ReSCo: A Middleware Component for Reliable Service Composition in Pervasive Systems

Brent Lagesse
Cyberspace Sciences and Information Intelligence Research
Oak Ridge National Laboratory
Oak Ridge, TN, USA
Email: lagessebj@ornl.gov

Mohan Kumar and Matthew Wright
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX, USA
Email: {mkumar, mwright}@uta.edu

***Abstract*—Service composition schemes create high-level application services by combining several basic services. Service composition schemes for dynamic, open systems, such as those found in pervasive environments, must be cognizant of the possibility of failures and attacks. In open systems, it is seldom feasible to guarantee the reliability of each node prior to access; however, there may be several possible ways to compose the same high-level service, each having a different (though possibly overlapping) set of nodes that can satisfy the composition. We approach this problem with a Reliable Service Composition middleware component, ReSCo, to determine trustworthy compositions and nodes for service composition in dynamic, open systems. ReSCo is a modular, adaptive middleware component that selects from possible composition paths and nodes to enhance reliability of service compositions. ReSCo can work with a broad range of both service composition algorithms and trust establishment mechanisms.**

***Keywords*-Dynamic Service Composition; Reliability; Security; Adaptive Systems**

I. Introduction

Service composition is the process of combining available low-level services in a system to create an application(or high)-level service. In pervasive systems, the low-level services that make up an application-level service may be available on nodes with variable connectivity or availability. While nodes that become unavailable can make it harder to compose a service, new nodes may become available that increase the possible composition possibilities. Adding further to the complexity, multiple compositions for a single high-level service may exist. For example, some services may accept or produce a wide variety of data formats, leading to a range of paths that might be possible, even when only a few data transformations are performed.

Ideally, service composition in mobile and uncertain environments should be: i) adaptive to the dynamic nature of connections between mobile nodes, so that it can overcome loss of services due to disconnecting nodes and exploit newly available services due to incoming nodes; and ii) cognizant of the trustworthiness and reliability of nodes.

An example of a service composition system, which we will use as a motivating example in this paper, is Collaborative Virtual Observation (CoVO). CoVO uses service composition to complete tasks that allow users to recreate and analyze an environment. For example, CoVO could be used to enable the emergency responders to conduct real-time monitoring, such as for a natural disaster in a populated area, using services available in an ad-hoc network. While there is not necessarily any single service that will allow the responders to accomplish this, embedded in the environment are traffic cameras, video cameras, microphones, and other such devices. Furthermore, the responders have access to a variety of software services available to them to perform tasks such as fusing information, converting information formats, and encrypting information. In such a situation the responders can compose high-level services by gathering information from camera and microphone services and fusing them together. The audio service can be connected with an audio to text converter and the resulting text file can be cross linked with other information or even translated if voices of those in danger are speaking in other languages. Finally, the resulting information can then be encrypted prior to transmission in order to enhance privacy and security. As one can see from the example, the composed high-level service has many possible points where it can break if attacked or if a node fails. There are also many possible ways to compose a satisfactory high-level service. The goal of ReSCo is to provide a mechanism for utilizing possible paths and nodes to compose reliable high-level services.

Service composition frameworks employ a variety of practices to build high-level services from the basic services available in the environment. Some of these techniques require exact matching of services [1], [2], [3], and others perform dynamic matching [4]. Existing service composition frameworks evaluate composition choices without considering the trustworthiness of nodes providing the available low-level services or through mechanisms that do not scale well to open and dynamic systems.

ReSCo abstracts the service composition problem into two sub-problems: composition path selection and node selection along a chosen path. ReSCo utilizes local experience information and reputation information to identify unreli-

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2010	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
ReSCo: A Middleware Component for Reliable Service Composition in Pervasive Systems	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Cyberspace Sciences and Information Intelligence Research Oak Ridge National Laboratory Oak Ridge, TN, USA**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADA539422. IEEE International Conference on Pervasive Computing and Communications (8th) (PerCom 2010) Held in Mannheim, Germany on March 29-April 2, 2010. U.S. Government or Federal Purpose Rights License., The original document contains color images.

