Service Composition for Mobile Environments *

Dipanjan Chakraborty Anupam Joshi Tim Finin Yelena Yesha

University of Maryland, Baltimore County.

1000, Hilltop Circle.

Baltimore. MD: 21250.

{*dchakr1,joshi,finin,yesha*}*@cs.umbc.edu*

Abstract

Service Composition, that is, the development of customized services by discovering, integrating and executing existing services has received a lot of attention in the last couple of years with respect to wired-infrastructure or Internet web services. With the advancement in the wireless technology and rapid deployment of mobile devices, we envision that in the near future wirelessly connected mobile devices in a given vicinity will also provide services that can be leveraged in the composition process. This is particularly true of what have been described as "pervasive computing" environments. However, wired-infrastructure based service composition architectures are not designed to consider the various factors like mobility, device heterogeneity, resource variability and reliability in a mobile environment. In this paper, we describe the issues related to service composition in mobile environments and evaluate criteria for judging protocols that enable such composition. We present a distributed architecture and associated protocols for service composition in mobile environments that take into consideration mobility, dynamic changing service topology and device resources. The composition protocols are based on distributed brokerage mechanisms and utilize a distributed service discovery process over ad-hoc network connectivity. We present simulation results of our protocols, and compare them with a centralized service composition protocol traditionally used for wired-infrastructure environments. The results show that our approach clearly outperforms the existing centralized approaches, and that our protocols are able to adapt and better utilize the changing

*This work is supported in part by NSF awards 9875433 and 0070802, DARPA DAML program and IBM

service topology and resources in a mobile environment.

1 Introduction

Service composition refers to the technique of creating *composite 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 the GPS on a mobile device to be a service that provides location coordinates of that device.

Technological advances in mobile device design and development as well as wireless networking are leading us towards a vision where devices all around a person, either embedded as a part of smart spaces, or being carried by other people in the vicinity, will provide an array of services [40, 24, 23] that she might want to use. Service discovery infrastructures [35, 2, 17, 16, 34, 39] are being developed to discover services in mobile environments. User queries often however look for services that are not *pre-existent* on any device but can be obtained by appropriately combining already existing services in the environment. Examples of such queries range from mobile commerce environments to army warfront environments, sensor networks to deep space exploration research. In mobile commerce environments, often times users look for the best deals in various shops in a mall that can be obtained by appropriately combining information given out by various *shopping services* in the physical vicinity. Warfront activities and sensor networks often need to integrate data (that are offered by services on various devices/sensors) from heterogeneous sources to discover meaningful trends. Other examples of service composition in mobile environments are dynamic traffic information aggregation from various services and sensor data collection from multiple sensors.

There has been research [37, 18, 4] in development of composition architectures for wired-infrastructure environments like Internet. In the future, along with services in the wired-infrastructure, we will also be able to access services present in the various mobile devices in our vicinity. Service composition architectures for mobile environments will be expected to utilize these services in addition to the services available in the wired-infrastructure.

Research in service composition has predominantly followed two directions. One direction defines languages to formally specify services and composite services [14, 22, 15] in terms of service input/output, service pre-condition and post-conditions, fault handling, joint exception handling and service invocation mechanisms. This research also includes developing complex planning engines [12, 25] that utilize these service descriptions to generate declarative

specification of workflows that compose different services. The other direction of research aims to develop architectures [38, 18, 19, 33, 5] that enable service composition. These architectures assume a declarative specification of a composite service[1] and performs the task of discovering, integrating and executing the actual services. In this paper, we focus on service composition architectures since we believe that it is critically important in a mobile environment and requires a different approach from currently available wired-infrastructure based composition architectures. We leverage our prior work [6, 7, 8] and the emerging DARPA Agent Markup Language for Services (DAML-S) [14] standard to provide the semantically rich declarative specification of a composite service.

Current service composition architectures [4, 18, 26] have been designed with the inherent assumption that the services are resident in the wired infrastructure. Thus, services are assumed to be on stable nodes/devices and connected to each other over high bandwidth reliable communication channels. Composition architectures are centralized and consist of a preconfigured *composition manager* or *broker* residing on high-end machines. The *composition manager* performs *service coordination and management* that involves managing issues like service path creation [11], delegation of service discovery to the proper discovery manager, appropriate combination of different services and management of the information flow between service components. Such architectures do not work well in mobile environments, especially ad-hoc mobile environments. Some of the notable limitations are: (1) Central Point of Failure: Centralized design approach of wired-infrastructure based composition architectures is prone to single point of failure in a mobile environment. (2) Mobility: Mobility changes the topology of a mobile environment. This changes the *service topology* (distribution of services on various mobile nodes). Current composition architectures lack support for mobility. (3) Fault Management: Mobile nodes are prone to various kinds of faults. Faults range from network level disconnection, service discovery failures, and service execution failures to node failures. Composition architectures for mobile environments need to be adaptive and resilient to these failures.

We categorize mobile environments into infrastructure-based and infrastructure-less or ad-hoc environments. Infrastructure-based mobile environments have access to services/resources in the wired infrastructure through gateways, proxies and base stations. Example of such environment are wireless office networks, WiFi. Thus these environments may be able to leverage from centralized composition managers to some extent (with appropriate support for mobility, faults). However, infrastructure-less mobile environments (e.g. warfront activities, ad-hoc sensor

[1] we shall use the term *composite service* and *composite request* interchangeably throughout the remaining of the paper. This is because composite request in this work is nothing but a declarative specification of a composite service.

networks, walkways in cities that are created on-the-fly) call for an alternate design approach of service composition architectures.

We focus our work on infrastructure-less mobile environments since we believe that (1) Infrastructure-based mobile environments are just a special case of the infrastructure-less environment where some devices are fixed, resource-rich and have infrastructure support. (2) Solutions for this environment can be as easily adapted to infrastructure-based mobile environments. For the remaining of the paper, we shall use the term *mobile environments (ME)* to mean *infrastructure less mobile environments.*

We have developed a distributed service composition architecture that primarily supports two distributed broker-based protocols for service composition in ME. We leverage our prior work in service discovery [9] and service-centric routing [10] to support discovery and routing underneath our composition protocols. This paper describes the architecture and two associated composition protocols. Our composition protocols are based on the concept of efficient selection of brokers to handle composite requests.

Salient features of our protocols are (1) **Decentralized Control:** Our composition architecture is immune to central point of failure. (2) **Mobility-Dependent Dynamic Composition Model:** Our composition protocols utilize the dynamically changing service topology to select the appropriate *atomic services* in a composite request. (3) **Efficient Utilization of Mobile Resources:** Mobile resources include mobile services, computation power on the devices, memory and battery life. Our composition protocols distribute the composite requests amongst devices in the mobile environment.

2 Background

Service discovery and composition has been studied widely in the context of wired-infrastructure based web services [1]. Systems like eFlow [4], CMI [37], Ninja Service Composition Architecture [18] and Sheng's framework [3] on declarative web service composition address various problems related to service composition in the wired-infrastructure. The architectures for wired-infrastructure service composition are centralized. All composition requests are directed to a preconfigured composition engine that manage the composition. Moreover, these architectures assume that the services are on static nodes. The architectures do not concern themselves with issues relevant in a mobile environment like hop distance of the service from the composition manager, network disconnection, and

current load on the node where the service resides. The underlying network infrastructure is invisible to the composition manager. We argue that in ME, the composition architecture should take the changing network topology (network graph) into account.

There has been very limited work in the development of distributed protocols for service composition targeted towards ME. In prior work [32], we developed a middleware-based architecture to handle composite requests from mobile devices with the help of infrastructure-based services. Our middleware platform sits on the edge of the wired network and serves as a proxy to integrate various web services and present them to mobile users. Limitation of such a system is that it depends only on the wired infrastructure for services. Moreover, middleware-oriented protocols are essentially semi-centralized and hence unsuitable for ME.

Basu et al. [31] have described a hierarchical task-graph based approach to enable service composition in ad-hoc networks. In this 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. Their approach selects the source of the request as the composition manager for itself. Our architecture considers the source of the request and the composition manager as logically separate entities. We believe that the proper selection of the composition manager for a request increases the efficiency of composition (defined in section 7) and decreases the network load. Moreover, their approach employs network intensive global broadcasting techniques whereas, our architecture is based on the strategy of localized search thus reducing the protocol overhead.

There is a plethora of work in trying to define languages to represent services, invocation mechanisms and composite services. This class of work in service composition, nonetheless important is not an issue of this paper. For the purpose of this paper, we assume that the composite service can be represented in any of these languages. Examples of such work include DAML-S[14], OWL [21], WSFL[22], XLANG[13] and BPEL[15]. In particular, we have used DAML-S to represent composite services. This is primarily because DAML-S derives semantically rich reasoning capabilities with the service description from DARPA Agent Markup Language (DAML) [20]. It also offers a rich set of declarative constructs to appropriately represent composite services.

3 Design Rationale

We describe issues that were of concern in the design of our architecture for ME.

