A Reactive Service Composition Architecture for Pervasive Computing Environments

Dipanjan Chakraborty, Filip Perich, Anupam Joshi, Timothy Finin, Yelena Yesha
Department of Computer Science and Electrical Engineering. University of Maryland, Baltimore County
dchakr1@cs.umbc.edu,fperic1@cs.umbc.edu,joshi@cs.umbc.edu,finin@cs.umbc.edu,yeyesha@cs.umbc.ed

Abstract Development of customized services by integrating and executing existing ones (refered to as service composition) has received a lot of attention in the last few years with respect to wired, infrastrutcure based web-services. With the advancement in the wireless technology and pervasive computing, we envison that in the near future, we will have such information or services embedded in various wireless devices in our vicinity. However, wired infrastructure-based service discovery and composition architectures do not take into consideration factors arising from the possible mobility of the service providers. In this paper, we present Anamika: a distributed, de-centralized and fault-tolerant design architecture for reactive service composition in pervasive environments.

1. Introduction

Service Composition can be defined as the process of creating customized services from existing services by a process of dynamic discovery, integration and execution of those services in a planned order to satisfy a request from a client. Research in the area of service discovery [1, 15, 3, 8, 18]and service composition [6, 17, 12, 16, 4, 14]has focused on trying to leverage the wide array of e-services available over the network to provide customized services to e-customers, for example planning a business trip for a person. There has been a sharp increase in these types of wired infrastructure-based services in the last few years. Existing service composition systems [14, 6, 12]broadly address the problems associated with composing various services that are available over the fixed network infrastructure. They primarily rely on a centralized composition engine to carry out the discovery, integration and composition of web-based e-services.

However, with computing today becoming increasingly pervasive, we envisage that in the near future, mobile and embedded devices will also be capable

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35618-1_37
C. G. Omidyar (ed.), *Mobile and Wireless Communications*

of providing customized information, services and computation platforms to peers in their vicinity. People will need the cooperation of services available in their resource-rich vicinity to satisfy their information needs.

Service Composition systems for the pervasive computing environments need a different design approach than those developed for wired services. This is because many of the assumptions of standard composition architectures of wired services are no longer valid in such dynamic environments. Service Composition architectures in wired infrastructure assumes the existence of a centralized composition entity that carries out the discovery, integration and execution of services distributed over the web. They also need a tighter coupling with the varying network layer protocols.

We have designed a distributed architecture to perform service composition in pervasive computing environments. Central to our system is the concept of a distributed broker that can execute at any node in the environment. An individual broker handles each composite service request, thus making the design of the system immune to central point of failure. A broker may be selected based on various parameters such as resource capability, geometric topology of the nodes and proximity of the node to the services that are required to compose a particular request. Current prototype of our system has been implemented over Bluetooth.

Service discovery and composition is an important and active area of research [5, 10]and has been studied widely in the context of web-services. Most of the research in realizing service composition systems for web-based services have a centralized architecture for service integration and execution management. We are aware of systems like eFlow [6], CMI [17], Ninja Service Composition Architecture [12], Sheng's framework [4]on declarative web service composition based on state charts that broadly address various problems related to service composition in the context of wired services. Due to lack of space, we are unable to present details of these systems.

2. System Architecture and Design Principles

In this section, we describe a general layered architecture that enables service composition in pervasive computing environments. Our architecture introduces two distributed **reactive** techniques to carry out service composition in purely ad-hoc environments. Our composition architecture primarily deals with the discovery, integration and execution of the components of a composite request. Figure 1 depicts the different layers and modules in the architecture. In the rest of the paper, we shall refer to a *client* as a device from where the service composition request originates. A *broker* is a device that coordinates the different components to calculate the result.

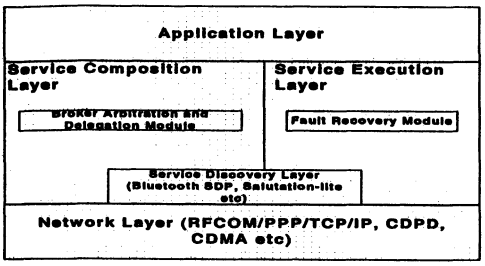

Figure 1. General Architecture for ad-hoc service composition

2.1 System Components

Network Layer:. The Network Layer forms the lowest layer in the architecture and encapsulates networking protocols that provide wireless/ad-hoc connectivity to peer devices in the vicinity. We assume the existence of a suitable network layer that provides us with connectivity to the neighboring devices.

