Critical Aware Community Based Parallel Service Composition Model for Pervasive Computing Environment

P. Kumaran and R. Shriram

Department of Computer Science and Engineering,
B.S. Abdur Rahman University, Chennai, Tamil Nadu, India
p_kumaran@hotmail.com,
shriram@bsauniv.ac.in

Abstract. Service composition in pervasive computing environments is needed to provide best quality of service. Services need to be discovered at run time and composed together for best possible user scenarios. The need for identifying the best services among the service nodes is essential in pervasive computing systems as the environment of operation can change rapidly. Pervasive computing demands systems that are scalable, adaptive, fault tolerant and can work in heterogeneous environments. Hence an adaptive method that takes into account the environment is the need of the hour. In this work, a dynamic parallel composition model to compose the best matched services is proposed for the pervasive computing environment exhibiting the quality of service and contingency management properties. The model ensures that the highest quality of service conditions is fulfilled. Facilities for contingency management ensure efficient fault tolerance and failure recovery. The proposed model uses the community framework for grouping the service nodes and composing the services provided by the nodes. This ensures that resultant composition mechanism is dynamic in nature to adapt to the service nodes failure without compromising the quality of service with better fault error recovery time. The model has been validated experimentally and the results show considerable promise. The work is unique in its extensive mechanisms for modeling the pervasive computing environment, failure handling, fault tolerance and best quality of service parameters.

Keywords: Community Manager, Fault Recovery, Pervasive Computing, Service Composition, Service Evaluation.

1 Introduction

Pervasive Computing is a vision of the world where the computing happens anywhere and at any time. Service Composition is the process of composing the unit services into an integrated service to provide end user requirement. Service oriented architecture relies on interaction between autonomic loosely couple services which can be composed together for delivery of goals by the users. These services can be expressed in a middleware system. The middleware system composes the services

D. Nagamalai, E. Renault, and M. Dhanushkodi (Eds.): PDCTA 2011, CCIS 203, pp. 112–121, 2011.

dynamically for accomplishing the goals. Service Composition in the pervasive environment is a challenging research problem due to the unpredictable dynamic behavior of the pervasive environment. The biggest challenge is in developing the middleware [2] for the pervasive environment. The services residing in the nodes need to be composed in parallel in the absence of a centralized entity as the environment needs services dynamically. The key objectives of this work are a) to propose a critical ware service composition model to improve the quality of service, b) to optimize the metrics for improved quality of service and c) to demonstrate the experimental prototype that validates our architecture.

The rest of the paper is organized as follows. In section 2, we discuss the related work carried out supporting our service composition model. In section 3, we describe our composition middleware. In section 4, we give the experimental results and analysis. Finally, section 5 concludes our work and gives research directions to the service composition problem.

2 Related Study

There have been many studies on Service composition in Pervasive Computing. In [8], the user preference is integrated with the system preference to evaluate the service model. The proposed work considers the services residing in the node are of equal priority and based on the utility function value, the higher utility value node in the ordered list is selected. We classified the services residing in the node as critical and non-critical service.

In PICO model [11], mobile and static delegents representing camileuns in communities try to use resources as effectively as possible. The challenges due to mobility and heterogeneity in the pervasive environment are addressed by providing transparent, autonomous and continual middleware services. We have adopted the community formation methodologies used in the PICO model to address the node mobility and heterogeneity. The mobile community network [4] is proposed for the interaction of middleware client installed in the mobile devices. The middleware client request contains device profile, network status, personal profile and requested service data.

In [5], graph based service composition mechanism is proposed. The inputs and the outputs of the services are represented as nodes in the graph and the dependencies of each service are represented as edges. The services' input and output are considered as single parameter which is not the practical and adaptive as the real time services have multiple parameters.

In [3], service composition for mobile environments is achieved through the Dynamic Broker selection and Distributed Broker selection based protocols and suggested to incorporate the parallel broker arbitration to handle parallel service flows. Our earlier work [6] focused on parallel service composition middleware model to addresses the pervasive attribute issues.

In [2], survey on Service composition middleware in pervasive environments discussed on the attributes such as Context Awareness, Interoperability, Discoverability, Adaptability, QoS management, Spontaneous Management, Managing Contingencies,

Leveraging Heterogeneous Devices and Security Mechanisms. These are the key design rationale in designing the middleware for the pervasive computing environments. Our work addresses the design rationale such as QoS management, Spontaneous management and managing contingencies effectively.