- **Distributed Coordination Model:** Service Composition involves the use of a coordinating entity (referred to as broker) that manages the discovery, integration and execution of a composite service. Coordination and management of a composite service in mobile environments (ME) essentially involve discovering the necessary component services, integrating the information obtained from the discovery, invoking the individual services, managing the order of execution and handling faults. Coordination and management need to be primarily adaptable to the topology changes of the mobile environment and handle mobility, network disconnections and other types of node or service failures. However, unlike infrastructure-based composition architecture, ME cannot presuppose the existence of a centralized powerful machine. A centralized composition manager might not be able to access all the necessary services and secondly, a centralized composition manager might become *unreachable* from a device in the ME. Moreover, such an architecture is prone to single point of failure. Hence, composition architectures for ME need a distributed coordination model.

- **Resource Heterogeneity and Awareness:** Devices with varying resources and capabilities exist in the geographical vicinity of one another. A service composition protocol should be able to utilize these varying resources available to it. Resources refer to either services or computation power that are available and vary amongst devices. An efficient composition protocol should ideally be able to compose a service with the most efficient use of the surrounding environment and with minimum network overhead. It should also be able to utilize the spatial distribution of the services (refered to as service topology) and computation resources in the vicinity. In sections 5 and 6, we show how our protocols are able to utilize the service topology for efficient composition.

- **Mobility Management and Adaptability:** Service composition protocols should be able to able to adapt themselves to the changing neighborhood. In particular, it should be able to utilize newly arrived services that were not available before. A composition protocol should be able to select a possible set of services based on their mobility patterns, platform battery life, how long they would be in the vicinity etc. Mobility changes the underlying service topology. Composition protocols for ME should be able to adapt their execution model to

suit the changing service topology.

- **Fault Tolerance and Reliability:** Mobile device often have a short *switched-on* time and often are unable to process remote requests due to system overload or network level disconnections. A robust composition protocol should be tolerant towards such failures and degrade gracefully with increasing service/resource unavailability. Faults may range from network level disconnections, service discovery failures to node failures and service execution failures. The composition architecture should be resilient towards these failures and apply appropriate fault control mechanisms to ensure the processing of the composite request.

- **Efficient Network Utilization:** Network bandwidth is a very important parameter to be considered while designing any composition protocol for ME. By default, service discovery and composition protocols are usually broadcast-based and essentially impose a relatively high network load. An efficient composition protocol should try reducing network wide broadcasts. However, reducing network wide searches may reduce the chances of a service being discovered and hence reduce composition efficiency. Our architecture follows a strategy of greedy localized searches to compose services and tries to conserve network bandwidth.

4 System Model

Our system consists of mobile nodes each providing one or more services that can be invoked by peer devices. We assume that these devices are connected to each other using ad-hoc networking protocols. Resources like computation power, battery life vary from one device to the other. We do not distinguish between a client and a service provider. Composite requests are sent out by multiple nodes. A node sending out a composite request can itself participate in another composition. Figure 1 depicts the system environment. We define some key terms that we would be referring to in the remaining part of the paper:

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 its request.

2. **Service Provider (SP):** Mobile device that contains services that are accessible from other peer nodes.

3. **Composition Manager (CM):** The device that 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 for specifying a composite service. Our specification 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 in order to make the service *atomic*.

7. **Composite Service Length:** Number of distinct atomic services that constitute a composite service.

Figure 2 shows an overview of the components that are resident on each mobile node in Figure 1. We briefly describe the functions of the various layers. This paper primarily describes the mechanisms in the service composition layer. Although our work leverages from our prior work in the lower layers (network and service discovery), our composition protocols can also run on other standard routing and discovery protocols.

Network Layer: The Network Layer encapsulates networking protocols that provide wireless or ad-hoc connectivity to peer nodes in the vicinity. Examples of such network connectivity range from protocols like 802.11(a,b,g) in ad-hoc mode, Bluetooth [27], Infrared to ad-hoc routing protocols like AODV [30], TORA [28], DSDV [29], GSR [10]. In prior work, we developed a routing protocol, Group-based Service Routing Protocol (GSR) [10] for providing efficient routing support below service discovery applications. We have used GSR as our routing protocol. GSR is on-demand and utilizes the path used during service discovery for end-to-end data transmission. We have shown in [10] that integrating routing with service discovery in mobile ad-hoc environments increase system efficiency. Additionally, our routing protocol provides end-to-end session management that detects disconnections, performs session packet buffering, handles retransmissions and facilitates service-centric routing.

Service Discovery Layer: The service discovery layer provides the system with the protocols required in order to discover services residing on nodes in ME. Our composition protocols use an efficient distributed group-based service

discovery protocol (dubbed GSD), described in a separate publication [9] to discover services. We briefly describe the protocol for the sake of completeness.

Standard protocols for service discovery in mobile environments are either request-broadcast based or advertisement-broadcast based [16, 6, 2]. A typical request-broadcast protocol broadcasts a service discovery request globally to all nodes in the network. Conversely, an advertisement-broadcast-based protocol broadcasts advertisements of the local services in each node to all other nodes in the network. These advertisements are cached by the nodes. Apart from the fact that these protocols are inefficient in terms of bandwidth and resource usage (as the request has to be processed by all the nodes including those with limited processing capability and battery power), they impose a huge load on the network [36]. In [9], we introduced GSD for MEs, which reduces global broadcasts to a large extent and efficiently uses network bandwidth for discovering services (if present). GSD is based on the concepts of *peer-to-peer caching of service advertisements in a limited vicinity* and *group-based selective forwarding of discovery requests*. We use and ontology developed using DAML [6, 20] to effectively describe services. Services are classified into a hierarchical group based on their functionalities. We use the class-subclass hierarchy in DAML to model service groups. Each SP (Service Provider) advertises its services to other peer nodes in a *limited* vicinity. Advertisements also include a list of the several service groups that the SP has seen in its vicinity. This information is used to selectively forward discovery requests to SPs. In [9], we show that GSD reduces network-wide broadcasts and achieves increased efficiency in discovering services.

Service Composition Layer: This layer encompasses protocols responsible for carrying out the process of managing the discovery and integration and execution of composite services. We introduce two distributed **reactive** protocols for service composition in ME. Reactive service composition refers to the type of composition that is executed on-demand. These two protocols are the primary contributions of this paper. Both reactive protocols are decentralized, immune to central point of failure and take maximum advantage of the mobility of the nodes, service topology and currently available resources in the vicinity. Both protocols carry out a broker arbitration that decides on the Composition Manager (CM) for a certain request. Thus, each composite request may be assigned a different CM.

The composite request consists of a Description-level Service Flow (DSF) of the composite service formulated using DAML-S. DSF of a composite service contains high-level description of the services needed for the composition

and the execution flow. Figure 3 presents a sampl of a DSF to view a PDF file (by first downloading it and then printing it) in case the client does not have the capability to do so.

CM on receiving a composite request performs the process of service integration where it discovers the required services and constructs an execution-level service flow (ESF). ESF contains specific information (device address, invocation mechanism among others) used to invoke various component services. The service integration is followed by service execution where each service participating in a composition is executed. This layer manages the execution order of the multiple services and monitors faults. The broker arbitration module, the service integration and execution module and the techniques employed within it plays a vital role in making our architecture fundamentally different from static infrastructure-based service composition. We describe our composition protocols in detail in sections 5 and 6.

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 composite request.

5 Dynamic Broker Selection based Protocol

The Dynamic Broker Selection-based protocol is primarily based on the concepts of distributing composite requests to composition managers (CM) distributed in the ME and single-CM execution policy per request. The Request Source (RS) of each composite request carries out a *broker arbitration* to select a CM from the ME. Each mobile device is a potential candidate for CM. The *broker arbitration* consists of controlled broadcasting of solicitation requests to devices in the nearby vicinity of the RS. The *broker arbitration* module in each device replies with information regarding its potential to be a CM for the request. The RS elects the best possible CM from the available devices and sends the Description-level Service Flow (DSF) to it. The elected CM on receiving the request increments its load level (number of executing composite requests) and starts processing it.

The CM performs the following tasks for processing the composite request. (1) It extracts the atomic services from the DSF and uses the underlying service discovery protocol (GSD) to discover the services; (2) It obtains invocation details of the services that has been discovered from GSD and stores them; (3) Once all services have been

discovered, the CM constructs an Execution-level Service Flow (ESF) of the composite service; (4) It then invokes the individual services following the execution flow as specified in the ESF and stores the temporary results; (5) On completion of execution of all atomic services, the CM transmits the final result to the RS; it decrements its load and awaits further messages;(6) During the execution phase, the CM (apart from storing the temporary results) periodically transmits checkpoints consisting of the partial results back to the RS. The RS uses these checkpoints to determine faults.

