A Distributed Service Composition Protocol for Pervasive Environments

Dipanjan Chakraborty
Ph.D Student
UMBC
dchakr1@cs.umbc.edu

Yelena Yesha
Professor
UMBC
yeyesha@cs.umbc.edu

Anupam Joshi
Associate Professor
UMBC
joshi@cs.umbc.edu

***Abstract*— Service composition in pervasive environments enables users to utilize services in the environment to solve complex queries. Current work in development of service composition architectures focuses on wired-networked environments where solutions are centralized and tailored towards a reliable network and fixed service topology. In this paper, we present an alternate and novel design architecture of a broker-based distributed service composition protocol for pervasive environments. We present simulation results by comparing our protocol to a centralized architecture for composition. Results show that our distributed broker-based composition architecture perform better than the centralized solution in terms of composition efficiency, broker arbitration efficiency and composition radius.**

Key Words: Mobile Service, Service Composition, Discovery, architecture, Ad-hoc Networks, Broker

I. Introduction

Service composition refers to the technique of creating complex services with the help of smaller, simpler and easily executable services or components. By "Service", we refer to any software component, data, or hardware resource on a device that is accessible by others. For example, we can consider a Bluetooth-enabled PDA carrying music files to be a service providing songs in various formats.

The spurt of e-services on the web has increased the importance of service composition. Service Composition enables us to integrate existing services to satisfy complex requests (requests that require cooperation of multiple services) by clients. There has been research [19], [12], [3] in trying to leverage the wide array of e-services available over the wired network to be able to provide more customized, complex services to e-customers. However, in the future, along with services in the fixed network infrastructure, we would also be able to access services present in the various mobile devices in our vicinity. Service composition protocols also need to take advantage of the services/resources available in the pervasive environment.

Research in Service composition has predominantly followed two directions. One direction of research tries to define rich languages [10], [14], [11] to describe services and workflows appropriately. The other direction of research aims in developing architectures [20], [12], [13], [18], [4] that enable service composition. These architectures assume a workflow specification of a composite service and perform the task of discovering, integrating and executing the services. In this paper, we focus on service composition architectures since we believe that it is critically important in a pervasive environment and requires a different approach from wired service composition architectures.

Current service composition platforms [3], [12], [15] have been designed with the inherent assumption that the e-services are resident in the fixed network infrastructure. Thus, they are running on a relatively stable platform and can access each other using relatively high bandwidth communication channels. The composition platforms are based on a centralized manager or Broker that manages the various issues like service path creation [7], delegation of service discovery to the proper discovery manager, appropriate combination of different services and management of the information flow between service components. In pervasive computing environments, firstly, the centralized design approach of wired-network based composition engines is prone to single point of failure, especially since all nodes are mobile and unreliable. Secondly, service topology (distribution of services on various ad hoc nodes) changes due to mobility. Service composition architectures should be able to utilize the spatial distribution of services to optimize service integration and execution. Fault management strategy has to take into consideration network level disconnection, service discovery failures, and service execution failures. These issues call for an alternate design approach of service composition systems for pervasive environments.

We have developed a distributed broker-based protocol for service composition in pervasive/ad hoc environments. In a nutshell, our protocols works like this: For each composite request, our protocol elects a Broker from within a set of nodes. The source node delegates the responsibility of composition (i.e. discovery, integration and execution) to the elected Broker. The elected Broker utilizes a distributed service discovery architecture [5] to discover services. It then integrates and executes services needed for a composite request. Our protocol is decentralized and immune to central point of failure. The Broker election mechanism has the capability of balancing the total load on the system appropriately within the set of cooperating nodes. We use a checkpoint-based source-monitored fault tolerance mechanism to detect execution-level faults. We present simulation experiments to demonstrate the effectiveness of our protocol.

II. Background

Service discovery and service composition is an important and active area of research [2], [8] and has been studied widely in the context of wired-networked services. Most of the research in realizing service composition systems for wired-networked services has a centralized architecture for service integration and execution management. We are aware of systems like eFlow [3], CMI [19], Ninja Service Composition Architecture [12], Sheng's framework [1] 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.