14. ABSTRACT
Service composition schemes create high-level application services by combining several basic services. Service composition schemes for dynamic, open systems, such as those found in pervasive environments, must be cognizant of the possibility of failures and attacks. In open systems, it is seldom feasible to guarantee the reliability of each node prior to access; however, there may be several possible ways to compose the same high-level service, each having a different (though possibly overlapping) set of nodes that can satisfy the composition. We approach this problem with a Reliable Service Composition middleware component, ReSCo, to determine trustworthy compositions and nodes for service composition in dynamic, open systems. ReSCo is a modular, adaptive middleware component that selects from possible composition paths and nodes to enhance reliability of service compositions. ReSCo can work with a broad range of both service composition algorithms and trust establishment mechanisms.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**SAR**	**6**	

able nodes and adapts to select reliable nodes more often. Because of its modular design, ReSCo is independent of the service composition framework that decides which high-level services are composable. This means that ReSCo can provide greater reliability for both existing and future service composition system. In technology-rich environments, ReSCo can take advantage of redundant services and adapt to select the most reliable nodes. Even in more sparsely-populated systems, though, ReSCo benefits the user.

II. Design

The objective of ReSCo is to minimize the effect of unreliable and malicious nodes in a service composition system. To accomplish this, a ReSCo instance running on node u keeps track of u's experiences with other nodes in the system in an *experiences database*. In this way, ReSCo can serve node u without relying on collaboration with any other nodes to achieve its goals.

Input: N, π, Nodes
Output: Ordered List of Nodes to Access
Compute path probabilities (Equation 5);
Select path;
foreach *ServiceInPath* s **do**
    Compute node probabilities for s (Equation 2);
    Select node and add to access set;
end

Algorithm 1: Selecting the Node Set to Access

Input: π, Set of Requests
foreach *request* r **do**
    Create Access Set from π (Algorithm 1);
    Use the service composed by Access Set;
    Evaluate Result;
    Update evaluation information to database;
end

Algorithm 2: Composing a Service

Algorithm 2 describes the process of selecting the set of nodes to be used for the construction of a high-level service. More precisely, Algorithm 2 shows the *default implementation* of ReSCo, in which ReSCo adapts to the reliability of nearby nodes to select trustworthy nodes more often.

ReSCo provides reliable path and node selection as part of a complete service composition system (as shown in Figure 1). ReSCo is designed so that it acts as a layer between the service composition framework and the network. We designed ReSCo to be modular, so that it is independent of the mechanism that determines valid compositions. This allows ReSCo to remain useful as new techniques for creating high-level services continue to improve. This evaluation of ReSCo utilizes SeSCo [4] for determining which high level service compositions are available because its graph-based composition provides a higher number of successful compositions than discover and match techniques.

The ReSCo middleware component consists of four primary parts: a Path Selector, a Node Selector, a Request Evaluator, and an Experience Database. The Request Evaluator determines if a service composition was successful. Ideally this should be an automated process, e.g. based on verifying a public key signature or checking a file format. In some applications, however, the Request Evaluator may rely on a user determining whether or not the result is satisfactory and providing feedback, such as viewing a video and clicking on a "thumbs-up" or "thumbs-down" icon. The Experience Database is a storage location for the results of local experience and the experiences reported from other users (via reputation). The database contains cumulative values based on an application-defined weighting for positive and negative experiences, both local and from reputation values. ReSCo can leverage techniques from other systems, such as AREX [8] to populate the experience database through exploration while increasing protection for the node utilizing it. Section II-D and Section II-E discuss the Path Selector and Node Selector components in more detail and explain the algorithms that underly their operations.

A. Request Evaluator

The ReSCo request evaluator defines two values, α and β that are designed to respond to negative and positive interactions. α is subtracted from a node's experience value if it was involved in an unsuccessful service access and β is added to the node's experience value if the node was involved in a successful service access.

B. Experience Database

The ReSCo database keeps track of the current experience values for the known services. Each database entry contains an identifier for the service and an experience value. Because the experiences are compressed into an scalar value, the individual entries for each node do not consume a significant amount of memory and the database scales linearly with the number of nodes in the system (databases in simulations were typically on the order of 1 KB).

C. Example

An example scenario of how ReSCo is used to compose an application-level service is now presented. For the details on the methodology of the calculations, see Sections II-D and II-E. In this example, the user is trying to compose a service to get a video observation of a remote event (possible compositions, the experience database, and available services are depicted in Figure 1). SeSCo is able to determine three ways to accomplish this task: i) Using a traffic video camera and displaying it on the laptop ii) Using a predefined set of cell phone video cameras in the area of the event and fusing the information to create a comprehensive picture then displaying it on the laptop and iii) Using a predefined set of still photo cell phone cameras in the area of the event and using an animator service to turn the sequential images into a video, then showing the animation on the PDA screen.