We incorporated the parallel composition model in this critical aware model to improve the quality of service by reducing the fault recovery time and service composition length. Our work uniquely differs from the existing service composition middleware for pervasive computing in classifying the services as critical and non-critical dynamically based on the community, evaluating the service nodes based on the service value, activating the critical service execution in parallel and handling the fault recovery. Our experimental result shows the better improvement in the fault recovery time and service composition length compared to the existing works.

3 Service Composition Model

The architecture of the proposed critical aware service composition model is shown in Fig 1.

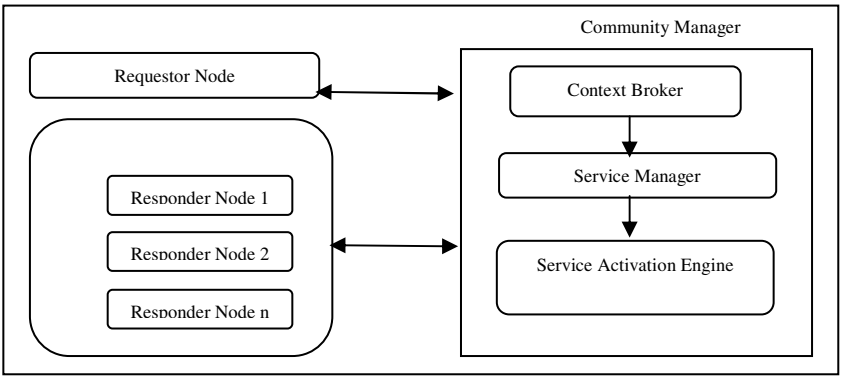


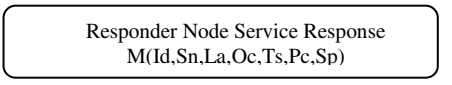
Fig. 1. Service Composition Architecture Model

The Requestor Node is the consumer of the service. Any node in the network can make a request. The Responder nodes are the set of the nodes which receives the service initialize request message sent by the requestor node. The Community manager is responsible for the execution of the service and sent back the required service to the requestor node.

The Community contains the set of nodes actively participating in the service execution. The service manager is responsible for the task coordination. The data communication includes the service initialize request, Community manager response messages and the data transfer between the responder nodes & community manager. The process steps involved in the model are explained in the following sections.

3.1 Service Request Initialization

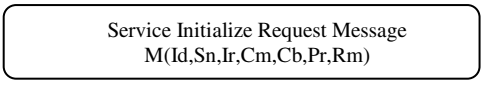
The requestor node initiates the service request by sending the broadcast message. The responder nodes having the exact composite, abstract service or the service link sends back the response to the requestor node.



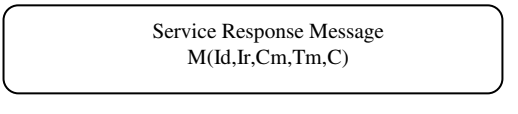
The reply message format of the responder node contains the Service Identifier(Id) to uniquely identify the service, Service Name(Sn) to describe the service, Abstract Level service definition (La) to represent the exact composite (0) or Service abstract(1) or service link(2), Child Object (Oc) to know whether the service requires to complete any pre-requisite services, Terminating Service(Ts) to represent whether the service is the end service, PreCondition for executing the service (Pc), Service Parameters(Sp) containing the list of service level parameter such as failure rates, trust binding values and the task decomposition.

3.2 Community Manager Selection

The service initialize request message contains the Service Identifier(Id) to uniquely identify the service, Service Name(Sn) to describe the service, Intermediate results (Ir), user preferences(Pr) containing the list of user preferences which helps the community manager to maintain the service quality as requested by the requestor node, community manager identifier (Cm) to uniquely identify the community manager, context broker identifier (Cb) to identify the context broker to handle the user inputs and the reply messages(Rm) sent by the responder nodes which contains the list of the information along with the service level information in each responder nodes which can be used as the results of the service discovery.



The Pr contains the list of user preferences which includes the quality of service parameters, task decomposition values, service level agreement between the community manager and requestor node. The requestor node simply acknowledges the message sent by the community manager.



The service response message sent by the community manager contains the service identification(Id) to uniquely identify the service, Intermediate results (Ir), community manager identifier (Cm) to uniquely identify the community manager, timeout (Tm) in milliseconds to denote the validity of the intermediate results sent by the community manager and the criticality flag (C) specifying the completion of the task. The last response message has the value true set in the flag.