There has been very limited work in the development of distributed protocols for service composition in ad hoc/pervasive environments. In prior work [17], we developed a middleware to handle composite requests from mobile devices using wired networked resources. Our middleware platform took into account mobility related issues like disconnections, bandwidth and resource constraints on mobile devices while trying to process a composite query from the device. The protocol can be considered as a "one-step" progress to enable mobile devices to compose services. However, it depends on the wired infrastructure for services. Moreover, middleware-oriented protocols are essentially semi-centralized and hence unsuitable for ad hoc environments. Basu et. al [16] has described a hierarchical task-graph based approach to do service composition in ad hoc environments. In their work, a composite service is represented as a task-graph with leaf nodes representing atomic services. Different sub trees of the graph are computed in a distributed manner in a MANET (Mobile Ad hoc Network). Service composition is coordinated by the source of the request. Even though the domain of the problem is the same as our's, the approaches are very different. In their approach, the coordinator uses global search across the whole network to do composition. This has a high network load on the system. We believe that the coordinator should be able to utilize the service topology and do the composition. To be precise, the coordinator node should be able to consider the spatial distribution of the services to optimize the cost of composition. Thus, our protocol does not have any restriction on the location of the coordinator and we use a distributed algorithm to decide the best coordinator for a request. Our protocol allows distributed execution of a composite service once the coordinator has been decided. Moreover, their approach uses a broadcast-based service discovery mechanism while we use a more efficient distributed service discovery mechanism [5] imposing lesser network load.

There is a plethora of work in trying to define languages to represent composite services in a workflow-oriented or semantic manner. This class of work in service composition, nonetheless important is not an issue of this paper. For the purposes of this paper, the composite service can be represented in any of these languages. Examples of such work include DAML-S[10], WSFL[14], XLANG[9], BPEL[11] etc. In particular, we have used DAML-S to represent composite services.

III. Protocol

Our composition protocol uses an efficient distributed service discovery infrastructure to discover services that is described in detail in a separate publication [5]. The discovery protocol uses intelligent selective forwarding technique and localized broadcasting combined with peer-to-peer caching to discover services in a MANET. We believe that a truly distributed service composition protocol should utilize the topology of the ad hoc network and the spatial distribution of the services present on them. To achieve this, our protocol executes a *Broker Arbitration* phase that decides the most capable node to act as the coordinator or *Broker* for a certain request. Thus, each request may be assigned to a different *Broker*. The protocol rapidly adapts to topology changes and network as well as service failures. We define some key terms in the protocol:

1) **Request Source (RS):** Mobile device from where a particular composite request originates. Note that a node is referred to as the *Request Source* only with respect to the request that it has originated.
2) **Service Provider (SP):** Mobile device that contains services that are accessible from other peer nodes.
3) **Broker:** The device that handles manages the discovery, integration and execution of a composite request. This node can itself be the *Request Source* or a *Service Provider* for another composite request.
4) **Description-level Service Flow (DSF):** Declarative description of a composite service or a composite request. We use DAML-S to specify a composite service. It simply consists of a list of service descriptions along with the desired flow of execution that constitutes the composite service.
5) **Execution-level Service Flow (ESF):** A complete specification of the composite service with execution-level details required to invoke the services in the SPs.
6) **Atomic Service:** A service that resides on a single SP and can be invoked from other devices. This service may consist of further components, but the components must reside on the same SP to make the service *atomic*.

Our composition protocol essentially consists of four phases. (1) The RS initiates a *Broker Arbitration Phase* that analyzes the composite request and elects a node to act as the *Broker* for that request. The composite request is then delegated to the Broker from the RS. (2) This is followed by the *Service Discovery Phase* where the Broker uses the underlying discovery infrastructure to discover required SPs for the composition. (3) This is followed by the *Service Integration Phase* where the Broker computes service-to-node bindings for the composite request. An Execution-level Service Flow (ESF) is computed from the Description-level Service Flow (DSF) of the composite request. (4) The *Service Execution Phase* phase executes the composite service following the ESF. This phase handles execution-level faults by employing checkpoint-based

monitoring of the execution. In the following subsections, we describe the different phases in greater detail.

