A Low-Latency Service Composition Approach in Mobile Ad Hoc Networks

Chenyang Liu, Jian Cao, Frédéric Le Mouël

A Low-Latency Service Composition Approach in Mobile Ad Hoc Networks

Chenyang Liu
Shanghai JiaoTong University
No.800 DongChuan Rd.,
Shanghai, China
Schumeichel_2003@163.com

Jian Cao
Shanghai JiaoTong University
No.800 DongChuan Rd.,
Shanghai, China
cao-jian@cs.sjtu.edu.cn

Frédéric Le Mouël
University of Lyon
INSA-Lyon, CITI-INRIA F-69621,
Villeurbanne, France
frederic.le-mouel@insa-lyon.fr

ABSTRACT
In order to offer complex services to the users, separate services located at different devices in MANET should be composed in a mobile ad hoc network. A distributed approach to search for the *Service Composition Path (SCP)* with a low latency is proposed, which is based on two methods, *Path Filtering* and *Path Combination*. These two methods avoid unnecessary message transmissions, and greatly improve the searching efficiency. The experiment results show the superiorities of the approach to its counterpart.

Categories and Subject Descriptors
C.2.2 [**Computer-Communication Networks**]: Network Protocol

General Terms
Algorithms, Performance, Design, Experimentation.

Keywords
MANET Service; Service Composition Path; Path Filtering; Path Combination.

1. INTRODUCTION
In recent years, service composition in pervasive environments [1-6], especially in MANETs, has attracted more and more research interests. A MANET consists of multiple stand-alone and self-configurable computing nodes, which use a multi-hop routing mechanism, such as DSR and AODV, to communicate with each other. With the emergence of various intelligent devices, the nodes in a MANET will be equipped with richer hardware and software resources than before. These resources pave the way for constructing micro-service frameworks on MANET nodes to facilitate cross-device service discovery and invocation, which lays the foundation for service composition in the MANET.

Many researchers proposed their frameworks for service composition in the MANET [7-10]. Their approaches can be roughly divided into two categories. In the first category, for each function, a path from the composite service-requesting node to the service-providing node corresponding to this function should be found. Obviously, this approach suffers from low efficiency for service execution, because of the messages transmitted back and forth on the path between the composite service-requesting node and the service-providing node. In another category a composite service is represented as an executable path, in which previous nodes provide services and intermediate results will be transmitted to successive ones through multi-hop transmission. Our approach belongs to this category.

Since there often exist more than one composition solution capable of fulfilling a request, searching for the optimal one in a MANET will lead to packet flooding problem. In order to obtain the optimal solution more efficiently, an approach is proposed in this paper and it has following features:

1. Making use of *Path Filtering* to discard non-optimal composition solutions to avoid unnecessary message transmissions;
2. Searching for the Forward and Backward partial composition paths at the same time, and then applying *Path Combination* to obtain the complete solution more efficiently.

The contribution of our approach is improving the quality of the composition solution evidently, and reducing the time and communication cost greatly.

2. RELATED WORK
In recent years, many researchers proposed their own architectures for service composition in pervasive environment. The approaches proposed in [1-2] are based on service model and user context information, while [3-4] dwell on reliability and availability modeling of service composition through probabilistic analysis on a MANET nodes' history data. In [5-6], parameter-based service matching approaches are proposed in the pervasive service environment, which employs *Semantics Networks* and *Ontology Learning* to match two services.

Confronted constantly changing network topology and resource limitations in the MANET environment, some researchers have proposed a mechanism, called *Broker Arbitration*, which selects the best broker to be responsible for service composition in the MANET [7-8]. However, *Broker-based* service composition will lighten heavy burdens on some nodes with poor resources, and may incur message flooding for updating the cached service information. This shortcoming is overcome by the mechanisms suggested in [9], which apply *Group-based Service Discovery (GSD)* [10], to search for candidate atomic services.

