A Global QoS-Aware Service Composition in Wireless Sensor Networks

V.Vanitha
Assistant professor
Kumaraguru college of Technology,
Coimbatore,Tamilnadu,India

Dr.V.Palanisamy
Principal,
Info Institute of Engineering
Coimbatore,Tamilnadu,India

ABSTRACT

Wireless sensor networks (WSN) present a challenging programming environment because of their limited resources, heterogeneity and highly dynamic nature. Service oriented computing (SOC) can simplify application development by hiding platform-specific capabilities behind services. Their services are dynamically discovered and used at run time, enabling application to be platform independent and adapt to network dynamics. While service-oriented computing is widely used on the internet, adapting it to WSNs is non-trivial due to the extremely limited resource available. The selection of which service provider to use and how to adapt as provider change significantly impacts application and network performance. In this paper, we present a global QoS optimizing and multiobjective service composition algorithm based on the construction of convex hull. Simulation experiments were conducted to show the efficiency of the proposed algorithm.

Keywords: WSN, SOC, QoS, service composition.

1. INTRODUCTION

A Wireless sensor network (WSN) is composed of a large number of sensor nodes that are densely deployed either inside the phenomenon or very close to it. The mainobjective of the wireless sensor networks is to observe an environment, collect information about the observed phenomena or events and deliver this information to the application.

The wireless micro sensor node consists of a sensing module, a processing element, and communication elements. The sensing module is an electrical part detecting physical variable from the environment. The processing unit (a tiny microprocessor) performs signal processing functions, i.e. integrating data and computation required in the processing of information. The communication elements consist of a receiver, a transmitter, and an amplifier if needed. Basically, all individual sensor nodes are operated by a limited battery, but a base station node as a final data collecting center can be modeled with an unlimited energy source. Nodes communicate wirelessly. Each node communicates directly (i.e., single hop) with a few other nodes within its radio communication range. A node may also transmit to distant nodes through multi-hop communication.

Sensors may be utilized in seismic, low sampling rate magnetic, thermal, visual, infrared, acoustic, and radar. Sensors may monitor temperature, humidity, vehicular movement, lighting condition, pressure, soil makeup, noise levels, presence/absence of certain kind of objects, mechanical stress levels on attached objects, and current characteristics of objects (speed, direction, and size). So that, the main applications of wireless sensor networks would be in health, military, security, home, environmental, and commercial.

Currently, sensor network architectures are tailored to specific applications with the intention of optimizing the sparse available resources, especially in terms of memory and battery. However, these approaches prevent the adaptation to time-specific operation requirements, the reuse of software components as well as the interoperability between different networks whose sensing range overlaps, thus boosting development costs. Such issues have already been solved in the last years in the field of enterprise information systems by SOA, which has been proven to support more effectively the requirements of business processes and users.

Service oriented architecture (SOA) is an architectural paradigm where a system is decomposed into smaller parts (components) which are able to provide certain functionality by exposing a number of services. Service oriented architecture is a loosely coupled approach that encourages component reuse. It provides developers with the ability to easily build composite applications and to dynamically discover and use services. SOA is a particularly popular paradigm in the community of the web software developers. WSNs can be viewed in part as a reduced copy of Internet, where different nodes or their groups provide different services to the end user. Therefore SOA can be tried out in WSN domain. By implementing a service oriented approach at all levels of WSN, the rapid development of applications as well as the thorough testing of sensor networks will be possible. Also service-oriented approach provides adequate abstractions for application developers, and that it is a good way to integrate the Internet with WSN.

In service oriented WSN, a service is a unit of runtime software that is accessible by others. One node can deploy more than one service while one service may be deployed on many nodes. Service composition is the process of assembling independent, reusable service components to construct richer application functionality. That is, the task of service composition is to assign each required service to an appropriate service provider according to certain criteria. If there is more than one service provider providing the similar or identical function to the requested task then service selection is needed. Here, the purpose of service selection is for service composition. QoS-aware service selection refers to the problem of selecting a set of appropriate services to instantiate composition logic while satisfying user's QoS requirements.