The RS interprets the loss of checkpoints from the desired CM as an indication of fault. Faults can be triggered due to (1) loss of checkpoint messages; (2) failure to discover a desired service (discovery-level fault); (3) failure to execute a service after having discovered it (execution-level fault); (4) CM might have shut itself down or become unreachable (node failure). Our current prototype signals a fault to the RS in case the above-mentioned events occur. The RS waits for a maximum time limit by which it should have received the final result after which it gives up and restarts the composition. We follow an optimistic strategy to address faults. 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. The battery life and bandwidth spent in computing the request is also lost. In our solution, RS computes another Description-level Service Flow (DSF) of the composite request by utilizing the partial results and pruning the part that has already been executed. The new DSF is treated as a new composite request in the system. The RS maintains a state of its composite requests and merges the results after the new DSF has been executed.

Broker Arbitration: We control the broadcast of solicitation messages by specifying the maximum number of hops for a message. This maximum hop count is increased if an appropriate CM is not found. The solicitation message consists of an enumerated list of atomic services that the composite request needs for its execution. Each device computes a *potential value* for itself that determines its suitability to be the CM for the request and returns it back to the RS. The potential value of a device is calculated by considering its local resources and by estimating the service *richness* of its neighborhood. Service richness is judged by considering the number of remote service advertisements cached by the device. Local resources considered are (1) number of matching local atomic services; (2) battery life; (3) current processing load on the node. Some attributes affect the potential value positively while some affect it negatively (e.g. current processing load). Formally, we denote $U(CM_i)$ as the potential value of each

CM (CM_i) for a composite request S.

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

where $S_c(CM_i)$ is number of service advertisements cached by CM_i; $S_m(CM_i)$ is number of services belonging to the composite request that are present in the cache of CM_i; $L(CM_i)$ is battery life of CM_i; $J(CM_i)$ is current number of requests being processed by CM_i. A Composition Manager, CM_i is selected based on the following equation:

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

We note that for an infrastructure-based mobile environment where the infrastructure may contain a centralized service manager on a server, the arbitration can be adapted by appropriately weighing the various parameters so that it selects the manager as long as it is *reachable* from the RS. This flexibility of our protocol makes it well suited for heterogeneous mobile environments. Figure 4 shows the flow diagram of this phase.

Service Discovery and Integration: Service Discovery and Integration aids in generating an ESF from the given DSF of the composite request using the underlying discovery protocol. Corresponding to each service description in the DSF, an actual atomic service is discovered. The network load during 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 extended to incorporate additional cost factors. An ESF is constructed to contain information on the actual services, its node binding and invocation details. It also contains the service flow (obtained from the DSF). This phase ends when all the required services have been discovered and a new ESF has been formed. Figure 5 describes the pseudo code of the operation of the protocol in this phase.

Service Execution: The CM coordinates the execution of the services in the order specified by the ESF. The Broker uses the underlying routing protocol [10] to transmit results received from one service to another during the execution. The partial results flow from one device to another through the CM. We call it *star-based* execution pattern. In future, we aim to incorporate *mesh-based* execution pattern where the result would directly flow from

one service to the next service. Our checkpoint-based source monitored fault tolerance scheme imposes additional overhead of transmitting checkpoints in the network. We are currently analyzing the effect of our check pointing algorithm on the network bandwidth. We assume that the source node would not fail before successful completion of the composition. This is a reasonable assumption, since the source node would ideally want to keep itself turned on till it receives a final reply.

6 Distributed Broker Selection based Protocol

The Dynamic Broker Selection based protocol suffers from the following limitations: (1)**Single-CM execution policy per request**: Single-CM execution policy per request assumes that a single CM is responsible to manage the entire composition. In ME, mobility often changes the service topology. This, along with other limitations like disconnections and network partition might make it impossible for the current CM to compose the whole request after having completed a partial execution. Ideally, the system should be resilient towards such failures. (2)**Load on RS and CM:** The RS monitors for faults using checkpoints through out the composition process. Environments where the CM often fails to completely execute the service or environments having high number of composite requests impose additional load on the individual RS. The CM has to maintain state information of the entire composite service and monitor the discovery and execution of the composite service. This overhead increases with the increase in the *composite service length*. Moreover, lengthy composite services are more prone to service topology changes and hence more prone to discovery and execution-level faults. This further results in an increase on the load of the RS as well (since it has to monitor more faults now).

Our Distributed Broker Selection based protocol relaxes the single-CM execution policy per request. The central idea behind this protocol is to distribute the task of composing a service to multiple CMs on an as-needed basis. A single composite request is initially assigned to a particular CM by the Requesting Source (RS). The assigned CM executes the request until it encounters a discovery or execution-level fault. In case of a fault, instead of directly notifying the RS, the CM enters a new phase called *arbitration control* phase. In this phase, the CM carries out the following steps: (1) It freezes the current process (*composition state freeze*) and extracts the un-executed part of the composite request and constructs a DSF from it; (2) It carries out a new *broker arbitration* to determine the best possible CM for the pruned request; (3) On successful selection of a new CM, the un-executed DSF together

with the partial result of the composition is transferred to it (*state transfer*); it sends a notification to the RS with the address of the new CM; (4) The new CM resumes the usual composition process with the un-executed DSF; (5) The old CM sends a fault event to the RS only if it fails to either select an appropriate CM or fails to transfer the composition to the new CM. Figure 6 shows the state diagram of the *arbitration control* phase.

This protocol reduces the burden on the RS since the composition transfer is transparent to the RS. In the previous protocol, the RS would have to handle such faults. After the composition transfer, the RS starts receiving checkpoints from a different CM for the same request and figures out that the CM has changed. The new CM could communicate with RS via the old CM or use a different path as determined by the underlying GSR routing protocol. We check for duplicate or spurious checkpoints to ensure consistency of information at the RS. Transparent composition transfer from one CM to another in the ME is inspired by the following observations: (1) Service topology change: Discovery or execution level faults are mostly triggered by service topology change due to mobility. However, except for special cases, it is probable that the required services have not moved too far away from the CM. Thus, even though the CM might not be able to compose the remaining services, some CM in its vicinity might have a better chance of composing the remaining part of the request. (2) Network bandwidth conservation: The old CM may find another CM more easily than the original RS. This is because, if the topology has not changed drastically, then chances of services being around in the vicinity are higher.

Broker Arbitration: We highlight some differences that are possible in the *broker arbitration* in the Distributed Broker Selection based protocol from the previous protocol. Suppose $\bar{\mathrm{S}}$ denotes the DSF of the composite service. Our distributed Broker Selection based protocol can exercise greedy alternatives for CM selection. As an example, the potential values of CMs can be calculated by giving more importance to services needed during the initial stages of the composition as opposed to services needed at a later stage. We recall from section 5 that $S_m(CM_i)$ is number of atomic services belonging to the composite service that are present in the cache of Broker B_i. Unlike in the previous protocol, we envision that the new value of $S_m(CM_i)=\sum V_{S_i}$ where V_{S_i} is height of the node S_i in the task graph[31] of $\bar{\mathrm{S}}$. The new potential value provides more importance towards the discovery and execution of services needed *immediately* as opposed to the services needed at a later stage of the composition. This is more appropriate for lengthy composite services. This would reduce the network load created due to the broadcast of control messages during this phase.

Arbitration Control: Service discovery and integration in this protocol is same as in the Dynamic Broker Selection based protocol. The service execution triggers the *arbitration control* after encountering a discovery-level or an execution-level fault. The maximum number of delegations of a composite request is regulated by the *distribution limit* parameter. The current CM sends a failure notification to the RS if the distribution limit is reached.

Theoretically, the *arbitration control* can be triggered immediately after the execution of each atomic service. However, this generates unnecessary network traffic as the current CM (until it encounters a fault) is capable of executing the next atomic service in the ESF. We trigger the *arbitration control* only when the current CM encounters a discovery or execution-level fault.

- **Composition State Freeze:** The CM monitors the *composition state* of each request it receives. The *composition state* consists of information on the discovery state of each service in the request, execution state of each service, partial results and additional performance parameters like average hop distance of the services to the CM. On a state freeze, discovery and execution requests that had been issued for other services needed in the composition are invalidated. Current CM then constructs a discovery-level service flow (DSF) of the unexecuted part of the composite service.

- **State Transfer and Source Notification:** The CM transfers partial results of the executed part of the composite request as well as a DSF of the unexecuted part to the next CM. The CM notifies the source of the request with the address of the new CM. It invalidates the state that it had maintained for the composite service. The new CM on receiving the composite service creates a composition state by combining the partial results and the un-executed DSF. It then continues composing the un-executed DSF.

The Distributed Broker Selection based Composition protocol offers us the following advantages over the Dynamic Broker Selection protocol. (1) **Better adaptability:** Our new protocol distributes the task of a single request over multiple CMs if needed. Change in the service topology may make a new CM more suitable to compose a request after the composition has started. Our protocol utilizes this and transfers the job to an appropriate CM.

(2) **Load reduction on CM for lengthy composite services:** A single CM does not need to compose the entire service. This imposes lesser load on a single CM since the load gets distributed amongst the participating CMs. (3) **Fault reduction for lengthy composite services:** The RS potentially receives fewer faults since the request transfer between CMs take place transparent to the RS. If the CM transfers the request, it has potentially

discovered a better CM for the request. This design reduces the number of discovery or execution-level faults in the system.

