm for other *active* nodes

```
1:  for $timer \leftarrow 0$ to $n * T(k)$ do
2:      Keep receiving messages and help to forward messages
        whose destinations are not itself
3:  end for
4:  if receive each type of $G_s$-child's at least one local_request
    message then
5:      For each type of $G_s$-child, select one message recording
        the highest quality
6:      Execute operation 2
7:      Record the IDs of the $G_s$-child nodes in the selected
        local_request messages
8:      Send a new local_request message with the updated
        quality to all its $G_s$-parent nodes
9:  end if
10: for $timer \leftarrow 0$ to $(L - n + m) * T(k)$ do
11:     Keep receiving messages and help to forward messages
        whose destinations are not itself
12: end for
13: if receive acknowledge messages from its $G_s$-parent nodes
    then
14:     record IDs of the $G_s$-parent nodes sending the acknowl-
        edge messages
15:     $node\_state$=$SELECTED$
16:     OUTPUT: IDs of the $G_s$-parent nodes recorded and IDs
        of the $G_s$-child nodes recorded
17: else
18:     $node\_state$=$FREE$
19: end if
```

The determination of k is quite important in the approach. A larger value of k denotes more accuracy in the solution of service composition but larger message overhead as well. Therefore a trade-off should be made properly to set the value of k. The specific value of k is application-specific, mainly depending on the topology of the service graph and the specific requirements. Take a service that consists of two kinds of sub-services for example, the value of k can be set as the average number of hops needed to discover these two kinds of sub-services. Other information also includes the constraints specified by the user, and detailed specifications of services. For example, the required latency, the allowed largest energy consumption, bandwidth, etc. Such assumptions are quite common in the existing works. For example, in [15], a number of K possible service compositions are found to determine a desirable solution, and in [14], a maximum number of hops are specified to control the message overhead. The value of k is not fixed and different in different scenarios.

It is noted that the determination of k is a trade-off between accuracy and efficiency, which is quite difficult in some situations. Therefore some extensions may be needed considering some special cases. For the case illustrated by Fig.3 (b), if the value of k is set to 1, the proposed approach may fail to find the solution. A simple extension also can be used to solve the problem, to increase k if no solution is found. Another case is in Fig.3 (c). If k is set to 1, the composition on the left is always chosen and the composition on the right is always not chosen, which may incur unbalance workload. In this case, we can increase k to find more solutions if current service compositions cannot provide sufficient resources.

The other thing we need to emphasize is the position we initiate the service composition. A trade-off is necessary about the latency and the energy combustion. In our LASEC algorithm, the leaf node starts this composition processing and the upper-stream nodes are waiting for the results from leaf nodes. It is suitable for the applications which do not have strict real-time requirement but only concern about energy consumption or traffic cost. However, for the application which has strict real-time requirements, we need to set a proper location for the start of service composition. A node between the root and leaf node can serve this purpose. It explores several levels downstream (but not deep enough to leaf nodes) to collect the real service benefit information, estimate the possible benefit, and then at one side directly report to the root and on the other side continue the exploring. In this way, the latency requirement can be met and a probabilistic optimal result can be achieved.

4.3 Analysis of the Algorithm

In this section, we first prove the correctness of the proposed algorithm, including *safety*, *liveness* and *uniqueness*. Then, we analyze its performance in terms of message cost, running time and quality.

4.3.1 Correctness of the Algorithm

First, we give some lemmas and the proofs as follows. For convenience, we call the process that an *active* node receives messages, selects the messages and records the IDs of nodes sending the selected messages, as making composition.

