Decentralized Service Composition in Pervasive Computing Environments

Joanna Izabela Siebert† Jiannong Cao† Long Cheng†‡ Edwin Wei† Canfeng Chen§ Jian Ma§
†Department of Computing, Hong Kong Polytechnic University, Hong Kong
‡State Key Lab of Networking & Switching Tech., Beijing Univ. of Posts and Telecomm., China
§Nokia Research Center, Beijing, China
csjcao, csjsiebert, fcslcheng @comp.polyu.edu.hk, fcanfeng-david.chen, jian.j.mag@nokia.com

ABSTRACT
In a pervasive computing environment, the devices are embedded in the physical world, providing services and interconnected by a communication network. Composition of these services is important issue of pervasive applications which integrate the physical and cyber worlds. Most existing research on service composition in pervasive computing relies on the existence of one or more entities that maintain the global service information. However, such an approach is not always practical due to dynamicity of the environment. In this paper, we propose a fully decentralized approach to service composition. We first model the service composition problem as finding an overlay of the communication network that matches the composition graph. The problem is proved to be NP-complete. We propose an algorithm for the devices to cooperatively construct the requested services through localized interactions. For the purpose of reducing redundant broadcast we propose the service composition backbone built in a fully localized way. We have carried out extensive simulations. to evaluate the performance of our algorithm. Compared with existing pull-based centralized techniques our decentralized service composition algorithm on the service composition backbone is more efficient in terms of response delay and message overhead, while achieving similar quality of composed service.

Categories and Subject Descriptors
C.2.1 *[**Computer-Communication Network**]:* Network Architecture and Design—*Distributed networks*

General Terms
Algorithms, Design

Keywords
Pervasive computing environment, service composition, decentralized control, localized interactions

1. INTRODUCTION
The 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. 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 multiple instances of requested functionalities can be found and services exist in the environment as well. According to requirements we need to select one instance for each service type.

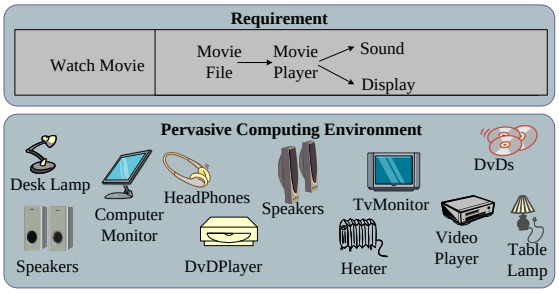

Figure 1. Service composition application.

To the best of our knowledge, previous research on service composition in pervasive computing assume collecting global information, which is impractical in large, dynamic environments. To address these disadvantages, we present in this paper a decentralized approach applied on the service composition backbone. Unlike previous algorithms that rely on global knowledge, in our algorithm each device only maintains local state information about its physical neighbors. As a result, service providers will build an overlay network graph which satisfies the requirement with the quality of service comparable to centralized approach.

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 distributed manner, through localized interactions between service providers. We transform this problem

to subgraph isomorphism problem. It is proven that this problem is NP-complete [10].

The rest of the paper is organized as follows. In Section 2, we present the related work. In Section 3, we describe our system model together with the problem formulation, and in Section 4 and 5 we describe the design of the proposed algorithms. Finally, the proposed approach is evaluated in Section 6, and concluded in Section 7

2. RELATED WORK

To the best of our knowledge, thus far there is no research work that achieves the same objectives as ours.

Many works [1-6] for PvCE propose centralized solutions to service composition. Service requestors submit requests to the centralized directory and the directory makes a decision on the list of services that should be returned to the requestors. 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.

Some works exploiting distributed approach have already been proposed. As the first step towards decentralization of service composition they introduce distributed directory of services. For example, in [9] 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 [7] 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 [8]. It is different from [7] 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. 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.

In our work reported in this paper there is no special entity to manage service composition process. Service providers communicate only with their local neighbors. No service provider knows the full global information or gathers it.

3. SYSTEM MODEL

3.1 Layered System Architecture

In this section, we describe the system model of the proposed approach, as 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. SCB layer describes an overlay communication infrastructure. Each node is in the SCB or 1-hop away from the SCB. Request layer shows the user-specified service description, which can be accessed by the service providers in the communication layer.

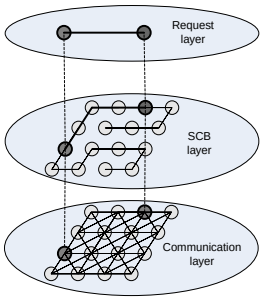

Figure 2. Decentralized service composition system.