7 Experiments

We implemented our protocol as part of the distributed event-based mobile network simulator Glomosim [41] (version 2.0). We present results comparing our protocols (Dynamic Broker-based Composition and Distributed Broker-based Composition) with each other using various service densities, node mobility and composite service lengths. We also compared our results with a broker-based centralized service composition protocol mostly used for wired networks or infrastructure-based mobile environments. In this protocol, all the composite requests are sent to a fixed composition broker or engine. We call it *Fixed Broker based composition*. For the purpose of our simulation, we placed this broker centrally in the network topology at the beginning of the simulation. We briefly describe the experimental model of the ME that we considered for our simulations.

ME Experimental Model: Our ME model consists of mobile service providers (SPs) connected to each other using an ad-hoc network (190m x 190m area). Each SP can become a CM for a request. The RS is also one of these devices. The devices assume an initial grid-based topology before becoming mobile. We studied the behavior of our protocols under various conditions of service topologies, service densities and node mobility. *Service density* in our model is defined as the percentage of SPs containing one or more of the atomic services required in a composite request. As an illustration, if a composite service consists of services belonging to the set [S1,S2,S3], then a service density of 60% would mean that 60% of the nodes in the system have either service S1 or S2 or S3 or a combination of them. Figure 7 enumerates the specific parameters of the ME experimental model that we used in our simulations.

Evaluation Metrics: We define the following additional evaluation metrics for our simulations.

(1) **Composition Efficiency:** The percentage of composite requests that were successfully composed (discovered, integrated and executed) by the system.

(2) **Broker Arbitration Efficiency:** The fraction of composite requests for which a CM was assigned and the task was received by the assigned CM. Due to disconnections and network topology change in ME, a CM might fail to receive a task from the RS. This metric measures the efficiency of the *Broker Arbitration Phase* in deciding and

actually successfully delivering the task to the CM.

(3) **Broker Instantiation Time (BIT):** The amount of time taken by a source to decide on a CM and delegate the task to it. It includes the overhead for carrying out the *broker arbitration* and transfering the task to the CM.

(4) **Composition Radius (CR):** The average number of hops needed by a CM (or a group of CMs) in order to discover the SPs and finally execute a composite service. A low value of composition radius means that most services were discovered in the nearby vicinity. This further implies that our *broker arbitration* selects the CM appropriately with respect to the service topology.

(5) **Distribution Index (DI):** This parameter defined only in the context of Distributed Broker-based Composition. It is defined as the number of composite request delegations required to completely compose a service. It is restricted by the *distribution limit* of the system.

Simulation Results: The experiments being reported correspond to a set of 100 composite requests for 3 different seeds of random way-point mobility pattern. Figure 8 shows the effect of service density on the Composition Efficiency. We observe that our distributed composition protocols clearly outperform the centralized service composition protocol as we expected based on our arguments in 3 and the adaptability of our protocol. We also see that composition efficiency increases with increasing service density. This is because required services are more easily available to the protocol. Moreover, our protocols utilize the service topology to perform the composition efficiently. We observe that composition efficiency of Centralized Service Composition decreases drastically with increase in the composite service length (observe the absolute values of the efficiencies in the 3 graphs in figure 8). Our protocols do not have appreciable decrease in the efficiency. We also observe that in general, the Distributed Broker Selection based Composition performs better than Dynamic Broker-based Composition in terms of composition efficiency. This is mostly because the former protocol does not have the single-CM execution policy and hence adapts itself better to the changing service topology. We also note that sometimes, due to the changing environment, the *broker arbitration* might not be successful in finding an appropriate CM (as indicated by the slight dip in the graph for composite service length=3 in figure 8). This is because, the environment beside the CM might change by the time the task assignment has reached it.

Figure 9 shows the comparison of Broker Arbitration Efficiency between the three protocols. We observe that Arbitration Efficiency is in general much higher in our protocols. This is mostly because, the CMs are selected from

nearby nodes in our protocols. Thus chances of a composite request not reaching its assigned CM are low. This contributes to the increase in the Arbitration Efficiency. In Centralized Service Composition, each composite request has to reach a fixed composition engine that is potentially multiple hops away. Hence chances of message losses are more. In terms of absolute measures, the efficiency observed in our protocols is well above 0.9. We also observe that generally the efficiency of Dynamic Broker Selection based Composition is fractionally higher than Distributed Broker Selection based Composition. We believe this is due to the network load generated due to greater number of *broker arbitrations* in the Distributed Broker Selection based Composition. We note that the broker arbitration efficiency does not show any particular relation to the service density in centralized service composition. This is expected since the service density does not affect the probability of a composite request reaching the selected CM.

Figure 10 shows how the Broker Instantiation Time (BIT) of different protocols compare with each other. We observe that the value of BIT is very high for Centralized Service Composition. This is attributed to the high end-to-end delivery delay across multiple hops in a ME. On the contrary, we observe that the BIT of our composition protocols is significantly lower. In our protocols, the RS selects CMs that are in the nearby vicinity and hence have much shorter routes. This signifies the need for composition protocols in ME to be able to utilize resources in the surrounding vicinity. We also see that the BIT does not show any definite relationship with respect to the increasing service density. This is because the service density only helps in deciding a CM. Since we have uniform service density, it does not have any implication on how far the CM is from the RS.

Figure 11 shows the comparison of the Composition Radius (CR) for the Dynamic Broker Selection based and Distributed Broker Selection based composition protocols. The CR observed during our experiments is in general lower for Distributed Broker Selection based Composition (except for composite service length of 7 where it is mostly higher). This is mostly due to our Brokerage Delegation mechanism. A CM transfers its brokerage to another (potentially better placed) CM on an unsuccessful discovery attempt. It does this instead of increasing its search span and trying again. This results in a lower CR for the Distributed Broker Selection based Protocol. We also observe contradictions where we see that the CR is fractionally higher for Distributed Broker Selection based Composition (composition length=7). We believe, it may be due to "inefficient" decisions made by our Broker Arbitration Phase or due to the dynamic change in service topology. The service topology might have changed in an unfavorable way in the time gap between during the arbitration period and the actual delivery of the composite request to the CM.

We believe that the composition radius for our distributed broker selection based protocol is higher for composite service length of 7 because the broker arbitration efficiency (figure 9) is lower. Thus many CMs actually increase their broadcast limit to compose required services.

We observe that the average composition radius decreases with increasing service density. This is because with increasing service density, hop count to discover services decrease. However, we observe an increase in the radius for Dynamic Broker Selection based protocol (by 0.1) with composition length of 3. This is due to a higher number of data points and some outliers in the data set for the service density of 40. Figure 12 shows the data distribution of the actual values, means of which have been plotted in figure 11. We observe that the median values (center lines at the boxes) and the range of data are very similar for densities of 20 and 40 % and decreases with further increase in the service density. This supports our claim that our protocol adapts itself with increasing service density to reduce the amount of network load due to service discovery.

We observe in figure 13 that the number of services that are discovered locally by the underlying GSD service discovery protocol is consistently high for Distributed Broker-based Composition. Thus the discovery protocol imposes lesser load on the network since more services can be discovered locally. This result shows that our Distributed Broker Selection based Composition achieves better utilization of the service topology. The Centralized Service Composition imposes maximum load on the network since most of its services are not available locally.

We measured the effect of service density on the average number of composite request delegations (viz. Distribution Index) for our Distributed Broker Selection based Composition. The results are plotted in figure 14. We observe that in a less dense service topology, the CM has to do a large number of request delegations. However, as the density increases, the number of such delegations decrease since more services are available near a CM. We also observe that as expected, the distribution index increases with increase in the composition length.

8 Future Work and Conclusions

We have presented distributed Service Composition protocols for mobile environments. Our protocols are decentralized and utilize the service topology to compose services. Each composite request is independently assigned a Composition Manager (CM). The *Broker Arbitration* mechanism uses a controlled broadcast-based scheme for soliciting information from nearby nodes. Selection of a CM depends on a device-specific potential value and takes into

account services present in the device, 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 CM. However, our Dynamic Broker Selection based protocol is based on the concept of single Composition Manager per request policy. Our Distributed Broker Selection based protocol uses multiple CMs to execute a request. Both protocols are immune to central point of failure and do not require infrastructure support.

We compared our protocols to a centralized solution (used in wired and WiFi/802.11 environments) where requests are sent to a central broker in the system. Simulation results show that our protocols perform better in terms of composition efficiency and broker arbitration efficiency. Our protocols achieves better results in terms of composition radius and utilizing service topology. We also show that judicious placement strategy of the CMs in our protocols impose lesser load on the underlying service discovery and network layers. In general, the Distributed Broker Selection based protocol performs better than Dynamic Broker Selection based protocol in terms of composition efficiency and composition radius.