3.3 Critical and Non-critical Service Identification

The proposed service composition model classifies the entire services as critical and non-critical based on the service level parameters. The critical services are those services whose providers are very less in number (N), higher failure rates and low trust binding values. The value of N is either system preferences or the user preferences. The requestor node sent the value for N as part of the user preference parameter Pr. If not specified, the system preferences is chosen.

The community manager forms the community using the Rm updated as part of the service initialize request message. It might not be necessary that all the nodes available in the Rm to be part of the community. The Service Evaluation model shown in Figure 2 is used to identify the nodes which are actively participating in the community by the community manager. The Service Evaluation model used in our work is the enhancement of the work proposed in [3]. We have enhanced the evaluation model to incorporate the identification of service criticality.

3.4 Service Evaluation Model and Community Formation

The parameter C is introduced for representing the criticality of the service and its value is decided based on the integration function of system and user preference.

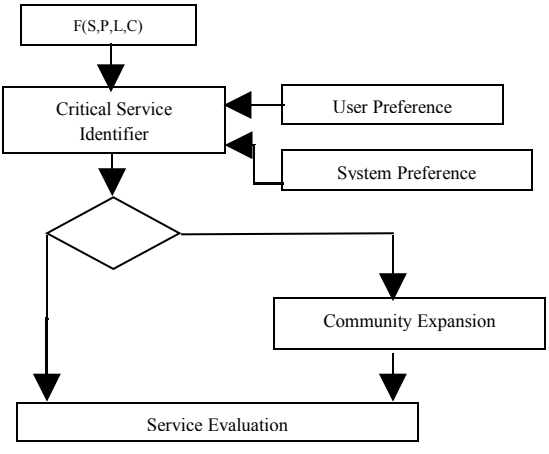


Fig. 2. Service Evaluation Model

If the C takes the value R, then Cs(No of critical service) is checked. Based on the value of Cs, the community manager decides to proceed for the dynamic community expansion or not. If the value of Cs is 1, then the community manager will go for the dynamic expansion of the community (Figure 3).

Once the service evaluation model is constructed, there may be more than one qualified node for a service. The objective of the service evaluation is to evaluate the service value for each node in the list L, and then sorts in the descending order. We are calculating the service value by taking into account the parameters such as quality

of service, time taken for completion and past trust rating. After the service evaluation nodes are ranked, the community manager identifies the top ranked nodes in the list for forming the community. Though only one service node is chosen for the non-critical service, two top ranked service nodes for critical services in the list is considered for forming the community. Hence the community manager constructs the community with the nodes providing both the critical and non-critical services.

Service Evaluation Function
F(S,P,L,C)

The community manager sends the message containing the information related to the community to the requestor node and the participating nodes in the community. The requestor node updates this message as part of the intermediate results(Ir) and sends back the responds to the community manager. The timeout for the next message will also be sent along with this message. The Intermediate result(Ir) format sent by the community manager for the community formation contains the number of nodes (N) to denote the set of nodes in the community, objective of the community (M) to represent the set of community goals or missions, define the community characteristics (Cp) such as community identity, number of nodes, community coordination manager, and community resources needed.

Intermediate Result Message
M(N,M,Cp)

3.5 Service Activation

In the service activation phase, the critical services are given priority and are executed using the parallel service composition model. The two top ranked service nodes in the service evaluation model ordered list (L) are executed in parallel. If either of the service faulted, the next ranked node providing the critical service is executed. If not available, the service discovery for critical service is initiated by the community manager. The parallel composition model we proposed in our earlier work [4] is used to execute the critical services in parallel to achieve better quality of service. We briefly have given below the service activation using our parallel composition model.

The composition task is implemented with two major components: a) the service manager, b) service activation engine. The service manager works for parallel composition by dividing the overall task into a set of independent set of service tasks that can be executed in parallel. The task is now split by the service manager into two independent threads. The threads are executed in the service activation engine.

After the execution of the critical services, the results are sent back to the requestor node with the next timeout period mentioned. The requestor Node acknowledges the message. The service nodes providing the critical services can be released by the community manager from the community. After all the critical services are executed the non-critical services are executed.

3.6 Fault Management

We have incorporated the fault handling mechanism for achieving the better quality of service. We have identified three fault origination points which includes Service Node failure, Community manager failure and the Requestor node failure. The service node failure happens when either the service node leaves the community, move away from the network or due to the any kind of network failure. If the service node was not able to activate the service due to any of the reasons mentioned above, the next ranked node in the list (L) providing the similar service is activated.