A. Broker Arbitration Phase

This phase is initiated by the RS. The composite request is parsed to generate a list of the required services. The RS queries nodes in its vicinity (with controlled broadcast) to determine their suitability to act as Broker for the composite request. It supplies them with the enumerated list of services. Each node computes a *potential value* for itself based on its own resources as well as the Service Providers (SP) located in the local vicinity of itself. The services located in the local vicinity are obtained from the discovery infrastructure (explained in the section III-B). This potential value is returned to the RS. The RS selects the Broker with the maximum potential value after it has obtained sufficient replies or after a certain timeout. The local resources considered by the potential function include number of matching local services, battery life and current processing load on the node. The potential function uses remote service advertisements cached by a node to approximate the number of services in its neighborhood. In future, we aim to incorporate node density as well as service density into the potential function. Service density may be different from node density since there may be more than 1 (or even 0) services on a node.

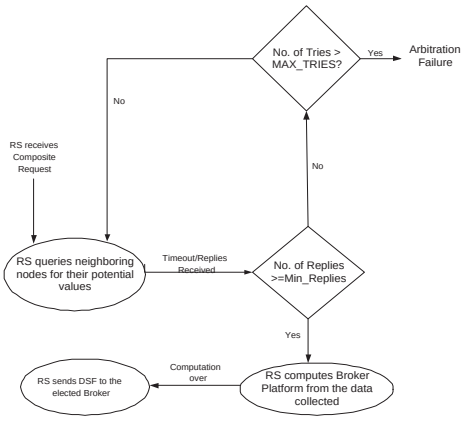

Fig. 1. Flow Diagram of Broker Arbitration Phase

The Description-level Service Flow (DSF) of the composite request is sent to the elected Broker using the underlying routing protocol. Figure 1 shows the flow diagram of this phase. We have used a service-based routing protocol developed by us [6] to do routing. In short, our routing protocol is on-demand and uses the path formed during service discovery to send data. In [6], we have shown that integrating routing with discovery in MANETs increase system efficiency.

In this phase, each composite request in the system thus may be assigned a separate Broker. This makes the architecture immune to central point of failure and the judicious choice of Brokers has the potential of distributing the load appropriately within different devices. Formally, we denote U(B_i) as the utility value of each potential Broker for a composite request S.

$$U(B_i) = f(S_c(B_i), S_m(B_i), L(B_i), J(B_i))$$

where $S_c(B_i)$= Number of service advertisements cached by B_i; $S_m(B_i)$=number of services belonging to the composite request that are present in the cache of B_i; $L(B_i)$=battery life of B_i; $J(B_i)$=Current number of requests being processed by B_i. A Broker B_i is selected based on the following equation.

$$\exists B_i \text{ such that } \forall B_j \ (U(B_i) \geq U(B_j))$$

B. Service Discovery Phase

Our composition protocol utilizes our previously developed service discovery infrastructure [5] (referred to as GSD) to discover services. It also uses the information provided by the infrastructure during the Broker Arbitration Phase. GSD is based on the concepts of peer-to-peer caching of advertisements of services and group-based selective forwarding of requests. The selective forwarding uses DAML-based semantic information present in the service requests and service descriptions. Each node advertises its own services via localized broadcast to limited vicinity. These service advertisements are cached by peer nodes. Service discovery requests are matched with the cached descriptions of services for possible match. Service discovery requests are also selectively forwarded (in case of a mismatch) to a set of neighbors based on certain service-group information that is propagated along with the advertisements.