Service Discovery Layer:. The service discovery layer is required for the proper functioning of the composition platform. There is a direct dependence of the success of the composition techniques on the underlying service discovery mechanisms. This layer encompasses the protocol used to discover the different services that are available in the vicinity of a mobile device. Our design of the service discovery mechanism is primarily based on the principles of *Peer-to-peer service discovery, Dynamic caching of neighboring service descriptions, Semantic description based service matching, service request routing and propagation control.* We do not employ central lookup-server based service discovery and maintenance. Each device has a Service Manager where the local services register their information. Service Managers advertise their services to neighboring nodes and these advertisements are cached. Services are described using a semantically rich language DAML+OIL [13]which is used in service matching also. On cache miss, the service request is forwarded to neighboring nodes. We describe our protocol in detail in [7].

Service Composition Layer:. This layer is responsible for carrying out the process of managing the discovery and integration of services to yield a

composite service. The process model of the composite service is supplied as input to this layer. In our current implementation, we have used the 'compositeprocess' definition of DAML-S to describe a process model. We describe our two reactive techniques of composition in detail in next two subsections.

Service Execution Layer:. The Service Execution Layer is responsible for carrying out the execution of the different services. Prior to this, the service composition layer provides a feasible order in which the services can be composed and also provides location and invocation information of the service(s). This layer has a module called the "Fault Recovery Module", which is responsible to guard against node failures and service unavailability. We explain the fault-tolerant techniques in the next two subsections. The Service Execution Layer and the Service Composition Layer are tightly coupled with each other due to their dependence on each other.

Application Layer:. The application layer embodies any software layer that utilizes our service composition platform. The application layer encompasses different GUI facilities to display the result of a composed service and provides the functionality to initiate a request for a composite service.

2.2 Dynamic Broker Selection Technique

This approach centers on a procedure of dynamically selecting a device to be a broker for a single request in the environment. In the following section, we describe three distinct features of the Dynamic Broker Selection Technique.

Broker Arbitration and Delegation:. When a request for service composition arrives at the service composition module in a mobile device it finalizes a platform that is going to carry out the composition and monitor the execution. Once the platform has been chosen, the device is informed of its responsibility. The mobile device acting as the *broker is responsible for the whole composition* process for a certain request. The selection of the broker platform may be dependent on several parameters: power of the platform (battery power left), number of services in the immediate vicinity, stability of the platform, etc. The brokerage arbitration might make the originator of the request to be the broker for that particular composition.

Each request thus may be assigned a separate broker. This makes the architecture immune to central point of failure and the judicious choice of brokerage platform has the potential of distributing the load appropriately within the different devices. This avoids the problem of swamping the central composition entity by numerous requests.

Service Integration and Execution:. The assigned broker's first job is to discover the services from its vicinity. The broker progressively increases its search "radius", a number of devices that it can reach by asking other devices in its radio range to forward service request, to discover all of the different services necessary for the composition. The broker returns failure when it fails to discover all of the required services. Service discovery and integration is followed by service execution. The information obtained during the service discovery (service address, port, invocation protocol) is utilized to execute the services.

Fault Recovery:. Faults in ad-hoc environment may occur due to a service failure, due to a sudden unavailability of the selected broker platform, or due to network partition. The standard solution to this problem is to make the requester to initiate a new request for every composite service. This is very inefficient and not applicable in our environment due the relatively high occurrence probability of the above failures. The fault-tolerance module in the architecture employs check pointing to guard against such faults. The broker for a particular request sends back checkpoints and the state of the request to the client of the request after a subtask is complete. The client keeps a cache of this partial result obtained so far. If the broker platform fails, the source node detects the unavailability of updates. The source of the request then reconstructs the query that is still left unsolved by the broker. This request is now treated as a different service composition request in the environment. Thus, If a node currently acting as the broker of a request fails, then using the same principle the architecture adapts itself to select other brokers dynamically.