In fact, a service composition solution can also be described as a sequential workflow. Based on this idea, [11-13] present algorithms for obtaining the optimal composition workflow with minimum execution time. These algorithms forward the composition request among the nodes, and regard the first complete workflow as the optimal solution. However, due to the lack of limitation on message forwarding, these algorithms will suffer from packet flooding in MANET. Contrast to these methods, our approach introduces *Path Filtering*, which limits the message forwarding effectively, and *Path Combination*, which saves an amount of time while obtaining the optimal solution.

3. PROBLEM FORMATION OF SERVICE COMPOSITION IN THE MANET

In our assumption, every function in the MANET is modeled as a 4-element tuple, $<N_i, S_j, F_k, D_j>$, which represents that service component S_j is deployed on MANET node N_i, and implements the function F_k. D_j is the service execution time. While we employ a 3-element tuple, $<N_i, N_j, D_{ij}>$ represents a bidirectional link between node N_i and node N_j, whose network communication delay is D_{ij}. Based on these definitions, *Service Composition Path (SCP)* is described as a sequential loop path, starting and ending at the requesting node, and consisting of all required *service nodes* and *relay nodes* and their corresponding communication links. Hence, the total path delay can be computed according to *Formula* (1):

$$T_{path} = \sum D_{si} + \sum D_{NiNj} \qquad (1)$$

where D_{si} is the D_i in $<S_i, N_j, F_k, D_i>$, and D_{NiNj} is the D_{ij} in $<N_i, N_j, D_{ij}>$.

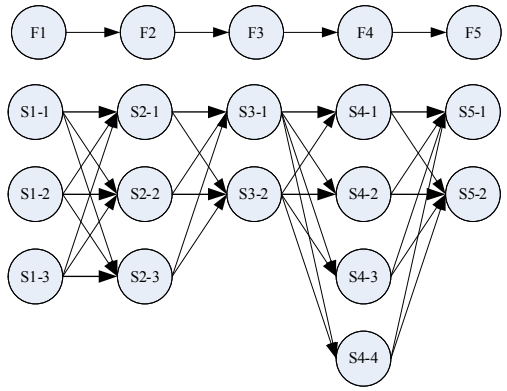

Fig. 1. An example of all SCPs for a 5-function-composition

For every sub-function in a service composition request, we can find several corresponding service components in the MANET so that there exist multiple SCPs satisfying the same composition request. For example in Fig.1, there is a functional graph that consists of 5 function nodes, and each function can be fulfilled by several services interchangeably. Our problem is to find the latency-optimal SCP in the MANET with few time and network messages as possible. This problem can be abstracted as a *Distributed Multi-stage Decision Problem*, which means the resolution is based on the collaboration among the MANET nodes.

4. FORWARD SEARCH AND PATH FILTERING

ForeNoRepSearch algorithm continues forwarding and updating the partial SCP unless it arrives at the requesting node or is discarded. In the searching process, this algorithm searches for a service for each sub-function in sequence. Any node receiving a request message will query the local system for current required service, update the partial SCP, and apply *Path Filtering* to decide whether to continue forwarding this path or not. When the requesting node receives a SCP who implements all the required sub-functions, it will compare the SCP with current optimal one to obtain a better solution. The algorithm will continue running until there is no request message on any node, which means the search process has converged to an optimal solution. The algorithm of *ForeNoRepSearch* on one node is shown below.

Algorithm 1 ***ForeNoRepSearch***

```
var
  reqMsg: current received SERV_COMP_REQ;
begin
  if all the services for the required sub-functions have been
  discovered && current node is the requesting node
    then replace the optimal SCP with reqMsg.scForePath;
  else
    Search the local service list for the next required service;
    Add a service or relay node to reqMsg.scForePath;
    if PATH-FILTER (reqMsg.scForePath) succeed
      then forward reqMsg to all neighbors;
end.
```