C. Service Integration Phase

The DSF specification of the composite service is converted into an ESF in this phase. This phase deals with combining the discovered services meaningfully and filtering out unnecessary components or services based on execution-level cost estimate of the composed service. More specifically, corresponding to each service description in the composite request, an actual service is discovered. The network load created due to the discovery process is controlled by regulating the number of hops within which the service discovery is performed. There could be multiple instances of the same service existing in the environment. Our protocol currently selects the nearest available service. However, it can easily be modified to incorporate other cost factors. A new Execution-level Service Flow is constructed that contains information on the actual service, its node binding etc. It also contains control flow related information and actual network parameters (number of hops, bandwidth etc) that would affect the flow. This phase ends when all the required services have been instantiated in the ESF. Figure 2 describes the pseudo code of the operation of the protocol during the discovery and integration phase.

D. Service Execution Phase

The Broker goes into this phase after it has discovered all the services and a complete ESF has been constructed. In this phase, the Broker coordinates the execution of the services in the order specified by the ESF. The execution of the individual services occurs in a distributed manner at the

```
For each service Si in DSF {
   broadCast_Diameter=MIN_DIAMETER;
   service_discovered=FALSE;
   no_retries=0;
   while(!service_discovered &&
         no_retries<=MAX_RETRIES) {
    Call GSD to discover Si;
    if Si has been discovered{
      ESF+=Invocation Details of S_i;
      service_discovered=TRUE;
      }
    else {
      broadCast_Diameter+=BROADCAST_INCREMENT;
      no_retries++;
      }
   }
}
```

Fig. 2. Pseudo code of Service Discovery and Integration Phase

nodes hosting those services. The Broker uses the underlying routing protocol [6] to transmit information received from the previously executed service in the ESF to the next service. Invocation information for services (e.g. node address, actual service name, arguments etc) is obtained from the ESF. We observe that the execution of each composite request follows a "star" pattern for data and control flow. In other words, the data flows from one node to another node through the Broker. Thus we call it "star" execution pattern. In our next phase, we aim to incorporate "mesh-based" formation of the execution pattern to the ESF where the data would directly flow from one service provider to the next service provider. We use a checkpoint-based source monitored fault tolerance scheme to detect node failures and hence failures of services residing in those nodes (if they are part of a composite request). We describe the process in the next section.

1) Checkpoint-based Fault Tolerance: The basic solution to address faults in composition is for the source to restart the whole process if any service has failed during the execution. This solution is unable to utilize the partial results obtained so far. The energy and bandwidth spent in computing part of the query is also lost. In ad hoc environments, services are not stable. So it is critically important to be able to utilize the results of the partial execution.

We use checkpoints at the RS with a service-level granularity of one to commit partial executions in the composition. The RS maintains an execution state for each request it generates. The Broker coordinating a composition updates the RS with the "execution-state" as well as the partial result obtained after a service has been executed. The source times out in absence of an update from the current Broker. It then re-computes the ESF by pruning the part that has already been executed and committed. This request is treated as a new composition request in the environment and adequate actions are taken. This solution however, imposes additional overhead of transmitting checkpoints in a MANET. We are currently analyzing the effect of our checkpointing algorithm on the network bandwidth. Thus, in case of failures of SPs or Brokers, our protocol adapts to the changes dynamically and tries to utilize the available resources in the new environment. We assume that the RS would not fail before successful completion of the composition. This is a reasonable assumption, since the RS would ideally want to keep itself turned on till it receives a reply.