2.3 Distributed Brokering Technique

The key idea in this approach is to distribute the brokering of a particular request to different entities in the system by determining their 'suitability' to execute a part of the composite request.

Broker Arbitration:. This module performs almost the same set of actions as described in the previous section. However, the key difference is that it only tries to determine the broker for the first few services (say S1 to Si) in the whole composition. This layer tries to utilize the resources available in the immediate vicinity instead of looking for the resources required to execute the whole composition. Thus, a single broker only executes a part of the whole composite process (based on the resources that it currently has available to it).

Service Integration and Execution:. The broker is responsible for composing the services S1 to Sn. The broker decides on a service search "radius". The composition is carried out among services discovered within this radius.

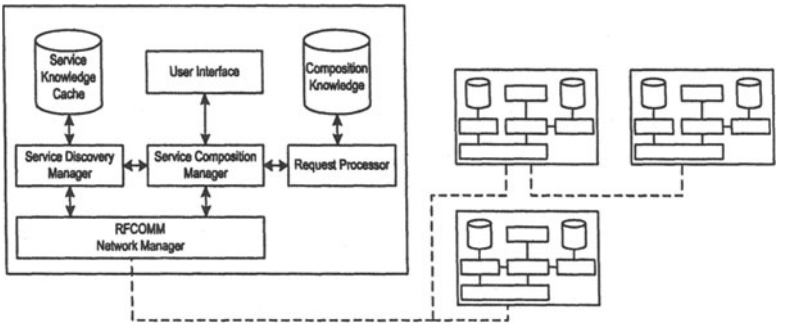

Figure 2. System Components in Anamika Reactive Service Composition Architecture

Suppose a broker determines that it has services S1 to Si available in its vicinity(within radius r). It goes ahead and carries out the partial integration and execution. It then informs the requester (source node) about the 'current state' of the execution. Secondly, it uses the 'Broker Arbitration' Module to select another broker which has the ability to carry out a subset or whole of the remaining composition. In this manner, the composition hops from one node to another till the final result is obtained. Then the current broker returns the final answer of the composition to the client.

3. Implementation and Experiments

In our initial implementation of the design architecture, we have developed a reactive service composition system called Anamika. The individual components of the Anamika system existing in participating mobile devices are described in Figure 2. Current prototype of the architecture has been implemented over Bluetooth [18]. Composition knowledge is described using DAML-S[11]in terms of subset of individual services that might be able to satisfy a composite request. Service discovery is done in a peer-to-peer manner using semantic description of services using DAML-S and our light-weight reasoning engine present on participating devices. Anamika implements *Dynamic Broker Selection* mechanism and decides the best platform to carry out the composition based on a combination of the processor power of the platform and number of services that the platform has. The Network Manager implements the API required for higher layers to reliably communicate to neighboring peers over Bluetooth. We implemented the networking level API over RFCOMM [18]protocol over IBM's BlueDrekar transport driver [2]for Linux kernel 2.4.2-2. Service Discovery Manager provides the functionality to local Composition Manager to discover services in Bluetooth peers. The principles of our Service Discovery Manager is described in detail in [7]. The Service Composition Manager is the principal component that is responsible for service composition

and management. Peer Service Composition Managers collaborate with each other to implement different techniques of service composition. The Composition Knowledge base has been modeled in DAML-S. Due to lack of space, we are unable to provide a greater detail of the implementation. However, it is available in our technical report [9].

Experiments:. We carried out various experiments to validate the proper functioning of the Anamika Reactive Service Composition system and test different features of the *Dynamic Broker Selection Technique.* In our experimental setup, services reside on different laptops. These services are registered to the local composition managers residing on the machines. Bluetooth modules provided by Ericsson and IBM's BlueDrekar software stack [2]and transport driver are used by the Network Manager to communicate between devices. Some of the laptops also have 802.11b connectivity to the Internet. Every Anamika system displays a graphic user interface for clients to access composite services formed by the services available in the vicinity. There is no central "Broker service" in the system and all the participating systems are liable to be brokers. We carried out experiments to test the proper functioning of the different mechanisms for service composition and under different states of the ad-hoc environment. However, due to space limitations, we are unable to provide details of the experiments.