3.2 Problem Formulation

Definition 1 [Service Provider]: A service provider is described as a directed attributed graph $\vec{G}_S = \{V_S, \vec{E}_S\}$, where V_S is a single element set that represents the service and $\vec{E}_S$ is a set of directed edges that represent the inputs and outputs of the service [9]. The vertex attributes include description of the provided service type.

Definition 2 [Communication Network]: The underlying communication network is described as a graph $G_E = (V_E, E_E)$, where the vertices correspond to service providers, and $(x, y) \in E_E$ if and only if y can transmit and receive message directly from x.

Definition 3 [Composite Service]: A composite service is modeled as a graph $\vec{G}_R = (V_R, \vec{E}_R)$ where V_R corresponds to required services and the set of edges $E_R \subseteq V_R \times V_R$ represents required composition relationships, such as output of one service is accepted by output of another service.

Definition 4: Service Composition Problem (SCP): Given a finite collection of service providers and a composite service request $\vec{G}_R$; construct in a decentralized manner an overlay graph $\vec{G}_A = (V_A, \vec{E}_A)$, such that $\vec{G}_A$ is the subgraph of $\vec{G}_E$ and $\vec{G}_A$ is isomorphic to $\vec{G}_R$.

4. SERVICE COMPOSITION ALGOTIHM

In this section, we describe our algorithm that enables the devices to cooperatively construct the requested service. Device candidates decide with whom to cooperate only based on the information of the devices within its physical neighborhood. Our idea is to grow sections of composed service from available services. First, service provider identifies what service type it needs to interlock with through his output and then searches for appropriate service provider with matching input type. It is possible that service provider forms a section with another section. By merging pieces with each other, eventually global solution emerges.

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

Table 1. Data structures in 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 2. Message types in service composition

Message	Purpose
invitation_card	Candidate node i announces invitation to nodes which service type succeeds the type of node i in the requested service.
response_card	Response to the invitation.

Decentralized service composition algorithm is summarized in Algorithm 1.

Algorithm 1 Decentralized Service Composition

INPUT: $G_R = (V_R, E_R)$, $N(x) \subseteq \mathrm{P}$, $E_E'(x)$
OUTPUT: $\vec{G}_A = (V_A, \vec{E}_A)$
Find Candidate Solutions() {
// The code executed by each node i
(0) *status* ←*WAITING*;
(1) Wait until received G_R,
(2) if (*in_edges* of *i* in G_R is empty) *status* ←*INITATOR*;
(3) if (*in_edgesr* of *i* in G_R is not empty) *status* ←*CANDIDATE*;
(4) if (*i* is not in G_R) *status* ←*ROUTING*;
// The code executed by each *INITIATOR* node
(5) Send (*invitation_card*) to *out_edges*
// The code executed by each *CANDIDATE* node
(6) Wait until received *(invitation_card*);
(7) *in_edges*= *in_edges* –(*sender)*;
(8) if (*in_edges* = 0) {Generate (*invitation_card*); send (*invitation_card*) to *out_edges*; };
(9) else store (*invitation_card*);
(10) Wait until received *(response_card*);
(11) if (merged *response cards* = G_R) become *DONE;*
(12) else send (*response_card*) to *in_edges;*
// The code executed by each *ROUTING* node
(13) Wait until received *(invitation_card*);
(14) Broadcast *(invitation_card*);
(15) Wait until received *(response_card*);
(16) Forward *(response_card*); }

At the beginning of the algorithm, the requested composite service $\vec{G}_R$ is submitted to each device in G_E. 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 invitation_card that is set with its designated service type according to the node that succeeds the type of node i in the requested service. When an invitation_card arrives at a device that is a candidate for the designated service type of that invitation_card, the invitation_card is stopped and merged with invitation_cards from other incoming edges. When there was at least one invitation_card coming from all incoming_edges, the device generates new invitation_card, with new designated service. When invitation_card arrives at the service type that has no outgoing edge in the requested graph, the device without outgoing edges returns response_card to the received invitation_card to its initiators. When any device receives the response_card from different devices, it merges them.

5. SCB CONSTRUCTION

In this section, we describe the algorithm that enables the devices to construct the service composition backbone. It is a fully localized algorithm, based on the approach proposed in [11].

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

Table 3. Data structures in SCB construction

Variable	Meaning
status	Status of the node i = {IN_SCB, NOT_IN_SCB }
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.
2-hop_n_Unconnected	Set of 2-hop neighbors that are not connected with node i through node from Connector_set
Connector_ set	Set of 1-hop neighbors that connect node i with its all 2-hop neighbors

Table 4. Message types in SCB construction

Message	Purpose
Req(id)	Node i requests from 1-hop neighbors their id
Req(1-hop_n)	Node i requests from 1-hop neighbors their set of all 1-hop neighbors
In_Connector _set_i	node i informs nodes in its Connector_set about being elected as a connector.