IV. Experiments

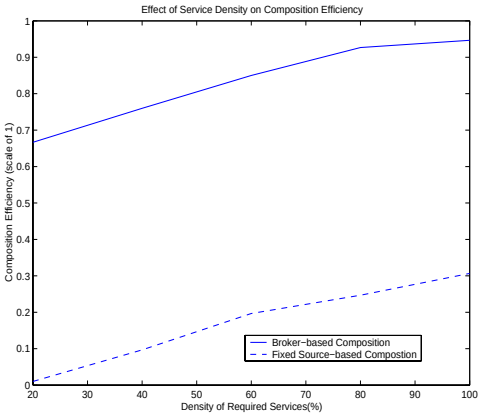

Fig. 3. Comparison of Composition Efficiency with respect to Service Density

We implemented our protocol on the well-known ad-hoc network simulator Glomosim [21] under various service densities, mobility and topologies. We compared our protocol to the often-used Fixed-Source based Composition protocol where all the composite requests generated in the system are sent to one fixed node that acts as the composition engine. We considered composite requests of various orders (in terms of number of distinct atomic services required). We define service density as the percentage of nodes containing one or more of the services that are required in a composite request. In this paper, we present results for composite requests having an order of three with density ranging from 20 to 100%. Simulation was carried over a set of 64 nodes, following random way-point mobility with speed of 2m/s and stoppage time of 5s. We considered a broadcast control hop count of 1 for the Broker Arbitration Process and underlying GSD protocol.

Figures 3 and 4 show the effect of service density on the efficiency of composition and broker arbitration. Composition efficiency refers to the fraction of composite requests that were successfully instantiated and executed. Broker Arbitration efficiency refers to the fraction of composite requests for which a Broker was assigned and the task was received by the assigned Broker. We observe that our Broker-based composition protocol outperforms the Fixed-Source based solution by approximately 60%. We also see that the efficiency increases with increasing service density. This is expected, since with increasing service density, we increase the chances of the services being discovered in the vicinity and executed. Our protocol performs better since it utilizes the services in the local vicinity of the RS and also assigns Brokers based on

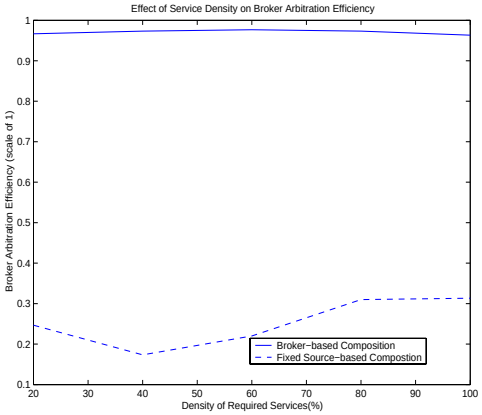

Fig. 4. Comparison of Broker Arbitration Efficiency with respect to Service Density

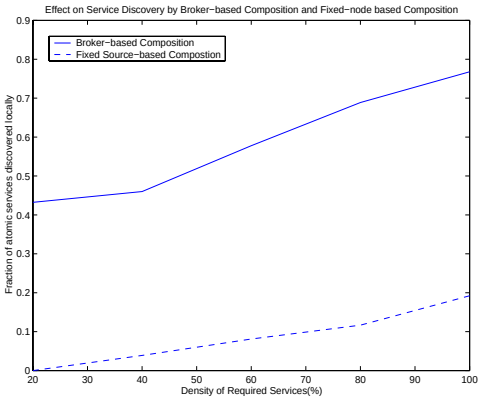

Fig. 6. Comparison of the Protocols in terms of Locally Discovered Services

the current network topology. Broker Arbitration Efficiency is also high in our protocol since the assigned Brokers are located near the RS. Hence, Broker acknowledgements (from the source) have to travel lesser hops. Thus, there are lesser acknowledgement message losses leading to greater arbitration efficiency.

Figure 5 compares our protocol with the centralized protocol with respect to composition radius. Composition radius is defined as the average number of hops needed by a Broker to discover all the required services for a particular composite request. The lesser the composition radius, the lesser is the network overhead in communication. Our protocol shows that it performs better in locating nearby nodes that contain the required services. This is more due to the topology-sensitive placement of the Broker in the Broker Arbitration Phase. This is also corroborated in figure 6 where we compare the percentage of discovered services that were found locally in the assigned Broker node. This result shows that our Broker placement strategy achieves better utilization of the service topology.