Path Filtering is proposed to judge whether current partial SCP is worth forwarding or not, whose principle is based on the concept of *Optimal Substructure* in *Dynamic Programming*. If current partial SCP is included in the optimal SCP, there is no other partial SCP including current node with the same number of services and less path delay. To implement *Path Filtering*, every node will establish a SCP list to store local delay-optimal partial SCPs. When a node receives a partial SCP, it will decide whether to discard, or forward the partial SCP and delete the worse ones in the list. The pseudo-code of *Path Filtering* is listed as following:

```
PATH-FILTER (scCurrPath: SCP)
{scCurrPath is the current partial SCP waiting for filtering}
begin
  For each scListPath in SCP list
    if scListPath has more required services than scCurrPath
    && its path delay is no more than scCurrPath
      then discard scCurrPath and return;
  For each scListPath in SCP list
    if scListPath has no more required services than scCurrPath
    && its path delay is more than scCurrPath
      then delete scListPath from SCP list;
  Insert scCurrPath into SCP list;
end.
```

5. FORWARD-BACKWARD SEARCH AND PATH COMBINATION

ForeBackRepSearch Algorithm queries for two required services to update the forward and backward partial SCPs respectively in the request message, and also apply *Path Filtering* to reduce network messages. At the same time, the current node tries to get a new complete SCP through *Path Combination*. If the new formed SCP has lower path delay than the current optimal one, it will be sent back to the requesting node. The following codes give out the logic of *ForeBackRepSearch* algorithm:

Algorithm 2 ***ForeBackRepSearch***

```
HANDLE-SERV-COMP-REQ (reqMsg: SERV_COMP_REQ)
var
  currOptSCP: optimal SCP stored on current node;
  combSCP: current SCP through Path Combination;
begin
```

Compare *reqMsg.scResultPath and currOptSCP;*
Search the local service list for the next required service;
Add a *service* or *relay* node to *reqMsg.scForePath*;
if *PATH-FILTER* (*reqMsg.scForePath*) succeed
then *combSCP* := *FORE-COMBINE-BACK* (*reqMsg.scForePath*);
if *combSCP* has smaller path delay than *currOptSCP*
then replace *currOptSCP* with *combSCP*, send a *SERV_COMP_REP* with *combSCP* to the requesting node;
/*similar operations for backward partial SCP*/
Selectively forward *reqMsg.scBackPath* and *reqMsg.scBackPath* to partial neighbors of current node;
end.

When a node gets a new forward partial SCP after *Path Filtering*, and there exists a backward partial one in the SCP list on current node, which has the rest required services that the forward one has not discovered, we can combine the two partial SCPs to constitute a complete one. Similarly we can apply these operations for a backward one. The following pseudo-code only describes the method for a forward partial SCP to combine the backward ones in the SCP list:

FORE-COMBINE-BACK (*scForePath*:SCP)
var
scResultPath: the optimal SCP returned by this combination;
scCompletePath: current complete SCP through Path Combination;
reqFlow: the workflow in service composition request;
begin
For each *scBackPath* **in** SCP list
if the sum of the number of services in *scForePath* and *scBackPath* is greater than that in *reqFlow*
then combine the *scBackPath* after the *scForePath*, and generate the delay-optimal SCP as *scCompletePath*;
if *scCompletePath* has smaller path delay than *scResultPath*
then replace *scResultPath* with *scCompletePath*;
return *scResultPath*;
end.

6. EXPERIMENT EVALUATION

We wrote a Java-based platform to simulate MANET service environment. Given the number of nodes, services, and functions, this platform can generate a virtual MANET, simulate all the services that implements different functions, and deploy these services uniformly and randomly on different nodes. As a test-bed, each node was modeled as a *Java Thread*, and we applied *Java Message Queue* mechanism to simulate message transmissions among the nodes. In this simulation environment, we implemented the above three service composition algorithms.

We conducted three groups of experiments. In each group, we ran the three algorithms in different environment generated by changing one of the following three configuration parameters: *i)Number of required services*; *ii)Number of MANET nodes; iii) Number of MANET services*. For each configuration, we generate 10 different requests randomly and evaluate the three algorithms in terms of the following metrics: *i)Total time needed for obtaining the solution*; *ii)Total number of network messages*; *iii)Number of nodes in the result SCP*; *iv)Path delay of the result SCP*.

In the first group of experiment, we randomly generated 40 MANET nodes, created 90 services which implemented 20 different functions, and deployed these services uniformly onto the nodes. We generated 10 service composition requests for each number of required services, from 3 to 9. For each request, we executed the above three algorithms, and utilized the mean value to measure the performance of these algorithms, which is shown in Fig. 3(a)~(d).