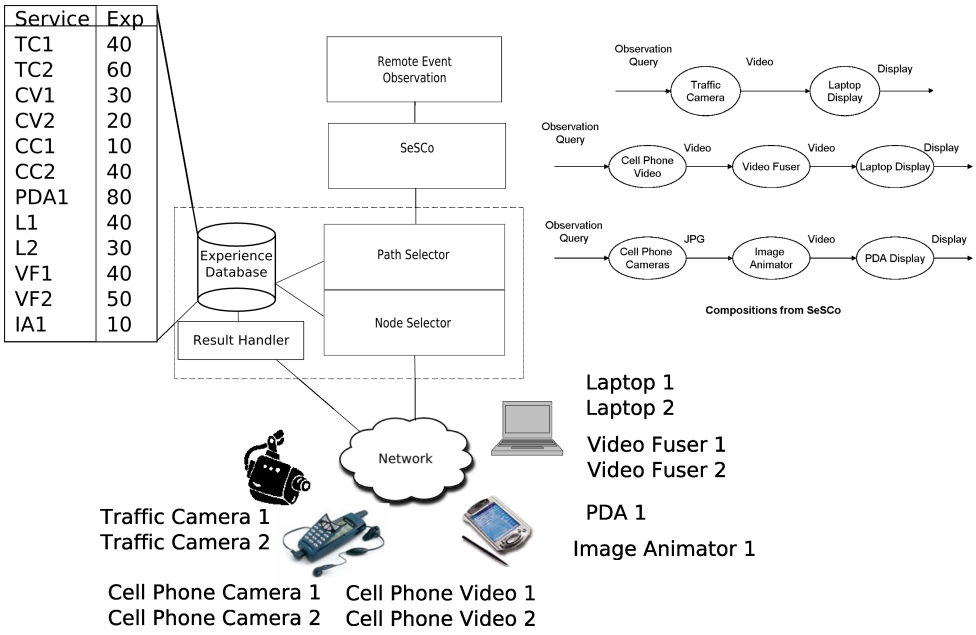

Figure 1. Example Composition Scenario

π	Set of possible paths
$P_{prune}(x, node)$	Probability of pruning a service from database
$Services$	Set of services used to compose paths
$Nodes$	Set of nodes that provide a service
$TE(x)$	Total Experience value for a service x
$N(x)$	Set of values of nodes with service x
$S(x)$	Set of services composing path x
$E_\sigma(x)$	Expected success for a service x
$E_\rho(x)$	Expected success of a path x
P_x	Probability of success for service x
P_σ	Probability of selecting a service
P_ν	Probability of selecting a node
P_ρ	Probability of selecting a path
P_α	Probability that a node attacks
P_s	Probability that a request was successful
t_{conn}	Amount of time a node is connectible
P_{avail}	Probability that a node is available
P_{awake}	Probability that a node is awake
$P_{inRange}$	Probability that a node is in range
α	Punishment for unreliable service
β	Reward for reliable service

Table I
TABLE OF COMMONLY USED TERMS

$S(x)$	$TE(x)$	$E_\sigma(x)$	$E_\rho(x)$	$P_\rho(x)$
TC, L	50, 35	0.19, 0.13	0.024	88.8%
CV, VF, L	25, 45, 35	0.09, 0.17, 0.13	0.002	7.5%
CC, IA, PDA	25, 10, 80	0.09, 0.04, 0.30	0.001	3.8%

Table II
SELECTION RATE FOR PATHS IN EXAMPLE APPLICATION (FIGURE 1)

These possible compositions are passed from SeSCo to ReSCo. ReSCo then looks up services in the experience database and uses the information to perform the composition path calculations (shown in Table II). After the calculations are performed, ReSCo selects a path and then

$N(x)$	$P_\nu(x)$
TC1	40.0%
TC2	60.0%
L1	57.1%
L2	42.9%

Table III
SELECTION RATE FOR NODES IN EXAMPLE APPLICATION (FIGURE 1)

performs the calculations (shown in Table III) to select the nodes to request for the low-level services.

D. Path Selector

A composition path refers only to the types of services that will be used, not the individual nodes that provide those services. To do this, ReSCo randomly selects a path with a probability computed by the portion of the expected reliability of the services that compose each path. The expected reliability is determined by the product of the expected reliability of a service. The reason ReSCo takes the product of the reliability percentage is twofold. Because the values are between 0 and 1, it means that longer compositions will be chosen less often than shorter compositions, unless they are significantly more reliable. Second, it means that if there is a choke point (a service that is significantly less reliable than other services), then that path will be chosen less often. The expected reliability of a service can be described by the expected reliability of the nodes that compose the service (which is discussed in Section II-E).