Lemma 1. *An active node makes composition only with its G_s-neighbor nodes in G_c.*

Proof: According to LASEC algorithm, only two kinds of messages are delivered between *active* nodes in G_c to implement the composition. Meanwhile, the *active* nodes send *local_request* messages only to their G_s-parent nodes to request composition, and they send *acknowledge* messages only to their G_s-child nodes to acknowledge the composition. Therefore, the *active* nodes make the composition only with their G_s-neighbor nodes. □

Lemma 2. *An active node makes composition with only one of its G_s-child nodes of each type.*

Proof: According to LASEC algorithm, when an *active* node receives *local_request* messages from its G_s-child nodes, it always selects one message recording the highest quality in each type and records the IDs of the G_s-child nodes who send the selected messages. When the *active* node delivers *acknowledge* messages, the node sends the messages only to the recorded G_s-child nodes to acknowledge the composition. Therefore, the active node makes composition with only one of its G_s-child nodes in each type. □

Lemma 3. *An active node makes composition with its G_s-child nodes only if they have already made a composition, except for the case that G_s-child nodes are leaf nodes.*

Proof: According to LASEC algorithm, an *active* node will not send *local_request* messages unless it has received each type of its G_s-child's at least one *local_request* message, which means the G_s-child sending the message has already received *local_request* messages and made a composition unless the G_s-child is a *leaf* node. Therefore, the *active* node can make composition only if its G_s-child nodes have already made a composition except for the case of *leaf* nodes. □

With the Lemmas above, we prove the correctness of LASEC algorithm including *safety*, *liveness* and *uniqueness* as follows.

Theorem 1: *Safety: non-active nodes in G_c will not change its state to* SELECTED.

Proof: According to LASEC algorithm, a *non-active* node surely cannot change its state to *SELECTED* during the composition, no matter what kind of messages it receives. Therefore, LASEC guarantees the *safety* property. □

Theorem 2: *Liveness: the proposed LASEC algorithm can implement an overlay graph if there is one in G_c.*

Proof: According to LASEC algorithm, only *leaf* nodes initiate composition request. The *local_request* messages of *leaf* nodes are sent to only G_s-parent nodes during the composition. According to **lemma** 1 and 3, each *active* node makes composition only with its G_s-neighbor nodes, and there is a sequential rule for the composition that the *active* node makes composition only if its G_s-child nodes have made one. Since G_s is a tree graph, each node in G_s has only one path to the root and one parent in G_s. Hence, when an *active* node receives *local_request* message from one of its G_s-child node, it can ensure the composition of an overlay graph's branch that is formed by each type of the descendant *active* nodes of the *active* node. With the constraint of **lemma** 1 and 3, *active* nodes will be composed gradually and orderly from the *leaf* nodes till the *root* nodes without any local endless loop. And the *root* nodes will not broadcast the *acknowledge* message until it affirms the completeness of the composition of all its descendant *active* nodes, i.e., the *overlay graph*. Hence, the proposed LASEC algorithm can implement completed *overlay graph.* □

Theorem 3: *Uniqueness: When the algorithm is finished, each* active *node with the $SELECTED$ state records only one G_s-child node in each type of service.*

Proof: According to **lemma** 2, each *active* node makes composition with only one of its G_s-child nodes in each type. When the *root* node receives all the *local_request* messages, it also selects each type of its G_s-child nodes' one *local_request* messages and form the *overlay graph*. Then, the *root* node broadcasts the *acknowledge* message to its descendant *active* nodes uniquely recorded by their G_s-parent nodes, and each *active* node receiving *acknowledge* messages will change its state to be $SELECTED$. Therefore, each *active* node with the $SELECTED$ state records only one G_s-child node in each type of service. □

4.3.2 Performance of the Algorithm

We give the performance analysis of the LASEC algorithm in terms of message cost, running time and the quality of the *overlay graphs* achieved by LASEC as follows.

Theorem 4. *Each active node in G_c sends no more than two messages during the execution of the LASEC algorithm.*

Proof: According to LASEC algorithm, an *active* node will send a *local_request* message if it receives *local_request* messages from its G_s-child nodes during a certain time $n * T(k)$. And, the *active* node will send an *acknowledge* message if it receives *acknowledge* messages from its G_s-parent nodes during a certain time $(L - n + m) * T(k)$. Therefore, each active node in G_c will send no more than two messages during the execution of the LASEC algorithm. □

Theorem 5. *The upper bound of the running time of the LASEC algorithm is $2L * T(k)$.*