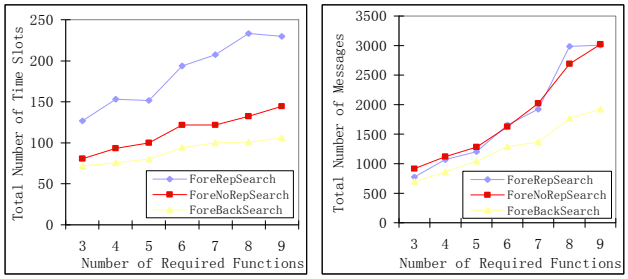

Fig. 1(a~b). Time, Messages vs. Required Services

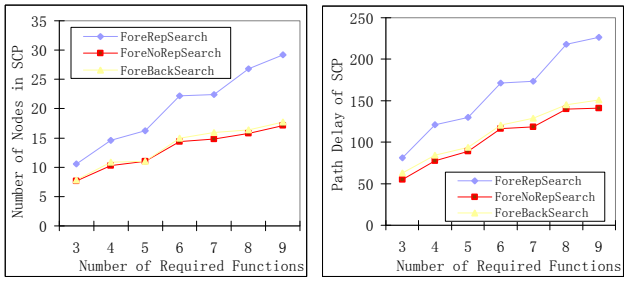

Fig. 3(c~d)Number of Nodes, Path Delay vs. Required Services

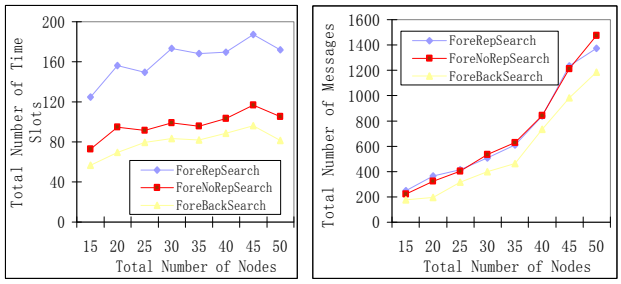

Fig. 4(a~b) Time, Messages vs. MANET Nodes

In the second group of experiment, we generated 90 services which implemented 20 functions, and randomly deployed these services on different number of nodes ranging from 15 to 50. For each environment, we generate 10 different 5-function-compostion requests. Fig. 4(a)~(d) show the statistical result of the experiment.

Similarly, in the third group of experiment, we fixed the number of MANET nodes and functions respectively as 30 and 20, made

the number of MANET services increase from 55 to 90, and ran three algorithms for different randomly generated 5-function-composition requests. The curve graphs of the results are omitted here due to the length limitation of the paper.

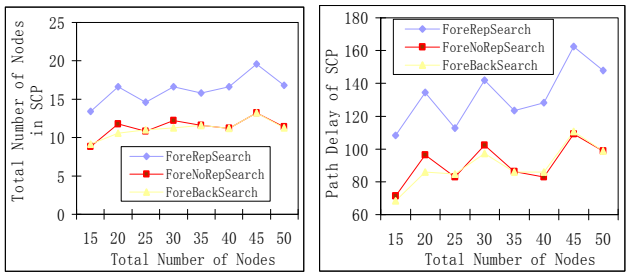

Fig. 4(c~d)Number of Nodes, Path Delay vs. MANET Nodes

7. CONCLUSIONS

In this paper, we define the MANET service composition as a problem of searching for the optimal *Service Composition Path (SCP)* in the network, and propose a distributed approach based on two algorithms, *ForeNoRepSearch* and *ForeBackRepSearch*. *ForeNoRepSearch* introduces *Path Filtering* to impose a constraint on message forwarding, and continue forwarding partial SCPs after finding all the required services, which improves the quality of the result SCP remarkably comparing with *ForeRepSearch*. On the basis of *Path Filtering*, *ForeBackRepSearch* applies *Path Combination* to obtain complete SCPs from combining forward and backward partial SCPs, which enhances the efficiency of searching process significantly. In our experiments, we made comparative assessment on our two algorithms and *ForeRepSearch*, and the results prove that our algorithms can obtain better SCPs with evidently fewer time and network messages than *ForeRepSearch* under different environment configuration.