Localized algorithm for SCB construction is summarized in Algorithm 2.

Determine connector set. At the beginning of this phase, node i collects neighborhood information from its 1-hop neighbors, based on which it builds information about its 2-hop neighborhood. Node i selects a set N of 2-hop neighbors that are connected with node i only through one 1-hop neighbor. 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 Connector_set. 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. At the end of this phase, node i informs 1-hop neighbors that are in its candidate set.

Determine SCB nodes. The purpose of this round is to determine if node i belongs to the SBC. In this phase two simple rules are employed. If node i has smallest id among his 1-hop neighbors it belongs to SCB. If node i is in the candidate set of its smallest neighbor it belongs to SCB.

Algorithm 2 SCB Construction

INPUT: *1-hop_n, id_i ,*

OUTPUT: *SCB*

Determine *Connector_set* () {

(1) Send *Req(1-hop_n)* to *1-hop_n;*

(2) *N* ←2-hop neighbors that are connected with node *i* only through one 1-hop neighbor;

(3) if (node *k* is the only 1-hop neighbor that connects node *i* with the 2-hop neighbor from set *N*) *Connector set*←node *k*;

(4) if (*2-hop_n_Unconnected* = 0) { Send (*In_Connector _set_i*) to *Connector set*; }

(5) else if (node *k* connects largest number of nodes from *2-hop_n_Unconnected*) *Connector set*←node *k*; }

Determine SCB nodes () {

(6) *status* ←*NOT_IN_SCB*;

(7) if (min (*id_i(1-hop_n))=id_i) status* ← *IN_SCB*

(8) if (received *In_Connecor_set_i* from min (*id_i(1-hop_n))) status* ← *IN_SCB* }

6. PERFORMANCE EVALUATION

We have carried out extensive simulations to evaluate the performance of our algorithm that has been applied to the network with the SCB formed according to the proposed localized algorithm (B-LOC). Moreover, to show the advantage of our decentralized service composition algorithm we have also implement a pull-based model of the centralized algorithm for service composition (CEN) [5], and have compared its results with ours. CEN is a centralized service composition algorithm, in which there is a dedicated service director, initially empty. When a user specifies a composite service request, the directory pulls from the environment the information related to service types.

We evaluated the performance of the algorithms in terms of message overhead, response delay and composed QoS. Message overhead is the number of messages generated by the composition process, without injecting the request. Response delay is the interval between the time a request is received by the system and the time corresponding reply is returned by the system. Composed QoS is a sum of QoS of atomic services that take part in the composition.

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. Each point is obtained by averaging 100 runs

Table5. Simulation parameters

Parameters	Values		
Number of nodes, (N)	64	81	100
Number of service types (n_s)	10	13	17
Service density	6, 7.5, 9, 12.5		
Request complexity	3, 4, 5, 6		
Territory scale (m^2)	200		
Transmission radio range (m)	40		

Effect of network size on message overhead. Our B-LOC service composition mechanisms require less messages than CEN for finding first result, which can be seen in Figure 3. 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 B-LOC takes advantage of localized interactions.

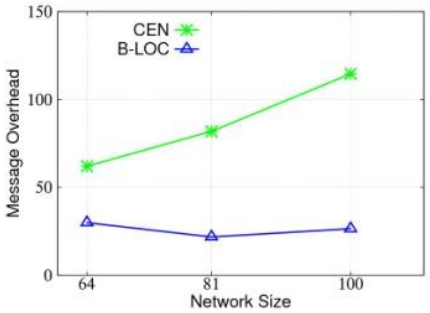

Figure 3. Impact of network size on message overhead.

Effect of request complexity on message overhead. Figure 4 shows that number of messages increases with the complexity of the request. B-LOC algorithm takes advantage of the localized approach and due to forwarding the messages only on the SCB. B-LOC avoids flooding the network with messages.

Effect of network size on response delay. As can be seen from the results in Figure 5, our B-LOC algorithm performs better than CEN. Since in CEN messages are flooded in the network, the collision effect causes increase in delay. B-LOC experiences lower rate of collisions, due to forwarding messages on the SCB.

Effect of request complexity on response delay. Figure 6 shows that our B-LOC 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.

Effect of service density on message overhead, response delay and composed QoS. We observe in Figure 7 and 8 that response delay as well as number of messages increase with service density. Our B-LOC service composition mechanism has smaller overhead than CEN for finding optimum result. However CEN is able to achieve slightly higher quality of composed service. This can be observed in Figure 9. This is possible, since in CEN, central coordinator has global knowledge about all service providers in the environment, while B-LOC searches for service providers only in local environment.