Proof: We discuss the upper bound of the running time for each kind of *active* nodes first. From **Algorithm** 2, it can be easily seen, the upper bound of *root* nodes' running time is $2L * T(k)$. From **Algorithm** 3, the upper bound of *root* nodes' running time is $(L + m) * T(k) < 2L * T(k)$. While, from **Algorithm** 4, the upper bound of other *active* nodes' running time is $n * T(k) + (L - n + m) * T(k) = (L + m) * T(k) < 2L * T(k)$. Since all nodes are assumed to execute the LASEC algorithm upon obtaining the information of G_s simultaneously, the upper bound of the running time of the LASEC algorithm is $2L * T(k)$. □

Theorem 6. *The* root *nodes in the LASEC algorithm can obtain the* overlay graph *with the highest quality, when given the constraint that* active *nodes send messages within local k-hop area.*

Proof: According to the LASEC algorithm, an *active* node always selects the local_request message recording the highest quality among the messages that it has received from its G_s-child nodes. Then, the *active* node adds its own quality into the quality recorded in the selected message, and sends a local_request message to its G_s-parent nodes. In this way, an *active* node always selects the composition with the highest quality within its local k-hop area. Hence, when a *root* node gets each type of its G_s-child nodes' *local_request* messages, it obtains the highest-quality *overlay graph* that it can to achieve in global view with the constraint that *active* nodes send messages within local k-hop area. □

Obviously, the larger k is, the higher quality of *overlay graph* can be obtained. When k equals to the maximum hop counts in G_c, the quality of *overlay graph* achieved by LASEC is equivalent to that of *overlay graph* achieved by a centralized algorithm. However, larger k leads to larger $T(k)$, thus resulting in higher running time of the algorithm.

Note that, we define the quality of the *overlay graph* as the sum of all individual quality of *active* nodes therein in this paper. When various weights of the nodes' quality need to be considered in the *overlay graph*'s quality, the proposed LASEC can just combine with the decentralized approach in [29] which focuses on computing the services' weight in the QoS of composition.

5 Performance Evaluation

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 implemented a distributed approach (DEC) [14] as well as a pull-based model of the centralized algorithm for service composition (CEN) [38], and have compared their 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.

DEC. Distributed embedding algorithm. In this algorithm for each service composition coordinator is assigned, which broadcasts queries for each service type requested, available service providers respond, coordinator sends ACK message to them which is followed by confirmation from service providers.

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 uniformly deployed. The total number of nodes in the network is varied to examine the effect of system scale on the performance. Also we studied influence of request complexity, which in this paper is defined as the number of services specified in the request graph. A service type is randomly assigned to each node. In each network we set the total number of service types so that the density of service types (ds=N/ns) is comparable. The simulation parameters are listed in Table 3.

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

TABLE 3 Simulation Parameters

Parameters	**Values**										
Number of nodes, (N)	64	81	100	121	144	169	256	400	441	484	529
Number of service types (n_s)	10	13	17	20	24	28	43	67	74	81	88
Service density	6, 7.5, 9, 11, 12.5, 14										
Request complexity	3, 4, 5, 6, 7, 8										
Territory scale (m^2)	200										
Transmission radio range (m)	40										

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. Each point is obtained by averaging 100 runs.

5.1 Scalability

5.1.1 Scalability with Regards to the Network Size

The number of messages generated by service composition protocol according to the size of the network is presented in Fig. 7. It considers the number of messages necessary to find first composition. Our LASEC service composition mechanism 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. DEC also scales very well, since it does not collect global information and considers only some limited neighborhood. However, for each interaction between service coordinator and prospective service provider it sends more messages (query, reply, ack, confirmation). Similar results can be observed for the time necessary to find first answer, according to the size of the network, which is presented in Fig. 9. As can be seen from the results, our LASEC algorithm performs better than DEC and 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 and DEC scale better than CEN.