Even though service composition is a well recognized problem in the domain of wired-infrastructure services, it has not been explored in the domain of mobile environments. There are multiple possible extensions to our current proposed protocols. Our architecture does not impose any dependence on the task-graph of the composite service. However, various optimizations might be possible if we take the task-graph into consideration. For example, a composite service might require a few services to be executed in parallel. It might be possible to enhance our protocol to start *parallel* broker arbitrations to handle parallel service flows in the task graph. Duplication of service instances can be taken into consideration while optimizing the discovery and execution cost of the composite service. We have used a checkpoint-based fault-tolerance strategy. However, it might be possible to vary the frequency of checkpoints according to the mobility of the environment. We plan to continue our research in this direction to realize more efficient distributed service composition protocols for mobile environments.

References

[1] Web Services Description Language 1.1. World Wide Web, `http://www.w3.org/TR/wsdl12`.

[2] The Salutation Consortium Inc 1999. Salutation Architecture Specification (Part 1), Version 2.1 Edition. World Wide Web, `http://www.salutation.org`.

[3] Marlon Dumas Boualem Benatallah, Quan Z. Sheng. The Self-Serv Environment for Web Services Composition. In *IEEE Internet Computing 7(1)*, pages 40–48, 2003.

[4] 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.

[5] 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.

[6] 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.

[7] Harry Chen, Anupam Joshi, and Timothy Finin. Dynamic service discovery for mobile computing: Intelligent agents meet jini in the aether. *Baltzer Science Journal on Cluster Computing, Special Issue on Advances in Distributed and Mobile Systems and Communications*, 2001.

[8] Tim Finin Deepali Khushraj and Anupam Joshi. Semantic Tuple Spaces: A Coordination Infrastructure in Mobile Environments. In *Poster paper, Second International Semantic Web Conference (ISWC)*, 2003.

[9] 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.

[10] 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.

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

[12] K. Erol, J. Hendler, and D. Nau. HTN Planning: Complexity and Expressivity. In *Proc. AAAI.*, 1994.

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

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

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

[16] Sumi Helal, Nitin Desai, and Choonhwa Lee. Konark- A Service Discovery and Delivery Protocol for Ad-hoc Networks. In *Proc. Third IEEE Conference on Wireless Communication Networks (WCNC), New Orleans*, March 2003.

[17] T. Hodes and Randy Katz et. al. An Architecture for a Secure Service Discovery Service. In *Proc. Fifth International Conference of Mobile Computing and Networks, Seattle WA*, August 1999.

[18] 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.

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

[20] DARPA Agent Markup Language. World Wide Web, `http://www.daml.org`.

[21] Ontology Web Language. World Wide Web, `http://www.w3.org/TR/2002/WD-webont-req-20020307`.

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

[23] Q. H Mahmoud. A Mobile Agent-based Approach to Web-based Distributed Computing. In *In Dimopoulos, N.J.; Li, K.F. (Eds.) High Performance Computing Systems and Applications. Kluwer Academic Publishers*, 2002.

[24] Alisha D. Malloy, Upkar Varshney, and Andrew P. Snow. Supporting Mobile Commerce Applications using Dependable Wireless Networks. *ACM Journal on Mobile Networks and Applications (MONET)*, 7(3):225–234, 2002.

[25] Naveen Srinivasan Massimo Paolucci, Anupriya Ankolekar and Katia Sycara. The DAML-S Virtual Machine. In *Proc. 2nd International Semantic Web Conference (ISWC)*, October 2003.

[26] David Mennie and Bernard Pagurek. An Architecture to Support Dynamic Composition of Service Components. Systems and Computer Engineering. Carleton University, Canada.

[27] Bluetooth White Paper. World Wide Web, `http://www.bluetooth.com/developer/whitepaper`.

[28] Vincent D. Park and M. Scott Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proc. IEEE INFOCOM.*, april 1997.

[29] C.E Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing(DSDV) for Mobile Computers. *Comp. Commun. Rev.*, pages 234–44, October 1994.

[30] C.E. Perkins and E.M Royer. Ad-hoc On-Demand Distance Vector Routing. In *Proc. 2nd IEEE Workshop. Mobile Computing Systems and Applications*, pages 90–100, February 1999.

[31] 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.

[32] 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.

[33] 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.

[34] Olga Ratsimor, Dipanjan Chakraborty, Anupam Joshi, and Timothy Finin. Service Discovery in Agent-based Pervasive Computing Environments. *MONET Special Issue on Mobile and Pervasive Commerce*, 2003.

[35] Rekesh John. UPnP, Jini and Salutaion - A look at some popular coordination framework for future network devices. Technical report, California Software Labs, 1999. Available online from.

[36] Y. Chen S. Ni, Y. Tseng and J. Sheu. The Broadcast Storm Problem in a Mobile Ad-hoc Network. In *Proc. MOBICOM. Seattle. Washington*, 1999.

[37] 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.

[38] 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.

[39] Jeffrey Undercoffer, Filip Perich, Andrej Cedilnik, Lalana Kagal, Anupam Joshi, and Tim Finin. A Secure Infrastructure for Service Discovery and Management in Pervasive Computing. *To appear in ACM MONET : The Journal of Special Issues on Mobility of Systems, Users, Data and Computing*, 2002.

[40] Upkar Varshney and Ron Vetter. Mobile Commerce, Framework, Applications and Networking Support. In *ACM/Kluwer publishers. Journal on Mobile Networks and Applications (MONET)*, June 2002.

[41] 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.

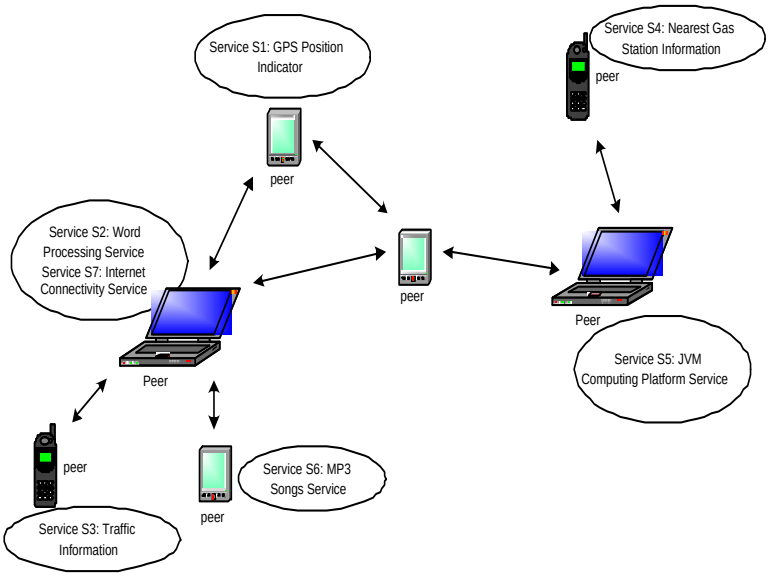

Figure 1: Infrastructure-less Service Composition Environment

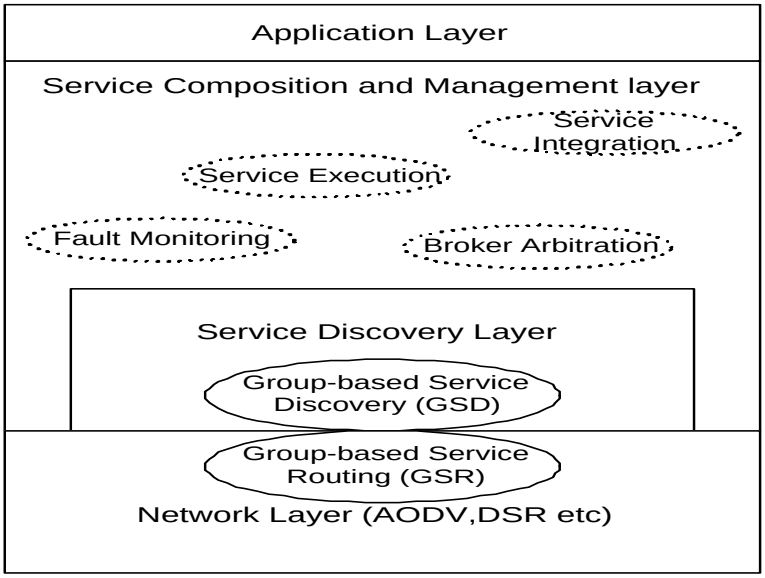

Figure 2: System Architecture

```
<rdf:RDF
  xmlns:rdf= "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs= "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:daml= "http://www.daml.org/2001/03/daml+oil#"
  xmlns:process="&DamlProcess;#">

<daml:class rdf:ID="ViewFileComposite">
 <rdfs:subClassOf rdf:resource="&DamlProcess;#CompositeProcess" />
 <rdfs:subClassOf>
   <daml:Restriction>
      <daml:onProperty rdf:resource="&DamlProcess;#composedOf" />
        <daml:oneOf rdf:parseType="daml:collection">
          <daml:Class>
              <daml:intersectionOf rdf:parseType="daml:collection">
                  <daml:Class rdf:about="process:Sequence" />
                  <daml:Restriction>
                    <daml:onProperty rdf:resource="&DamlProcess;#components" />
                        <daml:toClass>
                           <daml:Class>
                             <daml:listOfInstancesOf rdf:parseType="daml:collection">
                                <daml:Class rdf:about="#Filedownloader" />
                                <daml:Class rdf:about="#Printer" />
                             </daml:listOfInstancesOf>
                           </daml:Class>
                        </daml:toClass>
                  </daml:Restriction>
              </daml:intersectionOf>
        </daml:Class>
        <daml:Class>
          ......
</daml:Class>
      </daml:oneOf>
   </daml:Restriction>
 </rdfs:subClassOf>
</daml:class>
</rdf:RDF>
```