This work proposes a global QoS-aware multiobjective service selection algorithm for service composition. This algorithm optimizes several objectives simultaneously. Simulations were conducted and the experimental results were presented.

The rest of the paper is organized as follows.
Section 2 discusses related works. Section 3 introduces QoS aware service composition. Section 4 formulates the global service selection algorithm. Section 5 presents experimental results to show the effectiveness of the algorithm and section 6 concludes the paper.

2. RELATED WORK

Recently, the Service Oriented Architecture (SOA) has been considered as a good candidate to develop open, efficient, inter-operable, scalable and customizable WSN applications. In Service Oriented WSNs, node's sensing capability is exposed in the form of in-network services. Application development is simplified by providing standards for data representation, service interface description, and service discovery facilitation. By wrapping application functionality into a set of modular services, a programmer can then specify execution flow by simply connecting the appropriate services together. Some approaches are TinySOA [13], OASiS [11] and TinyWS [12]. In TinySOA, services are lightweight code units deployed directly on top of the operating system of nodes. Applications invoke services using a service-oriented query model. Queries are submitted to one of the established base stations or directly to individual nodes. OASiS also uses a passive discovery mechanism, but it is combined with an object migration approach instead of using remote query mechanisms. Finally, TinyWS is a small web service platform that resides on the sensor nodes. It hosts the web services and has SOAP processing engine. The sensor nodes are service providers, the application devices are service requestors and a distributed UDDI acts as an overlay entity.

Service composition with various performance metrics [3], [4], [5], e.g., load balance, end-to-end delay and resource, have been well studied. Service composition in WSNs has also recently been studied in [6], [7]. [6] Studies the minimum-cost service placement based on service composition graphs with tree structure. [7] Considers the optimal placement of filters (services) with different selectivity rates. [10] Considers service composition for persistent queries. It makes use of dynamic programming to reduce the transmission cost. Dynamic programming is suitable for small scale problems. For large scale problems the worst case time complexity of dynamic programming will increase exponentially. [14] Presents multi-dimensional multi-choice knapsack problem (MMKP), a variant of the classical 0-1 knapsack problem

3. QOS-AWARE SERVICE COMPOSITION

Now we focus on the service selection problem. Consider a composite service containing *n* tasks, *t1, t2,…, tn*. For each task t_i $(1 \leq i \leq n)$, there are *li* candidate services that can perform the task. Users may impose some constraints on the amount of various resources to be consumed (e.g. execution time, price). Let's assume that *m* resources *(r1, r2, …, rm)* are constrained. Then, the QoS-based service selection problem involved in service composition is, in fact, how to select one service for each involving task from its corresponding existing candidate service group, so that the overall QoS of the constructed composite service can be maximized while the constraints set by users are satisfied.

Based on different constraints users set, these selection problems can be divided into three categories: 1) QoS optimization problem without global constraints; 2) QoS optimization problem with a single global constraint; 3) QoS optimization problem with multiple global constraints. In the first category, users don't set any global constraints on QoS. So only a set of services need to be selected to maximize the overall QoS of the composite service. In the Second category Users often lay one global constraint on the QoS that they most care, such as "the total execution time of the composite service should be no longer than 90 seconds". Accordingly, the objective is to select one service from each task's candidate service group to obtain a complex service that meets this QoS requirement yet maximize the overall QoS. This is similar to a multi-choice knapsack problem, in which items are grouped and from each group only one item should be chosen to be contained in knapsack. If we map a task to a group, map selecting an actual service for a task to picking an item from a group, and also map a resource to a knapsack, a QoS optimization problem with one global constraint can then be seen as a multi-choice knapsack problem. In the third category users impose two or more global constraints on QoS, it is needed to select one service from each task's candidate service group to form a composite service that meets these constraints yet maximize the overall QoS. Similar to above, if a task is mapped to a group, selecting an actual service for a task is mapped to picking an item from a group, and multiple resources mapped to multi-dimensional knapsack, a QoS optimization problem with multiple global constraints can be modeled as a multi-dimension multi-choice knapsack problem.