8. ACKNOWLEDGMENTS

My Thanks to my Ph. D Supervisor for his great support for my research. Funding for this project was provided by a grant from the Rhône-Alpes Region, France.

9. REFERENCES

[1] Jiehan Zhou, Jukka Riekki. Context-Aware Pervasive Service Composition. Proceedings of 2010 International Conference on Intelligent Systems, Modeling and Simulation, Pages 437 - 442, Liverpool, UK, Jan. 27-29, 2010.

[2] Noha Ibrahim, Frédéric Le Mouël. A Survey on Service Composition Middleware in Pervasive Environments. International Journal of Computer Science Issues, Volume 1, Pages 1 - 12, Aug. 2009.

[3] Brent Lagesse, Mohan Kumar, Matthew Wright. ReSCo: A Middleware Component for Reliable-Service Composition in Pervasive Systems. Proceedings of 8th IEEE International Conference on Pervasive Computing and Communications Workshops, Pages 486 - 491, Mannheim, DE, Mar. 29 - Apr. 2, 2010.

[4] Jianping Wang. Exploiting Mobility Prediction for Dependable Service Composition in Wireless Mobile Ad Hoc Networks. IEEE Transactions for Services Computing, Volume 4, Issue 1, Pages 44-55, January-March 2011.

[5] Noha Ibrahim, Stéfane Frénot, Frédéric Le Mouël. User-Excentric Service Composition in Pervasive Environment. Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications. Pages 682 - 689, Perth, Australia, Apr. 20-23, 2010.

[6] Swaroop Kalasapur, Mohan Kumar, Behrooz A. Shirazi. Dynamic Service Composition in Pervasive Computing. IEEE Transactions on Parallel and Distributed Systems, Volume 18, Issue 7, Pages 907-918, July 2007.

[7] Dipanjan Chakraborty, Anupam Joshi, Tim Finin, Yelena Yesha. Service Composition for Mobile Environments. Journal of Mobile Networks and Applications, Volume 10, Issue 4, Pages 435-451, August 2005.

[8] Prasenjit Choudhury, Prasenjit Dutta, Subrata Nandi, Narayan C. Debnath. Mobility Aware Distributed Service Composition Framework in SOA based MANET Application. Proceedings of the 10th IEEE International Conference on Industrial Informatics, Pages 1016 - 1021, Beijing, CHN, Jul. 25-27, 2012.

[9] Jianping Wang. Exploiting Mobility Prediction for Dependable Service Composition in Wireless Mobile Ad Hoc Networks. IEEE Transactions on Service Computing, Volume 4, Issue 1, Pages 44-55, January-March 2011.

[10] Eric Karmouch, Amiya Nayak. Capability Reconciliation for a CSP Approach to Virtual Device Composition. IEEE/ACM Transactions on Networking, Volume PP, Issue 99, Pages 1-13, Apr. 3, 2012..

[11] Weiwei Sun, Zhuoyao Zhang, Weiyu Chen, Bo Peng, Yingxiao Xu. Decentralized Execution of Composite Service in MANET. Proceedings of the 8th IEEE Asia-Pacific Services Computing Conference, Pages 355-360, Yilan, Taiwan, CHN, Dec. 9-12, 2008

[12] Weiyu Chen, Jingjing Wu, Weiwei Sun, Zhenying He. A Location-Based Execution Path Selection for Composite Service in MANETs. Proceedings of the 9th International Conference for Young Computer Scientists, Pages 533-538, Zhang Jia Jie, Hunan, CHN, Nov. 18-21, 2008.

[13] Xi Zhou, Junshuai Shi, Yingxiao Xu, Yinsheng Li, Weiwei Sun. A Backup Restoration Algorithm of Service Composition in MANETs. Proceedings of the 11th IEEE International Conference on Communication Technology, Pages 588-591, Hangzhou, CHN, Nov. 10-12, 2008.