Figure 3: Description-level Service Flow (DSF) to compose a Printer service and a File Download service

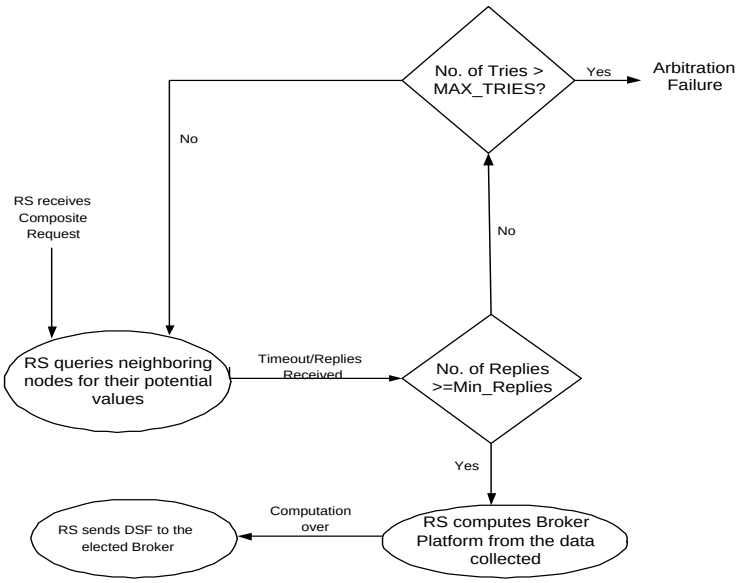

Figure 4: Flow Diagram of the *Broker Arbitration* phase in the Dynamic Broker Selection based Protocol

```
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++;
      }
   }
}
```

Figure 5: Pseudo code of Service Discovery and Integration Phase

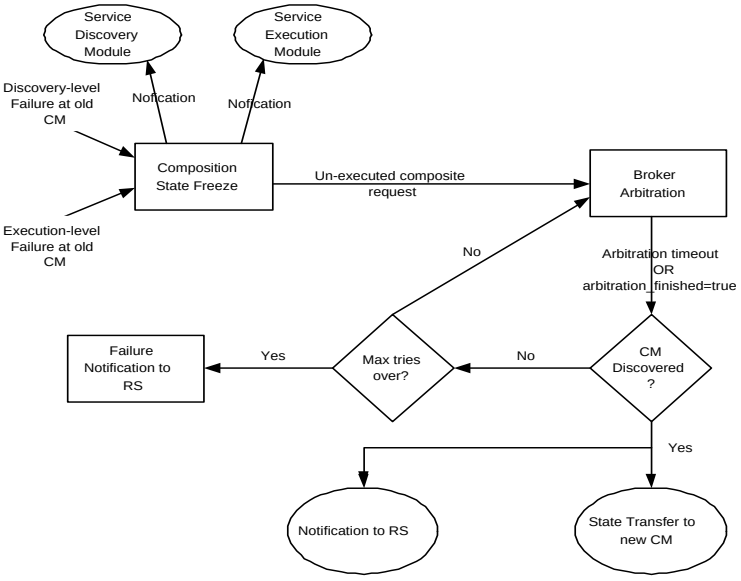

Figure 6: State Diagram of the *Arbitration Control* phase in the Distributed Broker Selection based Composition

Duration	3600 seconds (1 hour)
Space (x,y)	(190 x 190m)
Tx Range (Transmission Range)	30m
Tx Throughput	20kbps
No. of Nodes	64
Advertisement Interval	10 seconds
Advertisement Timeout	5 seconds
broadcast jitter	10 milliseconds
Mobility	Random way-point with 3 m/s speed and 5 s stoppage time
Initial topology	grid topology with nodes equally spaced out in (x,y)
Service density	20% to 100%
Composite service length	3,5,7
Broker Arbitration MAX_HOP_COUNT	1
MAX_RETRIES to discover an atomic service	3
MAX_CM_DELEGATIONS (*distribution limit*)	7