Greedy Algorithm can be adopted to solve the problem of composite services selection with no global constraint. Just use QoS scores as the selection criterion. For each task, compute the scores of all its candidate services, and then designate the service which scores highest to it. To solve the QoS-based service selection problem for composite services with one global constraint, the most direct algorithm is to enumerate all possible execution plans of the composite service, compare their QoS

scores, and select the plan which maximizes the overall QoS while satisfying the imposed constraint. Obviously, the computation cost of this algorithm is high. Its time complexity is $O(L^n)$, so it cannot be used to solve large scale problems. As discussed already, QoS optimization problem with one global constraint can be mapped to a multi-choice knapsack problem. Approaches to a multi-choice knapsack problem can be divided into two categories: one is using exact algorithms to get exact optimal solutions; the other is using approximation algorithms to get near-optimal solutions. Finding exact solutions is NP-hard. Dynamic programming is a common approach to compute the optimal solutions to the problem. It is suitable to be used to solve some small-scale online service selection problems which don't require high precision of the solution. However, with the increasing scale of problems, the worse-case time complexity of dynamic programming will increase exponentially.

A QoS optimization problem with multiple global constraints can be modeled as a multidimension multi-choice knapsack problem. It is also an NP complete problem. Similar to the last subsection, we use a heuristic algorithm to derive near-optimal solutions. The idea in it is similar, but there is some difference in that an additional transformation is needed here to transform multiple resources into one single dimension, since multiple constraints have been imposed by user.

We used the network architecture proposed in [10]. We assumed all the information about the QoS parameters are available in the service layer. Some of the Quality parameters for wsn are listed in the table 1 below.

Parameter	Definition
Latency	$Time_{process}(op)$ + $Time_{results}(op)$ where $Time_{process}$ is the time to process *op* and $Time_{results}$ is the time to transmit/receive the results
Reliability	$N_{success}(op)/N_{invoked}(op)$ where $N_{success}$ is the number of times that *op* has been successfully executed and $N_{invoked}$ is the total number of Invocations
Availability	UpTime(*op*)/TotalTime(*op*) where UpTime is the time *op* was accessible during the total measurement time TotalTime
Fee	Dollar amount to execute the operation
Residual energy	The remaining energy of a node in the wireless sensor network. Residual energy must be greater than energy needed for the service execution.

Table 1 QoS Parameters

4. CONVEX HULL ALGORITHM FOR SERVICE SELECTION

In this paper, we propose to use an efficient heuristic algorithm to compute the near-optimal solutions, which leverages the idea of constructing the *convex hull* of related points for approximation. A convex hull of a set S of points in the plane is defined to be the smallest convex polygon containing all the points of S. A polygon is defined to be convex if for any two points p1 and p2 inside the polygon, the desired line segment from p1 and p2 (denoted <p1,p2>) is fully contained in the polygon. The vertices of the convex hull (called as extreme points) of a set S of points form a subset of S. There are two different groups of line segments in the convex hull connecting the bottommost and the topmost points. Each of the groups of line segments are called frontier of the convex hull. QuickHull and Graham's scan are well known algorithms for the construction of convex hull.

Assume the x-coordinate represents resource consumption; the y-coordinate represents QoS score. Then every actual (candidate) service can be mapped to a point in the two-dimensional space which has an x-coordinate indicating the resource consumption of the service and a y-coordinate indicating its QoS score. So each task in the composition corresponds to a set of points in the space. If there are *n* tasks involved in the composition model, there will be *n* sets of points, each containing *li* points respectively. The objective of the optimized selection is to pick up exactly one point from each set of points so that the total resource consumption represented by the selected *n* points can meet the user's requirements while maximizing their overall QoS value. To achieve this, the convex hulls of those sets of points are respectively constructed (through use of algorithms like Graham-scan or Quickhull). *N* efficient convex hull frontiers are computed. Then the gradients of these segments can be used as heuristic information guiding our point selection. The algorithm is presented as *serviceselection* in the following.