5.1.2 Scalability with Regards to the Request Complexity

Fig. 8 gives the number of messages generated by LASEC according to the size of the request. It shows that number of messages increases with the request complexity. LASEC avoids flooding the network with messages due to its localized nature. However, we observe, that along with the growth of request complexity, LASEC and DEC generate more messages, while CEN is more scalable. Since CEN collects global information, it collects the same amount of information for different request complexity, and incurs only a little additional cost for computing the result, while LASEC and DEC cost depends on the result complexity. Fig. 11 gives the time necessary to find first composition, according to request complexity. LASEC performs better than DEC and CEN. 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, DEC, CEN and LASEC perform similarly 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, CEN needs additional time for processing the information.

Figures 9 and 12 show the time and message overhead necessary to compose a service, according to the service density in the environment. We observe that response delay as well as number of messages increase with service density. LASEC has smaller overhead than DEC and CEN for composing the service.

5.2 Composition Locality

We also evaluated our approach in terms of composition locality, which refers to the distance between the services used in composition. In large scale and dynamic pervasive computing environments composition locality should be as small as possible. It is especially important when the services specified in the request need to interact with each other while they are embedded on different nodes in the network. To minimize cost of interaction, as well as composition, all atomic services in the composed service need to be located as close to each other as possible. Fig. 12 shows the result of our simulation. For the request of 6 types of services two representative compositions are shown. Through simulation we have observed that LASEC naturally tends to select compositions with high degree of composition locality. For service composition approaches which collect global information, it is difficult to guarantee composition locality, unless they collect additional information referring to the location of the services and take it into consideration during selecting of services, which may introduce additional overhead.

5.3 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 recompose the service as quickly as possible considering new state of the environment.

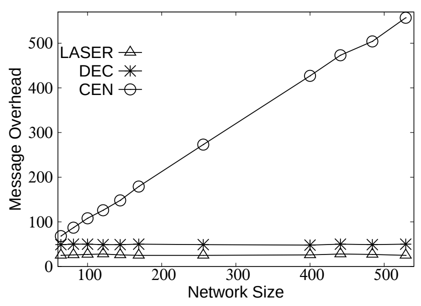


Fig. 7 Impact of network size on message overhead.

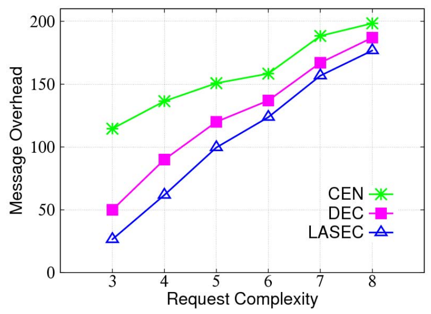


Fig. 8 Impact of request complexity on message overhead.

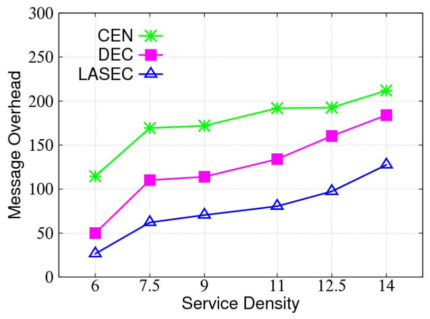


Fig. 9 Impact of service density on message overhead.

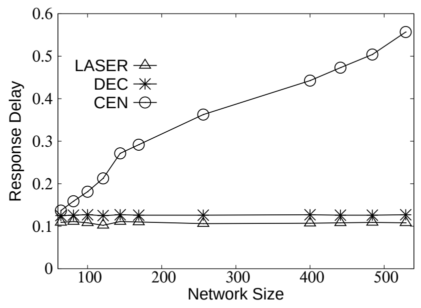


Fig. 10 Impact of network size on response delay.

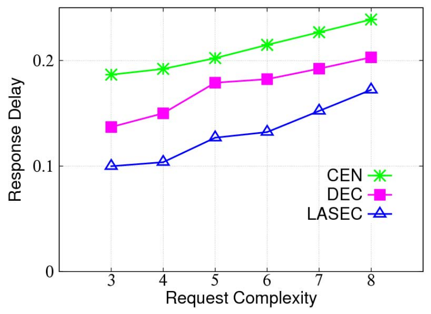


Fig. 11 Impact of request complexity on response delay.