Figure 7: ME experimental model parameters

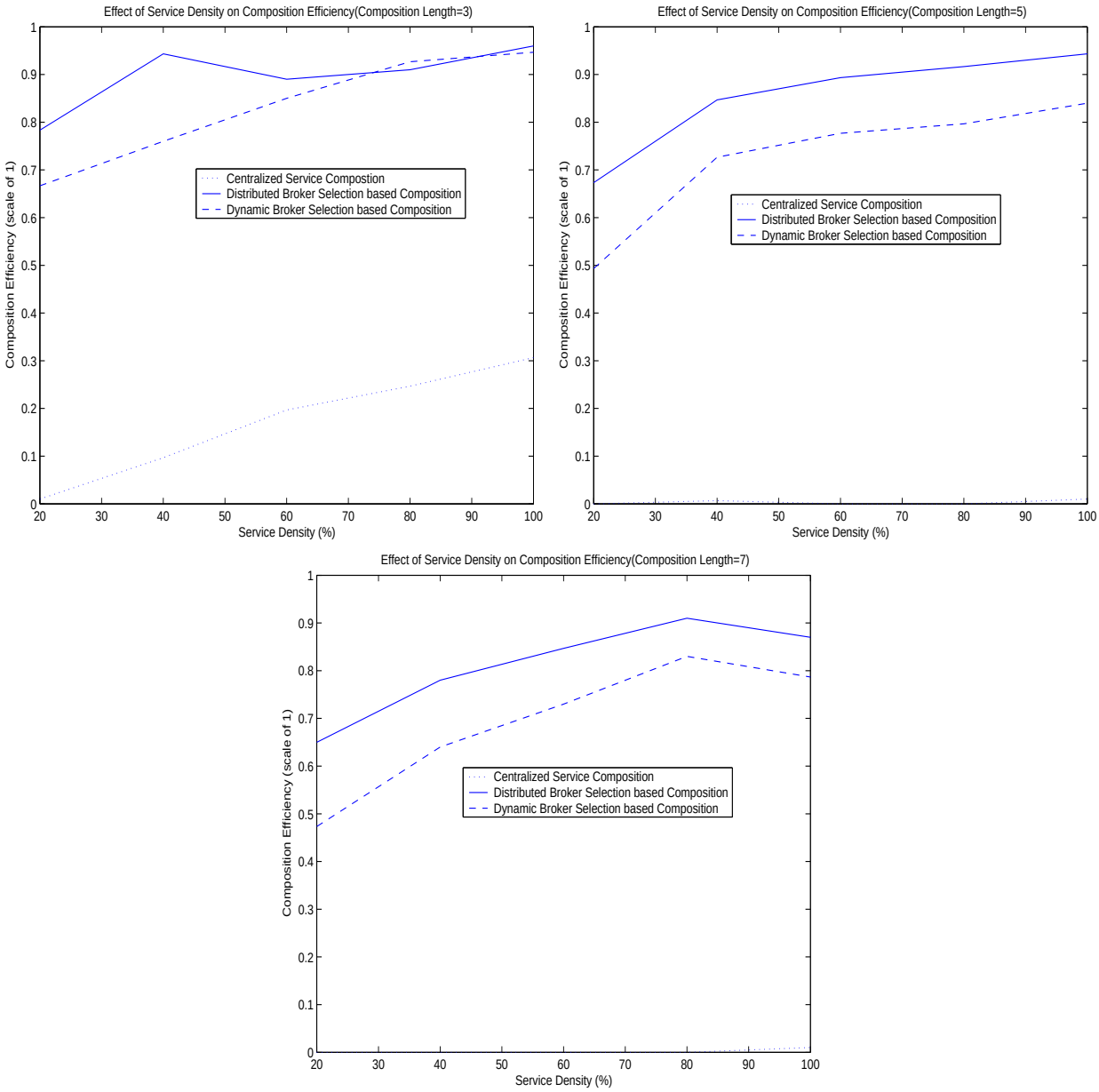

Figure 8: Comparison of Composition Efficiency of the different protocols

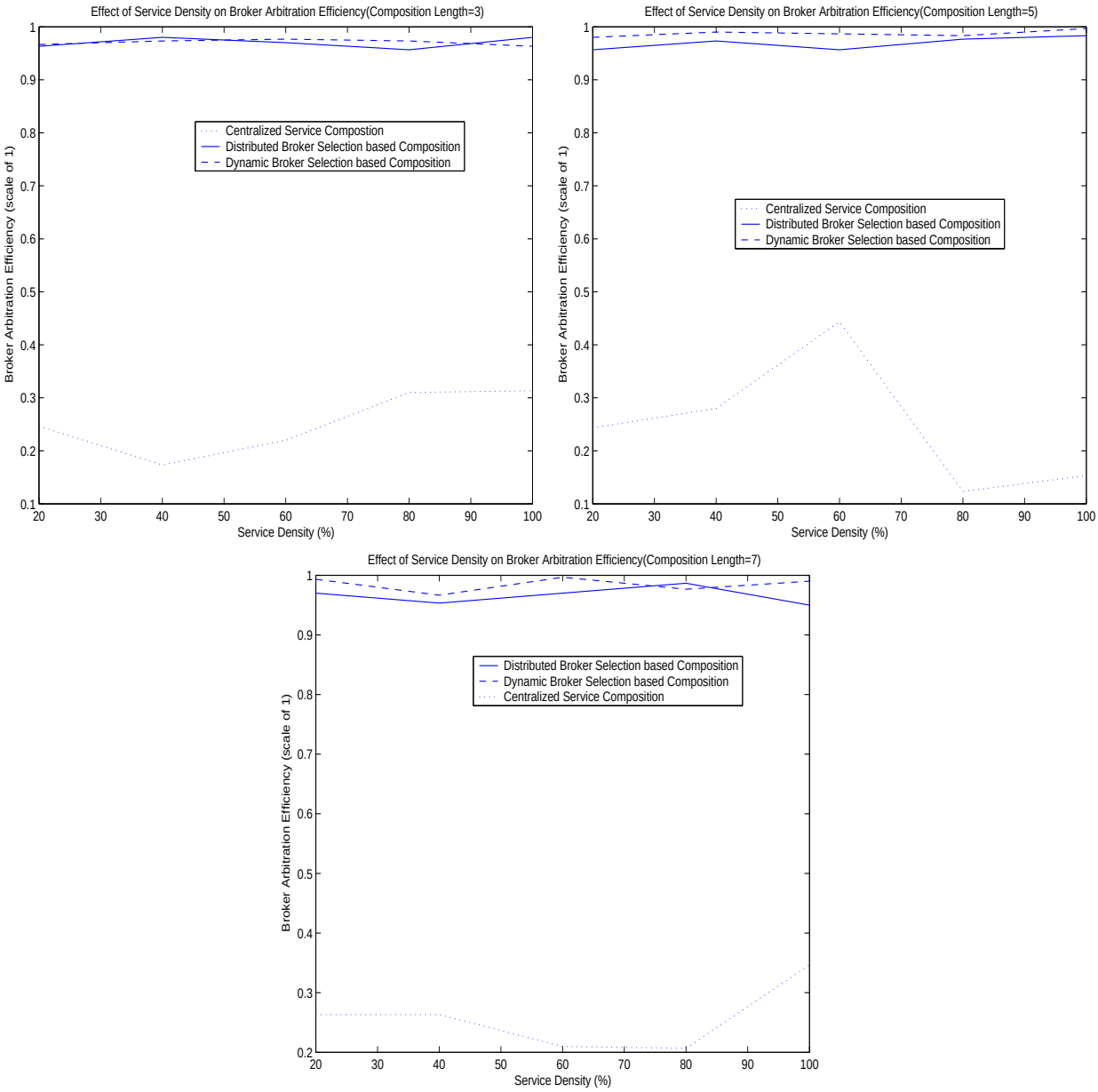

Figure 9: Comparison of Broker Arbitration Efficiency of the different protocols

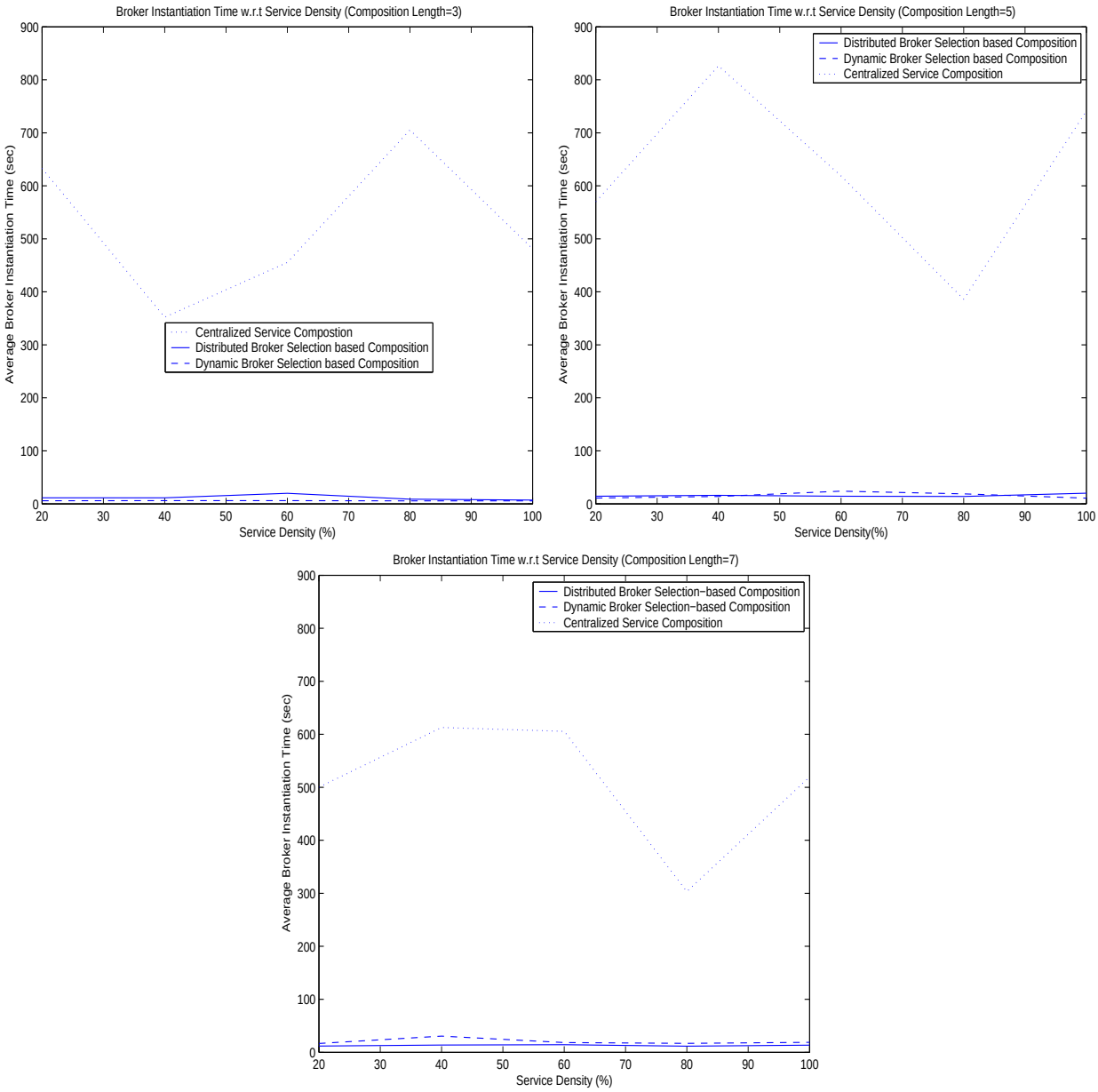

Figure 10: Broker Instantiation Time of the different protocols

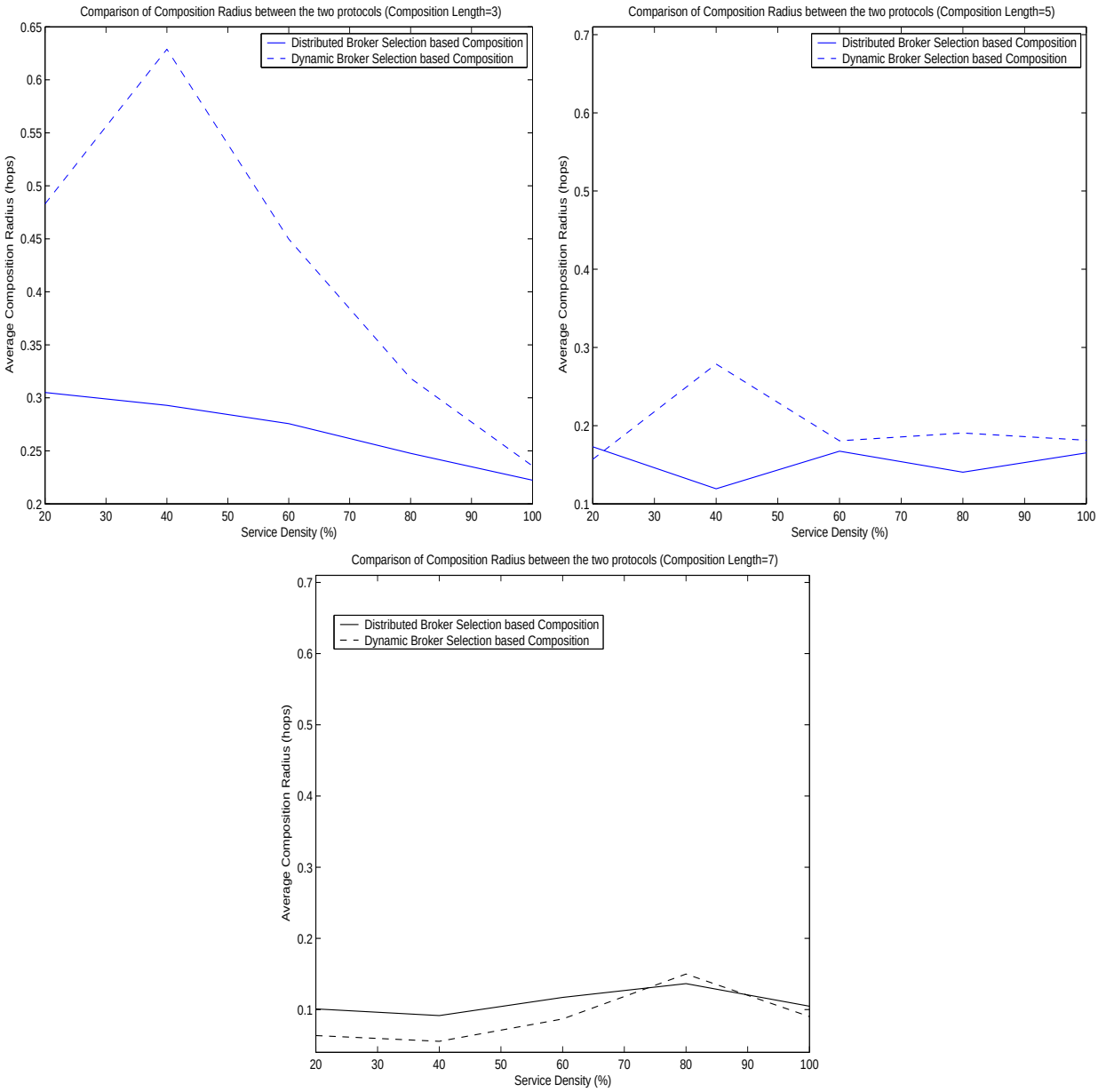

Figure 11: Comparison of Composition Radius of the different protocols