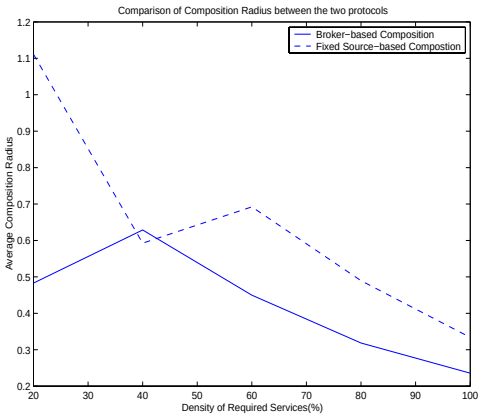

Fig. 5. Comparison of the Protocols in terms of Composition Radius

V. Conclusions

In this paper, we have presented a Broker-based distributed Service Composition protocol for MANETS. Our protocol is decentralized and efficiently utilizes the spatial locality of the services. Each composite request is independently assigned a Broker. The Broker Arbitration mechanism uses a controlled broadcast-based scheme to collect information from nearby nodes. Broker Selection is based on a utility value for each node. The utility value takes into account services present in the node, computation and energy resources and most importantly, service topology of the surrounding vicinity. Service composition is carried out in a distributed manner utilizing the resources/services surrounding the assigned Broker. We have compared our protocol to a Fixed-node based Composition protocol where all the requests are sent to a preconfigured node in the system. Simulation results show that our protocol performs better in terms of composition efficiency and broker arbitration efficiency. We also show that our protocol achieves better results in terms of composition radius and utilizing spatial locality of the required services participating in a composition.

References

[1] 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.

[2] F. Casati, D. Georgakopoulos, and M. Shan editors. Special Issue on E-Services. *VLDB Journal*, 2001.

[3] 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.

[4] D. Chakraborty and A. Joshi. Dynamic Service Composition: State-of-the-Art and Research Directions. Technical report, University of Maryland Baltimore County, December 2001. TR-CS-01-19.

[5] Anupam Joshi Dipanjan Chakraborty. GSD: A novel group-based service discovery protocol for MANETS. In *IEEE Conference on Mobile and Wireless Communications Networks, Stockholm, Sweden.*, September 2002.

[6] Anupam Joshi Dipanjan Chakraborty. An integrated service discovery and routing protocol for ad hoc networks. Technical Report, TR-CS-03-23, University of Maryland, Baltimore County, March 2003.

[7] The Ninja Project. UC Berkeley Computer Science Division. `http://ninja.cs.berkeley.edu`.

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

[9] XLANG. Web Services for Business Process Design. World Wide Web. `http://xml.coverpages.org/xlang.html`, 2001.

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

[11] BPEL4WS. Business Process Execution Language for Web Services. World Wide Web. `http://xml.coverpages.org/bpel4ws.html`, 2002.

[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] Joost N. Kok and Kaise Sere. Distributed service composition. In *Technical Report No. 256. Turku Centre for Computer Science. Finland.*, march 1999.

[14] Web Services Flow Language. World Wide Web, `http://xml.coverpages.org/wsfl.html`.

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

[16] Thomas. D.C. Little Prithwish Basu, Wang Ke. A novel approach for execution of distributed tasks on mobile ad hoc networks. In *IEEE WCNC. Orlando. Florida*, 2002.

[17] 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.

[18] Pierre-Antoine Queloz and Alex Villazon. Composition of services with mobile code. In *Proc. First International Symposium on Agent Systems and Applications Third International Symposium on Mobile Agents. Palm Springs. California.*, 1999.

[19] 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.

[20] C. Thompson, P. Pazandak, V. Vasudevan, F. Manola, G. Hansen, and T. Bannon. Intermediary architecture: Interposing middleware object services between web client and server. In *Workshop on Compositional Software Architectures. Monterey. California*, 1998.

[21] Mario Gerla Xiang Zeng, Rajive Bagrodia. Glomosim: A library for parallel simulation of large-scale wireless networks. *Proc. 12th Workshop on Parallel and Distributed Simulations*, 1998.