Algorithm serviceselection

1. Initialize current solution vector
2. Apply transformation technique to map the multidimensional resource consumption into a single dimension using the vector transformer.
3. Construct convex hull.
4. Find efficient convex hull frontiers.
5. Find the gradient of all the segments in the effcient convex hull frontiers.
6. Sort all the segments in the frontier in descenting order according to the gradient of each segment.
7. For each segment calculate the total QoS score. If this new score is greater than the current score then update the solution vector and update the tranformer, otherwise ignore the new QoS score.
8. Repeat the process and output the final solution.

In the algorithm above, transformer is an m-dimensional vector used to transform the *m* constrained resources to one dimension, in other words, to give an overall cost for each candidate service. If transformer is (q(1),q(2), …, q(m)) and the constrained resource vector is (r(1),r(2), …, r(m)), the

transformed resource vector will be (r(1)q(1), r(2)q(2), …, r(m)q(m))=(rt1,rt2, …, rtm) which can then be turned into one dimension. The formula used to initialize transformer can be q(k) = rs(k)/R(k) +1 (k=1,2,…,m), where *R* represents the resource vector of user requirements, and *rs* represents the vector summation of resource consumption vectors of every candidate service for every task. Resources in higher demand are assigned higher transformers. The algorithm takes the gradients of the segments in efficient convex hull frontiers as the heuristic which helps to obtain the near-optimal solution to the problem. The efficient convex hull is the frontier which earns more QoS score and gradient is a vector that always points in the direction of maximum change, with a magnitude equal to the slope of the tangent to the curve at the point. "*Sort all the segments in frontier_segments in descending order according to the gradient of each segment*" implies that the segments listed in the front would have greater gradients, which means the benefit gained consuming a unit of resource. So the solving process starts with the points that yield greatest gradient, and tries to go towards the direction that leads to a higher QoS score yet not breaking the resource constraint. Each time the solution vector is successfully updated, the total QoS score will rise. Note that in certain cases such that the resource constraint set by user is upon execution time, not the resource like network bandwidth, there are often some services with higher scores consuming less resource than those with lower scores, that is to say, they provide better QoS with less resource consumption. So, in the algorithm, we sort the segments in descending order according to the angle between the segment and the positive direction of x-axis, in order that their corresponding segments are listed in the front for prior consideration of selection. The method *adjust_transformer()* is used to update transformer.

5. SIMULATION RESULTS

We implemented the algorithm using VC++. For simplicity of the implementation we assumed every task in the composition has equal number of candidate services. Given the total task number *n* and the candidate number for each task *L*, the simulation program can randomly generate QoS matrix in a specified range. The figure 1 shows the time efficiency of our algorithm with respect to number of candidate services for each task and figure.2 shows the time efficiency with respect to respect to number of tasks.