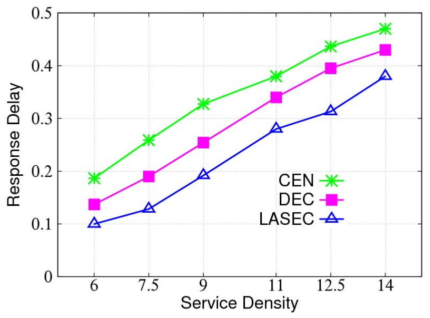


Fig. 12 Impact of service density on response delay.

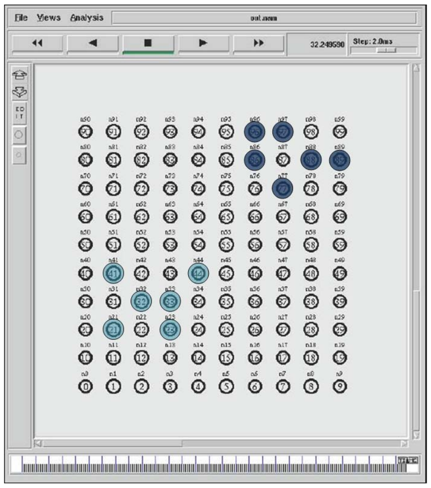


Fig. 12 Composition locality. Atomic services in the composed service should be close to each other.

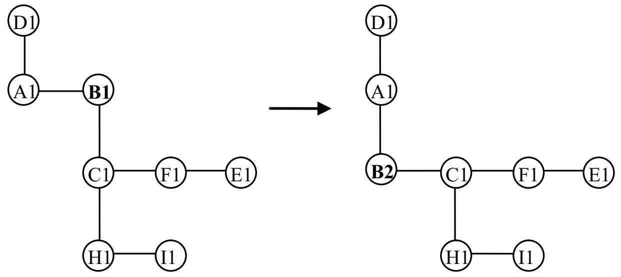
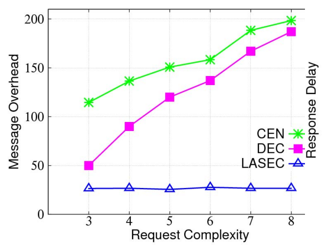
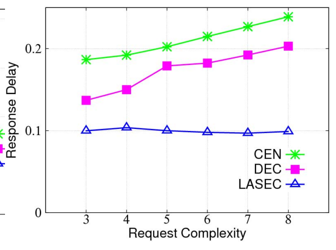


Fig. 13 Dynamicity - one of the services becomes unavailable and it must be replaced.

With our localized approach, it is possible to ensure that the request can be recomposed with minimal overhead. Although in extreme cases of dynamic (mostly theoretical) the composed service may need to be completely recomposed, typically, 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. In our simulations, we considered the scenario shown in Fig. 13. 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. Each node in the request graph corresponds to one service provider in S. If one of the service providers from S becomes unavailable, then corresponding node in the request graph must be recomposed. Based on our localized approach, only immediate neighbors of the unavailable service provider will be responsible for a new composition. That means service providers that corresponds to the nodes in the request graph which share links with the node corresponding to unavailable service provider. Through simulations, pictured in Fig. 14(a) and Fig. 14(b), we have observed, that recomposition cost is stable and similar to that of composing simple requests.



(a) Impact on message overhead.



(b) Impact on response delay.

Fig. 14 Coping with dynamicity.

5.4 Prototype Implementation

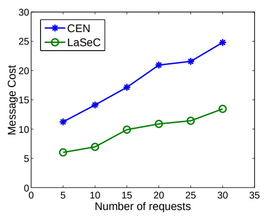
5.4.1 System Architecture

We developed a prototype for our service composition approach to study its performance in the real environments. The system architecture for the prototype is shown in Fig. 15(a). 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. As shown in Fig. 15(b)- 15(c), we used 40 MicaZ nodes with a MIB600 gateway for the hardware and deployed them in an indoor environment.