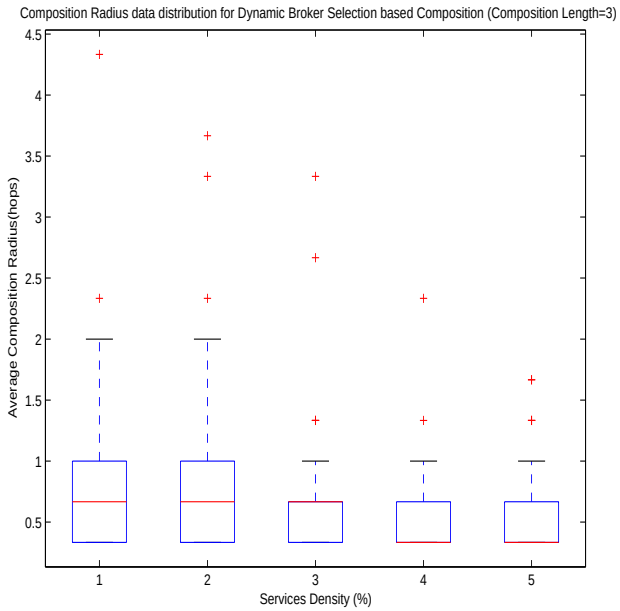

Figure 12: Data Distribution of composition radius to explain the distribution of mean composition radius for Dynamic Broker Selection based composition for composition length=3

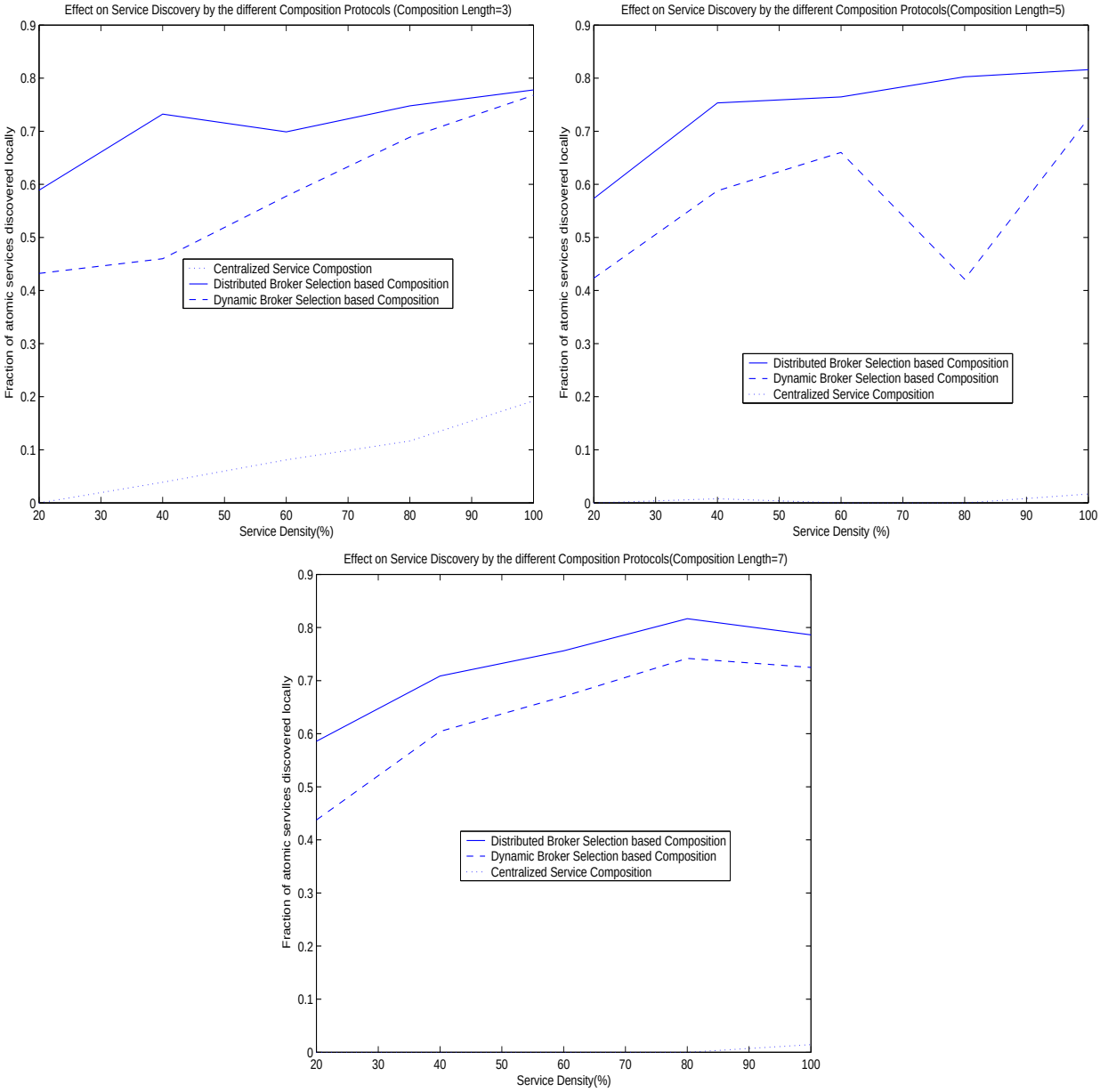

Figure 13: Efficiency of our protocols in terms of locally discovered services

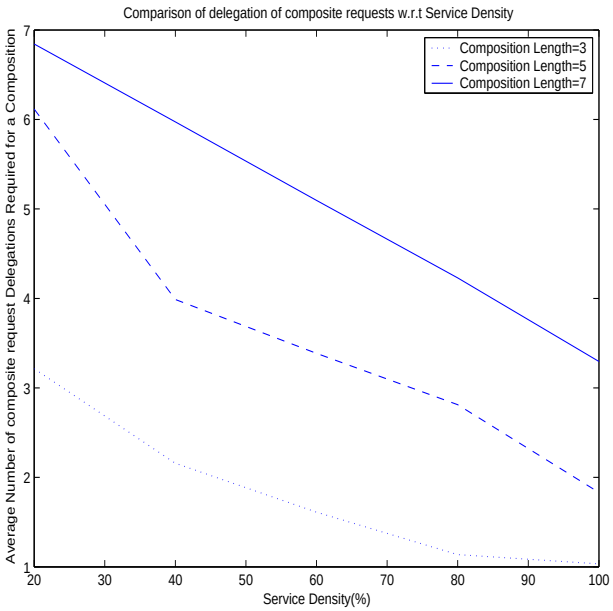

Figure 14: Effect of Service Density on the number of composite request Delegations for Distributed Broker-based Composition