The service experience is calculated with Equation 1 and is the total sum of the experiences that the node running ReSCo has had with other nodes that provide that particular

service. Table II-B provides a list of commonly used terms.

$$TE(x) = \sum_{i=0}^{|N(x)|} N(x)_i \quad (1)$$

The probability of selecting a node for a service is given by Equation 2. It is equivalent to taking the ratio of the experience values of a node to that of the total experience value of that particular service.

$$P_\nu(x, node) = \frac{N(x)_{node}}{TE(x)} \quad (2)$$

The expected experience value of a service (Equation 3) describes the weighted average of all the nodes that provide a particular service. In particular it is calculated by summing the probability that a node will be selected by the current experience value of that node over all the known nodes that provide the particular service.

$$E_\sigma(x) = \sum_{i=0}^{|N(x)|} P_\nu(x, N(x)_i) \times N(x)_i \quad (3)$$

A path is composed of one or more services. To determine an expected experience value for the entire path (Equation 4), ReSCo takes the product of the expected values of each individual service. While this value has little meaning in isolation, it is used in Equation 5 to calculate the probability of selecting a path. Similar to the computation for selecting a node, a path is selected with a probability equivalent to the ratio of its expected experience value to the sum of the expected values of all possible paths.

$$E_\rho(x) = \prod_{i=0}^{|S(x)|} E_\sigma(i) \quad (4)$$

$$P_\rho(x) = \frac{E_\rho(x)}{\sum_{i=0}^{|\pi|} E_\rho(i)} \quad (5)$$

ReSCo makes the selection randomly instead of only choosing the best path for several reasons. First, it results in a natural load balancing. The most trustworthy paths will obviously receive more of the traffic, but among reasonably trusted paths, the traffic will be balanced relative to their trustworthiness. Next, randomly selecting paths allows us to explore the service-space. This point is particularly important as ReSCo is designed for dynamic environments. Since nodes may be mobile (both the node accessing services and the service nodes themselves), the best service selected by a composition scheme may not be on a static node and may vary significantly throughout the time in the system and random selection allows us to transition as the available system changes.

E. Node Selector

Once a path has been chosen to construct, ReSCo must then choose the individual nodes that will provide the services that are required to construct the high-level service. The selection process is similar to that of Path Selection. The node that will be accessed for each service is selected randomly with the probability equal to the proportion of its contribution to the total experience value (Equation 1) for that service as described in Equation 2.

Number of Simulation Executions	200
Number of Peers	100
Average Benign Reliability	95%
Execution Time (seconds)	1000
Attack Rate of Malicious Peers	100%

Table IV
DEFAULT SIMULATION PARAMETERS

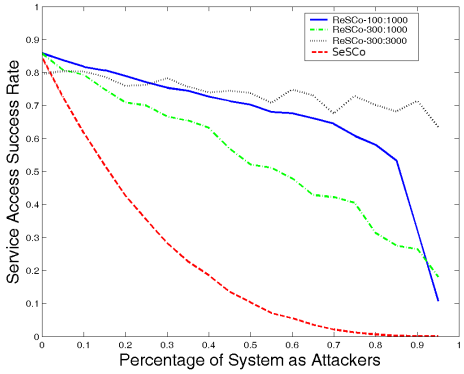

Figure 2. Effect of Attackers on Success (Approach:NumPeers:NumIterations)

Algorithm 2 is the operating procedure for the architecture shown in Figure 1. The composition layer presents the ReSCo layer possible compositions for a request, then ReSCo computes which composition to use and which nodes to access for each service. After the services have been accessed, ReSCo adds any available evaluation information from the result to the experience database.

III. EVALUATION

A. Simulation Setup

The simulator is designed to take an input consisting of the possible paths that can be used to create a service, and then create that service from the underlying nodes available in the network. For most simulation experiments, ReSCo is simulated and compared with a system in which all parameters are the same except that the service path and the nodes that compose it are chosen randomly from the set of available paths and available nodes that can successfully fulfill the composition request. Where relevant, ReSCo is compared to the default behavior of the service composition system that does not use ReSCo. The simulation results are grouped into several sections. This section examines ReSCo's response to varying numbers of attackers and unreliable nodes, how ReSCo adapts over time, the effect of differing composition requirements on ReSCo, the effect of mobility on ReSCo, and the effect of augmenting ReSCo with reputation.