There might be the scenario where the community manager can itself get faulted due to network instability. The community manager updates the requestor node with the intermediate results and the timeout for the arrival of the next message. If the community manager itself faulted and not able to communicate to the requestor node, the requestor node will wait till the timeout period lastly updated by the community manager along with the intermediate result. After the timeout period, it recognizes that the community manager is not reachable and initiates the new service initialize request for the new community manager selection and the intermediate results are sent to the new community manager for further processing.

If the community manager does not get the acknowledgement from the requestor node after sending the intermediate results, then it considers the requestor node gets faulted. If the user preference parameter for erred acknowledgement is false, then it stops the service execution and stores the intermediate results to the PSNR with a timeout period above which the data are deleted. If the requestor node had a chance to acknowledge with the delay or trying to initialize the same service initiate request, then the community manager shares the data from the PSNR. If the user preference parameter for erred acknowledgement is true, then the community manager continues the service activation till the end and stores the service results in the PSNR.

4 Experimental Setup and Results

The middleware was implemented in J2ME with an Apache Server backend working as the community network server. The community network server in turn contacts the various service providers and gets the jobs executed. The faults are injected at the service nodes at runtime. If any service node faults, the community manager selects another service node from the ordered list and the service execution proceeds. This is done till the job is completed. At any point of time, for the critical service two alternative parallel services by service nodes are always executing.

We have conducted ten experiments with different number of mobile nodes forming different communities. For experiment purpose, currently QoS and Service Execution Time for each mobile node are taken for the evaluation. The two top ranked service nodes in the ordered list are executed in parallel. The failure nodes and their failure times are defined at runtime during the service execution. The Service reconfiguration is done when the node gets faulted, the next service node in the ordered list is chosen and the time taken for reconfiguration is noted. The Service Composition Time and the fault recovery time are calculated at the completion of task. For the experimental purpose, we have considered the independent service tasks.

In the service initialize request message the nodes that provide the required tasks are identified by the community manager. Based on the service response messages which contains the failure rates and trust binding values the requestor node selects the community manager. We used N0 as the community manager.

Table1. Experimental Datasets and Results

DATASET	NO OF MOBILE NODES	PARALLEL COMPOSITION			CRITICAL AWARE COMPOSITION		
		TOTAL TIME TAKEN	RECOVERY TIME	QOS	TOTAL TIME TAKEN	RECOVERY TIME	QOS
1	6	16	0	35	10	0	42
2	5	11	0	26	7	0	35
3	5	37	19	23	20	10	31
4	6	69	38	65	45	16	72
5	6	51	9	18	32	6	29
6	5	37	18	47	18	10	60
7	5	39	0	18	19	0	23
8	4	40	16	22	20	10	30
9	4	49	14	30	24	9	40
10	6	38	20	19	17	11	34

In the service evaluation phase, we identified the critical and non-critical services. We then execute the function to rank the service nodes. The ordered list is updated based on the function. The QoS values are updated as part of the ordered list. We injected the fault he node dynamically into the service node. In our model, the top two ranked service value threads are executed in parallel and based on the rankings, the clusters are formed with the given set of nodes. In the service activation phase, the top ranking nodes are executed in parallel to accomplish the task. If any error occurs during the activation phase, the faulted sub tasks need not be re-executed as the same is done in parallel with the another set of nodes which leads to the minimal composition length. The same pattern is followed for the entire cluster in the execution phase till the task is completed. In our proposed work, instead of the rankings based on the QoS parameter, we have used the function to construct the list of service nodes for parallel execution which improves the performance of the overall system.

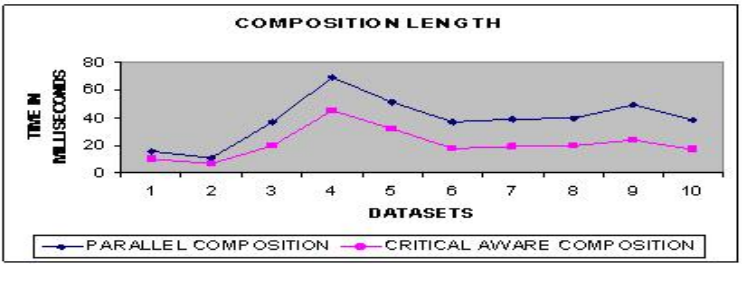


Fig. 3. Results of Critical Aware Service Composition Models – Composition time

The result graphs in the Figure 4 shows that the critical service composition model gives better results compared to the parallel composition model for the data sets.

The Figure 5 infers that for a set of independent subtasks taken, the failure recovery time is comparatively less in our model compared to the traditional model.