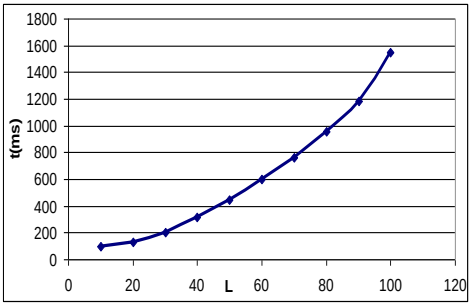

Figure 1. Time performance of serviceselection with increased L

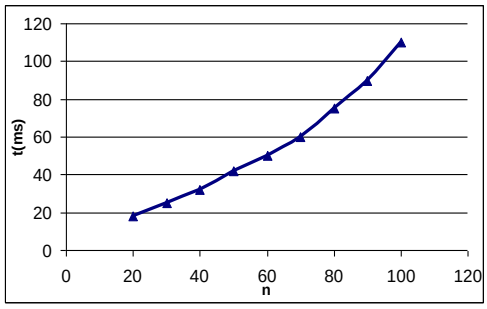

Figure 2. Time performance of serviceselection with increased n

6. CONCLUSION

In this, we explore the problem of service composition in WSN. The QoS based service selection problem for composite services actually is how to select one service for each task from its corresponding service group, so that the overall QoS of the constructed composite service can be maximized while the constraints set by the user are satisfied. A simulation result proves the efficiency of the proposed algorithm.

7. REFERENCES

[1] A. Rezgui and M. Eltoweissy, "Service- Oriented Sensor-Actuator Networks," in IEEE Communications, Volume 45, No. 12, pages 92-100, December 2007.

[2] D. Graˇcanin, M. Eltoweissy, A. Wadaa, and L. DaSilva, "A service-centric Model for Wireless Sensor Networks" in JSAC, Vol. 23, No. 6, pp. 1159- 1166, June 2005

[3] B. Raman, R. H. Katz, "Load balancing and stability issues in algorithms for service composition", in IEEE InfoCom 2003, pp.1477-1487.

[4] J. Jin, K. Nahrstedt, "Source-based QoS service routing in distributed service networks", in ICC2004, pp.20-24, June 2004.

[5] X. Gu, K. Nahrstedt, R. Chang, C. Ward, "QoS-assured service composition in managed service overlay networks", in ICDCS 2003, pp.19-22,May 2003.

[6] Z. Abrams and J. Liu, "Greedy is good: On service tree placement for in-network stream processing", in Proc. of ICDCS, 2006.

[7] U. Srivastava, K. Munagala, and J. Widom, "Operator placement for innetwork stream query processing", in PODS'05, Baltimore, MD, June 2005

[8] R. Ha, P.-H. Ho and X.S. Shen, "Cross-Layer Application-Specific Wireless Sensor Network Design with Single-Channel CSMA MAC over Sense-Sleep Trees", accepted by Elsevier Journal: Computer Communications.

[9] J. Wang, D. Li, G. Xing, and H. Du, "Cross-layer Sleep Scheduling Design in Service-Oriented Wireless Sensor Networks", Technical report, City University of Hong Kong, 2007.

[10] Xiumin Wang, Jianping wang, Zeyu Zheng, Yinlong Xu, Mej Yang, "Service composition in service oriented Wireless sensor networks with persistent queries", In Consumer communications and Networking Conference, 2009. CCNC 2009. 6th IEEE.

[11] M. Kushwaha, I. Amundson, X. Koutsoukos, S. Neema, and J. Sztipanovits. OASiS: A Programming Framewor for Service-Oriented Sensor Networks. In Proceedings of the 2nd IEEE/Create-Net/ICST International Conference on COMmunication System softWAre and MiddlewaRE

(COMSWARE' 07), Bangalore, India, January 2007. IEEE Computer Society Press.
[12] N. Y. Othman, R. H. Glitho, and F. Khendek. The Design and Implementation of a Web Service Framework for Individual Nodes in Sinkless Wireless Sensor Networks. In Proceedings of the IEEE International Conference on Computers and Communications (ISCC'07), pages 941–947, Aveiro, Portugal, July 2007. IEEE Computer Society Press.
[13] A. Rezgui and M. Eltoweissy. Service-oriented sensor actuator networks: Promises, challenges, and the road ahead. Computer Communications,30:2627–2648,2007.
[14] M. M. Akbar, M. S. Rahman, M. Kaykobad, et al. "Solving the multildimensional multiple-choice Knapsack Problem by constructing convex hulls," Computers and Operations Research, 2006, vol. 33, pp.1259-1273, 2006.