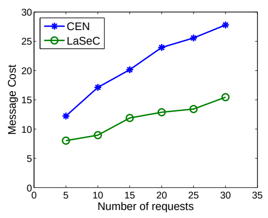
5.4.2 Experiments

We conducted experiments based on our prototype. In order to enlarge the number of services in the prototype, we implemented several services on one node and predefined a number of services for each sensor node in our testbed. Similar to our simulation, we use CEN as a reference for performance comparison. We implemented CEN based on existing multihop routing protocols provided by TinyOS [35] where the nodes will simply send their available services to sink for composition. We use message cost and response delay as metrics for the experiments and study the performance according to the size of the request.

The experimental results for message cost are shown in the Fig. 16(a) and in the Fig. 16(b). Data in the Fig. 16(a) are obtained from experiments on 10 predefined services while the data in the Fig. 16(b) are obtained from experiments on 20 pre-defined services. Compared to CEN, LASEC can reduce the message cost by over 50 percent. This is primarily because of LASECs decentralized service composition. Unlike CEN where all the messages must be transmitted to the sink, a lot of messages in LASEC are processed in-network. In addition, LASEC also scale better when the number

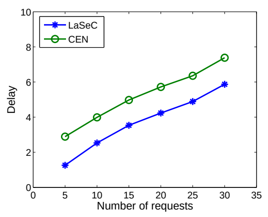


(a) 10 pre-defined services.

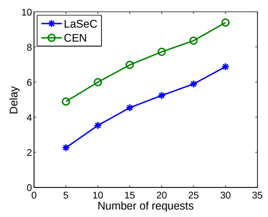


(b) 20 pre-defined services.

Fig. 16 Experimental results for message cost.



(a) 10 pre-defined services.



(b) 20 pre-defined services.

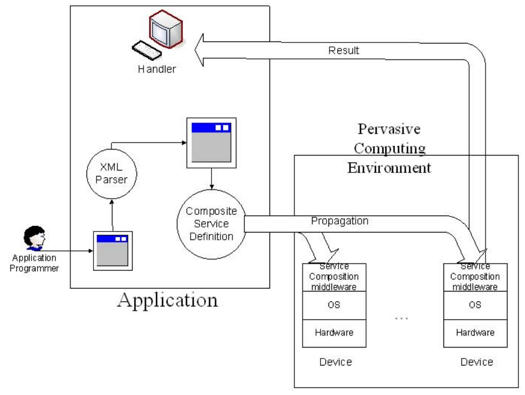
Fig. 17 Experimental results for message delay.

of services increases. This is because when the number of services increases, the number of messages to be sent to the sink in CEN will increase accordingly. This inevitably causes a lot of packet collisions and message re-transmission. We believe such cost saving is of particular significance when considering the fact that many devices in pervasive computing environment are battery-powered.

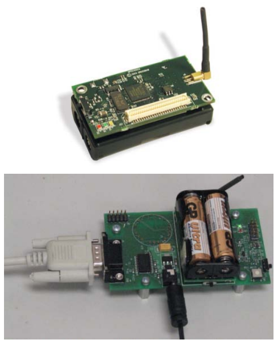
The experimental results for message delay are illustrated in the Fig. 17(a) and in the Fig. 17(b). Similarly, the data in the Fig. 17(a) are obtained from experiments on 10 pre-defined services while the data in the Fig. 17(b) are obtained from experiments on 20 pre-defined services. We can see that on average, LASEC achieves smaller delay, especially when there are more services in the environment. This is because in CEN, the delay is primarily introduced by message collision and packet retransmission when the devices try to deliver messages to the sink. On the other hand, the decentralized service composition in LASEC allows more data to be processed in-network and hence, reduces the chance of packet collision.

6 Conclusion

In this paper, we have proposed a localized algorithm named LASEC for service composition. The LASEC algorithm is suitable for service composition applications deployed in large scale and dynamic environments. In LASEC, no coordinator collects global information, nodes and links between them are mapped in subsequent stages and nodes construct only local part of the solution, while together achieving global goal. Comparing with existing decentralized and pull-based centralized approach, the proposed localized algorithm


Handler
Result
Pervasive Computing Environment
XML Parser
Composite Service Definition
Propagation
Application Programmer
Service Composition middleware
OS
Hardware
Device
Application

(a) System architecture.