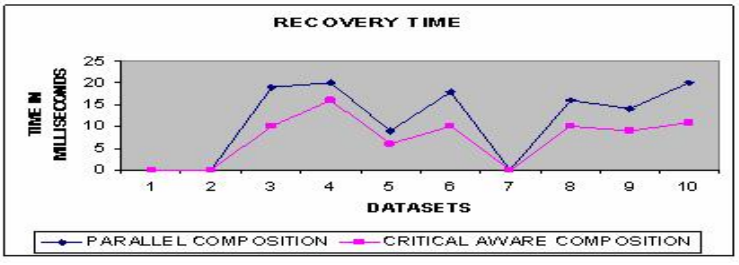


Fig. 4. Results of Critical Aware Service Composition Models – Recovery time

The Figure 6 shows that the critical aware service composition model shows better quality of service than the parallel model for the set of independent sub tasks services.

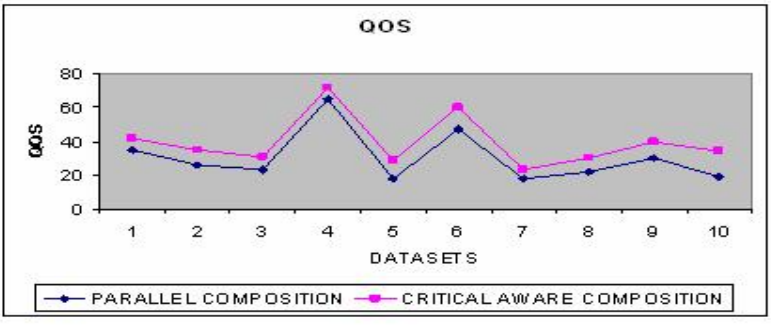


Fig. 5. Results of Critical Aware Service Composition Models – QoS

5 Conclusion and Future Work

We introduced a novel approach to study the service composition problem for pervasive composition environments. The services have been classified as critical and non-critical based on the service parameters. We modeled the systems operation for various eventualities and designed a fault tolerant critical aware parallel service composition model using an ad-hoc community network. The system was tested experimentally. The proposed critical aware composition model gives better performance in the fault situation as the recovery time is comparatively less. In future, the work will be tested in a large scale with over 100 nodes for more parameters and the operation of the overall system improved. Our work is unique and different from the existing systems in that the work combines the best of a parallel, critical task aware community based architecture for the pervasive computing system.

References

1. Brønsted, J., Hansen, K.M., Ingstrup, M.: Service Composition Issues in Pervasive Computing. IEEE Journal of Pervasive Computing 9(1), 62–70 (2010)
2. Ibrahim, N., Mouël, F.L.: A Survey on Service Composition Middleware in Pervasive Environments. IJCSI International Journal of Computer Science Issues 1 (2009)
3. Chakraborty, D., Joshi, A., Finin, T., Yesha, Y.: Service Composition for Mobile Environments. Journal on Mobile Networking and Applications, Special Issue on Mobile Services 10(4), 435–451 (2005)
4. Shriram, R., Sugumaran, V., Vivekanandan, K.: A middleware for information processing in mobile computing platforms. International Journal of Mobile Communications 6(5), 646–666 (2008)
5. Kalasapur, S., Kumar, M., Shirazi, B.: Dynamic Service Composition in Pervasive Computing. IEEE Transactions on Parallel and Distributed Systems 18(7), 907–918 (2007)
6. Kumaran, P., Shriram, R.: Service Composition Middleware for Pervasive Computing. In: 3rd International Conference on Network and Computer Science, vol. 6, pp. 26–28 (2011)
7. Chang, S.-C., Liao, C.-F., Liu, Y.-C., Fu, L.-C., Wang, C.-Y.: A spontaneous Preference Aware Service Composition Framework for Message-Oriented Pervasive Systems. In: 4th International Conference on Pervasive and Computing Applications (2009)
8. Chang, H.-C., Liao, C.-F., Fu, L.-C.: Unification of Multiple Preferences and Avoidance of Service Interference for Service Composition in Context-Aware Pervasive Systems. In: 7th ACM International Conference on Pervasive Services (2010)
9. Qian, Z., Wang, Z., Xu, T., Lu, S.: A dynamic service composition schema for pervasive computing. J. Intell. Manuf. (2010)
10. Kumar, M., Shirazi, B.A., Das, S.K., Sung, B.Y., Levine, D., Singhal, M.: PICO: A middleware framework for Pervasive Computing. IEEE Pervasive Computing 2(3), 72–79 (2003)