1) Attackers and Unreliable Nodes: The first experiment is designed to show the resiliency of ReSCo against an

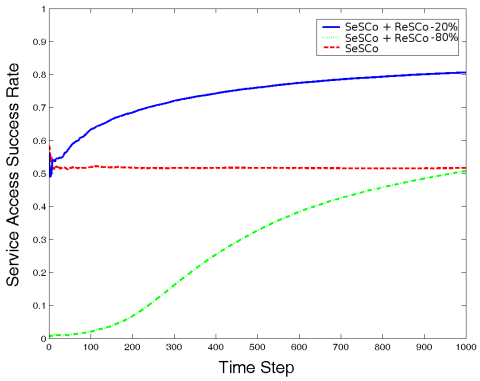

Figure 3. Cumulative Success at Each Timestep

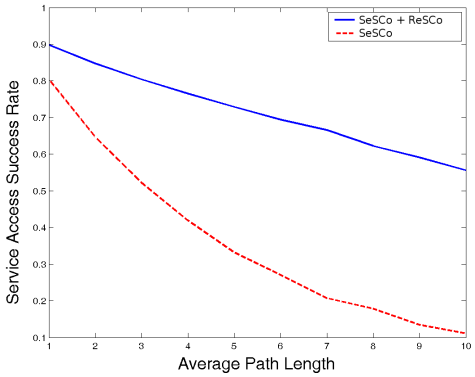

Figure 4. Effect of Path Length on Success

increasing number of attackers in a system. Figure III-A1 shows that as the number of attackers increases, random selection shows an exponential decrease in the percentage of successful high level services it can access. ReSCo naturally decreases in the percentage of successful constructions accessed, but is successful over half the time as long as less than 85% of the system is attacking. The reason for the sharp decrease after this point in success is not a failure of ReSCo, but rather is a result of the fact that when simulating 100 peers in a system, there exist some cases in which there is no possible way to construct a path out of trustworthy nodes. To illustrate this, consider a system in which is trying to compose a high level service from two possible paths consisting of three services, $\{A, B, C\}$ and $\{D, E, F\}$. If 95% of the system is attacking and there are 100 nodes in the system, then there will be 5 nodes that are benign. This means there there may be no benign paths possible. To demonstrate that the effect is not present in larger systems, Figure III-A1 also shows ReSCo in a system with 300 nodes. Since more nodes are included, it takes more time to adapt, so a 300 node simulation plot is shown after 3000 time-steps (Labeled ReSCo-300:3000) in addition to the default of 1000. After this additional time is used, the adaptation produces favorable results that do not include a sharp decrease in performance with high levels of attackers.

2) Adaptation over Time: This section shows the adaptation of ReSCo as time progresses. Figure 3 shows the cumulative success rate up to each given time step in a system with 20% attackers. The result shows how a standard setup of the system progressively adapts to its environment in order to provide better service over time. Also the results show user expected success rate over a given time. The third plot in Figure 3 shows ReSCo's cumulative success in a system that has 80% attackers and that after 100 time steps, ReSCo has had the same success as the naive approach in a system with 20% attackers.

3) Path Length: The length of a path can have significant effect on the ability to successfully create and access a high-level service. As the number of services needed to create a composed service increases, the opportunity for a single node in the service to fail increases. The effect is clearly shown by the exponential decrease of the Random Selection curve in Figure III-A1 with 20% of the system as attackers. Through its adaptive selection algorithm, ReSCo is able to mitigate this problem and still successfully complete requests 55% of the time, even when the requests are 10 services long.

IV. Related Work

Significant research has recently taken place in service composition. While much of the work in service composition has been focused on composing web-services, there has also been some work in the dynamic environments encountered in pervasive computing.

Composition Trust Bindings [9] verify the integrity of composed services using the service composition equivalent of digitally signed software. In contrast, ReSCo is concerned with dynamic compositions of potentially unknown nodes, services, and composition paths.

SAHARA [10] is a service composition framework that provides authorization control based on local rules and credentials from other domains. SAHARA also relies on a central Authentication, Authorizing, and Accounting (AAA) server and a requirement for credentials that are not feasible in our environment.

Bartoletti, et al. [11] model service composition with security constraints using an extension of λ-calculus. This approach requires a statically determined abstraction of service behavior which may not be feasible in our environment. In particular, our system should not need to be aware of what services will be available in an environment, the nodes that will be providing these services, and the method of composition until run-time.