4. Conclusions

In this paper, we have introduced a novel design approach for service composition in pervasive computing environments. Our architecture for service discovery and composition is distributed, decentralized and fault-tolerant to service and network unavailability. Service Discovery is done in a peer-to-peer mode rather than a centralized mode, and service descriptions are cached for scalability. We introduce two *reactive* techniques, *Dynamic Brokerage Selection* mechanism and *Distributed Brokerage technique* to accomplish service composition in dynamic environments. Our approach enables any device participating in the composition to act as the broker, making the design immune to single point of failure. We use a source-monitored fault-tolerance mechanism using checkpoints and rollbacks to the last completed service. We have also presented the Anamika system, an initial implementation of our design architecture. Anamika has been implemented over Bluetooth using RFCOMM as the network layer.Our future work includes implementing the "Distributed Brokerage" mechanism in Anamika, perform assessment of the different mechanisms with respect to factors like mobility of the environment, availability rate of the services.

References

[1] The Salutation Consortium Inc 1999. Salutation architecture specification (part 1), version 2.1 edition. World Wide Web, `http://www.salutation.org`.

[2] IBM alphaworks. BlueDrekar protocol driver. World Wide Web, `http://www.alphaworks.ibm.com/tech/bluedrekar`.

[3] Ken Arnold, Ann Wollrath, Bryan O'Sullivan, Robert Scheifler, and Jim Waldo. *The Jini specification*. Addison-Wesley, Reading, MA, USA, 1999.

[4] B. Benatallah, M. Dumas, Q. Sheng, and A. Ngu. Declarative composition and peer-to-peer provisioning of dynamic web services. In *18th International Conference on Data Engineering.*, February 2002.

[5] F. Casati, D. Georgakopoulos, and M. Shan editors. Special issue on e-services. *VLDB Journal*, 2001.

[6] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M. Shan. Adaptive and dynamic service composition in eflow. Technical Report, HPL-200039, Software Technology Laboratory, Palo Alto, CA, march 2000.

[7] Dipanjan Chakraborty, Anupam Joshi, Tim Finin, and Yelena Yesha. GSD: A novel group-based service discovery protocol for MANETS. In *4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN 2002). Stockholm. Sweden*, September 2002.

[8] Dipanjan Chakraborty, Filip Perich, Sasikanth Avancha, and Anupam Joshi. Dreggie: A smart service discovery technique for e-commerce applications. In *20th Symposium on Reliable Distributed Systems*, october 2001.

[9] Dipanjan Chakraborty, Filip Perich, Anupam Joshi, Timothy Finin, and Yelena Yesha. A service composition architecture for pervasive computing environments. Technical report, University of Maryland Baltimore County. USA, March 2002. TR-CS-02-02.

[10] G. Weikum. Editor. Special issue on infrastructure for advanced e-services. *IEEE Data Engineering Bulletin, 24(1)*, March 2001.

[11] DARPA Agent Markup Language for Services. World Wide Web, `http://www.ai.sri.com/daml/services/daml-s.pdf`.

[12] R.H. Katz, Eric. A. Brewer, and Z.M. Mao. Fault-tolerant, scalable, wide-area internet service composition. Technical Report. UCB/CSD-1-1129. CS Division. EECS Department. UC. Berkeley, January 2001.

[13] DARPA Agent Markup Language and Ontology Inference Layer. `http://www.daml.org/2001/03/daml+oil.daml`.

[14] David Mennie and Bernard Pagurek. An architecture to support dynamic composition of service components. Systems and Computer Engineering. Carleton University, Canada.

[15] UPnP White Paper. World Wide Web, `http://upnp.org/resources.htm`.

[16] Chaitanya Pullela, Liang Xu, Dipanjan Chakraborty, and Anupam Joshi. Component based architecture for mobile information access. In *Workshop in conjunction with International Conference on Parallel Processing (ICPP).*, August 2000.

[17] H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker. Modeling and composing service-based and reference process-based multi-enterprise processes. In *Proc. Intl. Conference on Advanced Information Systems Engineering, Sweden.*, June 2000.

[18] Bluetooth Specification. World Wide Web, `http://www.bluetooth.com/developer/specification/Bluetooth_11_Specifica%tioBook.pdf`.