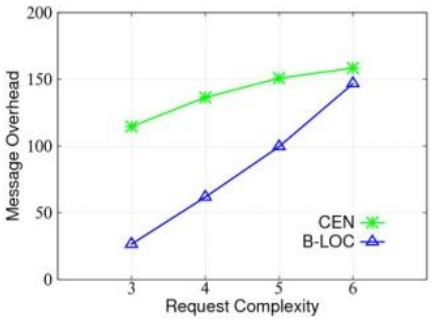

Figure 4. Impact of request size on message overhead.

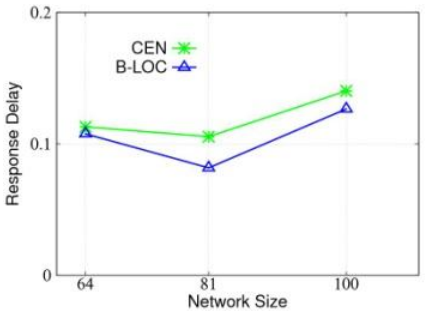

Figure 5. Impact of network size on response delay.

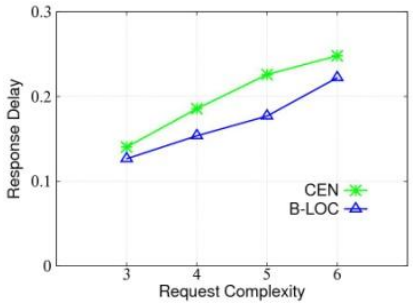

Figure 6. Impact of request complexity on response delay.

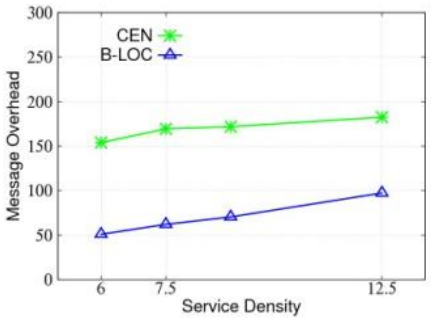

Figure 7. Impact of service density on message overhead.

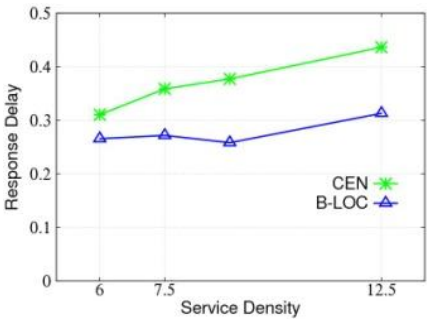

Figure 8. Impact of service density on response delay.

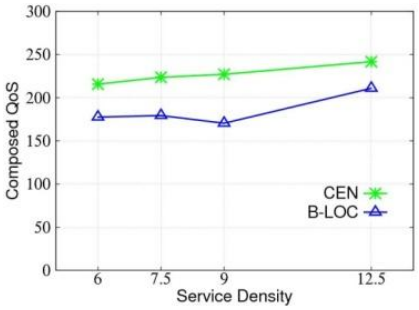

Figure 9. Impact of service density on composed QoS.

7. CONCLUSION

This paper presents a decentralized approach to service composition using localized interactions. We have implemented our decentralized service composition algorithm. We have applied our algorithm utilizing localized interactions to the proposed SCB (B-LOC). Our evaluation has shown that compared with existing pull-based centralized techniques our algorithm is more efficient in terms of message cost and response time. However there is a trade-off between overhead and QoS of composed service. While our B-LOC mechanism has smaller overhead, CEN is able to achieve slightly higher composed QoS.

8. ACKNOWLEDGMENTS

This work is partially supported by Nokia's Collaboration Grant under grant No.H-ZG19

9. REFERENCES

[1] 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.

[2] 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.

[3] Z. Song, Y. Labrou, and R. Masuoka. Dynamic service discovery and management in task computing. Mobiquitous, 00:310-318, 2004.

[4] 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.

[5] 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.

[6] A. Bottaro, J. Bourcier, C. Escofier, and P. Lalanda. Autonomic context-aware service composition. 2nd IEEE International Conference on Pervasive Services, 2007.

[7] 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.

[8] 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.

[9] S. Kalasapur, M. Kumar, B. Z. Shirazi. Dynamic Service Composition in Pervasive Computing. IEEE Transaction on Parallel and Distributed Systems, 2007.

[10] Z. Ling. An Effective Approach for Solving Subgraph Isomorphism Problem. In Proceedings of the IASTED International Conference, 1996.

[11] C. Adjih, P. Jacquet, and L. Viennot, "Computing Connected Dominated Sets with Multipoint Relays," Technical Report 4597, INRIA-Rapport de recherche, Oct. 2002.