(b) Prototype hardware.

(c) Part of the testbed.

Fig. 15 Prototype implementation

is delay-constrained and message-efficient. With an efficient composition process, the proposed algorithm can be easily combined with the existing calculation methods on composition QoS to achieve high quality of composed service.

References

[1] D. Sakamoto, K. Honda, M. Inami, T. Igarashi, *Sketch and Run: A Stroke-based Interface for Home Robots*, In: 27th In-ternational Conference on Human Factors in Computing Sys-tems, pp. 197C200 (2009)

[2] S. Zhao, K. Nakamura, K. Ishii, and T. Igarashi, *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] M. Chu, B. Dalal, A. Walendowski, and B. Begole, *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 Fac-tors 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] G. Ngai, S. C.F. Chan, W. W.Y. Lau and J. 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. 17C29.

[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 Net-works,"* 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 Intl Conf. System Sciences, p. 298, Jan. 2002.

[28] M. Wang, J. Cao, J. Siebert, V. Raychoudhury, J. Li, *Ubiquitous Intelligent Object: Modeling and Applications,* The 3rd International Conference on Semantics, Knowledge and Grid (SKG2007)

[29] X. Gu and K. Nahrstedt, *Distributed Multimedia Service Composition with Statistical QoS Assurances,* IEEE Trans. Multimedia, 2005.

[30] X. Gu, K. Nahrstedt, and B. Yu, *SpiderNet: An Integrated Peer-to-Peer Service Composition Framework*, Proc. IEEE Intl Symp. High-Performance Distributed Computing (HPDC-13), June 2004.

[31] X. Gu and K. Nahrstedt, *"On Composing Stream Applications in Peer-to-Peer Environments"*, IEEE Transactions on Parallel and Distributed Systems, , vol.17, no.8, pp.824-837, Aug. 2006

[32] X. Gu, K. Nahrstedt, R.N. Chang, and C. Ward, *QoS-Assured Service Composition in Managed Service Overlay Networks*, Proc. IEEE 23rd Intl Conf. Distributed Computing Systems (ICDCS 03), 2003.

[33] X. Gu, P.S. Yu, and K. Nahrstedt, Optimal Component Composition for Scalable Stream Processing, Proc. IEEE 23rd Intl Conf. Distributed Computing Systems (ICDCS 05), 2005.

[34] S. Herborn, Y. Lopez, and A. Seneviratne, "A distributed scheme for autonomous service composition", In Proceedings of the first ACM international workshop on Multimedia service composition (MSC '05), 2005.

[35] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks", In Proceedings of the 1st International conference on Embedded networked sensor systems (SenSys '03). 2003.

[36] X. Su and J. Rao. A Survey of Automated Web Service Composition Methods. In Proceedings of First International Workshop on Semantic Web Services and Web Process Composition, July 2004.

[37] A. Klein, I. Fuyuki, S. Honiden, "SanGA: A Self-Adaptive Network-Aware Approach to Service Composition," IEEE Transactions on Services Computing, vol. 99, no. PrePrints, p. 1, , 2013

[38] I. Paik, W. Chen, M.N. Huhns, "A Scalable Architecture for Automatic Service Composition," IEEE Transactions on Services Computing, vol. 99, no. PrePrints, p. 1, , 2012

[39] A. Passarella, M. Kumar, M. Conti, and E. Borgia, "Minimum-delay service provisioning in opportunistic networks," IEEE Transactions on Parallel and Distributed Systems, Vol. 22, no. 8, pp. 1267-1275, August 2011

[40] S.A. Tamhane, M. Kumar, A. Passarella, andM. Conti, "Service Composition in Opportunistic Networks," Green Computing and Communications (GreenCom), 2012 IEEE International Conference on , vol., no., pp.285,292, 20-23 Nov. 2012