QUEST [12] is a service composition infracture designed to assure QoS constraints. While QUEST does handle multiple nodes providing each lower-level service, it does not address multiple different paths that could be used to achieve a compositions.

Jiang, et al. [13] focus on reliable service composition in mobile ad-hoc networks (MANETs). The work is focused on minimizing MANET disruptions in the service composition process. The system is viewed in two tiers, a service layer and a network layer. A dynamic programming solution and a heuristic-based solution (to loosen the requirements of the dynamic programming solution) are presented to provide both network-level and service-level recoveries.

Prior work lacks the flexibility and adaptability required for service composition in dynamic systems.

V. Conclusion

ReSCo is a lightweight, modular middleware component for increasing the reliability of service composition in open, dynamic, pervasive systems. Through stochastic selection that adapts based on personal and reported experiences, ReSCo provides reliability to service composition systems despite the presence of unreliable nodes and attackers. ReSCo provides secure service compositions for systems like CoVO whilst remaining flexible enough to operate in an open and dynamic environment. ReSCo also is lightweight enough to operate on a wide variety of low-resource devices such as sensor and PDAs.

This paper presented simulation results that illustrate the effectiveness of ReSCo. In particular, ReSCo shows significant improvement in reliability over the naive approach against attackers (about 700% improvement when half the system attacks), erroneous entities (about 500% improvement when nodes make errors 50% of the time), and in mobile environments (nearly 28% improvement when all nodes are mobile).

Acknowledgments

The work presented in this paper was partially supported under US National Science Foundation Grant ECCS-0824120.

Notice

References

[1] D. Chakraborty, F. Perich, A. Joshi, T. W. Finin, and Y. Yesha, "A reactive service composition architecture for pervasive computing environments," in *PWC '02: Proceedings of the IFIP TC6/WG6.8 Working Conference on Personal Wireless Communications*. Deventer, The Netherlands, The Netherlands: Kluwer, B.V., 2002, pp. 53–62.

[2] X. Gu, K. Nahrstedt, and B. Yu, "Spidernet: An integrated peer-to-peer service composition framework," in *HPDC '04: Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 110–119.

[3] J. Robinson, I. Wakeman, and T. Owen, "Scooby: Middleware for service composition in pervasive computing," in *MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*. New York, NY, USA: ACM, 2004, pp. 161–166.

[4] S. Kalasapur, M. Kumar, and B. Shirazi, "Dynamic service composition in pervasive computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 7, pp. 907–917, 2007.

[5] L. Xiong and L. Liu, "Building trust in decentralized peer-to-peer electronic communitties," in *International Conference on Electronic Commerce Research*, 2002.

[6] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in P2P networks," in *WWW*, 2003, pp. 640–651.

[7] K. Walsh and E. G. Sirer, "Experience with an object reputation system for peer-to-peer filesharing," in *NSDI*. USENIX, 2006.

[8] B. Lagesse, M. Kumar, and M. Wright, "AREX: An adaptive system for secure resource access in mobile P2P systems," in *Eighth International Conference on Peer-to-Peer Computing*. IEEE Computer Society, 2008, pp. 43–52.

[9] J. Buford, R. Kumar, and G. Perkins, "Composition trust bindings in pervasive computing service composition," in *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops*. IEEE Computer Society, 2006.

[10] B. Raman, S. Agarwal, Y. Chen, M. Caesar, W. Cui, P. Johansson, K. Lai, T. Lavian, S. Machiraju, Z. M. Mao, G. Porter, T. Roscoe, M. Seshadri, J. S. Shih, K. Sklower, L. Subramanian, T. Suzuki, S. Zhuang, A. D. Joseph, R. H. Katz, and I. Stoica, "The SAHARA model for service composition across multiple providers," in *Pervasive '02: Proceedings of the First International Conference on Pervasive Computing*. London, UK: Springer-Verlag, 2002, pp. 1–14.

[11] M. Bartoletti, P. Degano, and G. L. Ferrari, "Enforcing secure service composition," in *CSFW '05: Proceedings of the 18th IEEE workshop on Computer Security Foundations*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 211–223.

[12] X. Gu, K. Nahrstedt, R. N. Chang, and C. Ward, "QoS-assured service composition in managed service overlay networks," in *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2003, p. 194.

[13] S. Jiang, Y. Xue, and D. Schmidt, "Minimum disruption service composition and recovery in mobile ad hoc networks," in *Computer Network Journal, Special Issue on Autonomic and Self-Organizing Systems*, 2008. [Online]. Available: http://www.truststc.org/pubs/442.html